

“十二五”高等院校规划教材



内含源代码和课件

CPLD/FPGA

设计与应用 高级教程

郭利文 邓月明 编著
黄智伟 主审



北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

“十二五”高等院校规划教材

CPLD/FPGA 设计与应用高级教程

郭利文 邓月明 编著
黄智伟 主审

北京航空航天大学出版社

内 容 简 介

本书结合目前主流的 CPLD/FPGA 产品以及最流行的设计理念,系统、详细地介绍 CPLD/FPGA 的硬件结构、硬件描述语言与验证语言的基础应用以及高级应用;详细介绍如何使用 Verilog HDL 语言进行有限状态机设计和 testbench 设计,以及如何使用 Modelsim 进行功能仿真和时序仿真;简要介绍验证方法学的基本概念以及验证语言的比较,并就 CPLD/FPGA 的系统应用进行了详细探讨,包括 DSP 设计、嵌入式处理器设计、HardCopy 设计、嵌入式逻辑分析仪的使用以及 CPLD/FPGA 的板级设计。

本书既可作为电子信息、通信工程以及相关工科专业的本科高年级学生和研究生教材,也可作为全国大学生电子设计竞赛的培训教材,以及从事电子电路系统设计与 CPLD/FPGA/ASIC 设计的工程技术人员的参考用书。

图书在版编目(CIP)数据

CPLD/FPGA 设计与应用高级教程 / 郭利文, 邓月明编

著. —北京: 北京航空航天大学出版社, 2011. 1

ISBN 978 - 7 - 5124 - 0246 - 1

I. ①C… II. ①郭… ②邓… III. ①可编程序逻辑器件—教材 IV. ①TP332.1

中国版本图书馆 CIP 数据核字(2010)第 209231 号

版权所有,侵权必究。

CPLD/FPGA 设计与应用高级教程

郭利文 邓月明 编著

黄智伟 主审

责任编辑 冯 颖

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) <http://www.buaapress.com.cn>

发行部电话:(010)82317024 传真:(010)82328026

读者信箱: bhpres@263.net 邮购电话:(010)82316936

北京市松源印刷有限公司印装 各地书店经销

*

开本: 787×1092 1/16 印张: 20 字数: 512 千字

2011 年 1 月第 1 版 2011 年 1 月第 1 次印刷 印数: 4 000 册

ISBN 978 - 7 - 5124 - 0246 - 1 定价: 39.00 元(含光盘 1 张)

前言

本书基于当前主流的 CPLD/FPGA 器件及最流行的设计理念,根据作者多年来的实际设计经验,系统介绍了 CPLD/FPGA 的硬件结构、硬件描述语言与硬件验证语言的基础应用以及高级应用,详细介绍了如何在工程中利用 Verilog HDL 语言进行有限状态机设计和 Testbench 设计,以及如何使用 Modelsim 进行功能仿真和时序仿真;通过相关实例具体阐述了怎样实现 SOPC 的设计以及 Nios II 的应用开发,并就 CPLD/FPGA 的系统应用进行了详细探讨,包括 DSP 设计、嵌入式处理器设计、HardCopy 设计、嵌入式逻辑分析仪的使用以及 CPLD/FPGA 的板级设计,从而很好地满足了可编程逻辑器件工程应用整个流程的知识需要。全书实例丰富、图文并茂,由浅入深、由易到难详细介绍了 CPLD/FPGA 的设计与应用。

全书共分 11 章:第 1 章简要介绍了 CPLD/FPGA 的发展历程、设计语言与设计方法、主要产品以及前景展望,介绍了硬件语言和软件语言的区别与联系,重点阐述了 CPLD/FPGA 的设计、验证流程及其注意事项;第 2 章分别从传统和最新 CPLD/FPGA 的硬件结构阐述了乘积项和查找表的基本原理以及 CPLD/FPGA 的选择指导,重点讲述了最新 CPLD/FPGA 的硬件结构(这是本书紧扣最新技术发展的体现之一);第 3、4 章主要讲述了目前工程界中使用最为广泛的 Verilog HDL 硬件描述语言,从工程实践的角度具体介绍了它的基本语法及其主要特点,重点讲述了 Verilog HDL 语言的高级编程应用以及参数化设计;第 5 章通过实例说明着重阐述了有限状态机的特点、设计和应用——包括目前最流行的设计方法;第 6 章简单地介绍了约束的基本概念以及约束的方式应用,重点阐述了在 CPLD/FPGA 设计中的时延设计与分析;第 7 章是对前面几章的概括,总结了采用硬件描述语言进行 CPLD/FPGA 设计时的一些基本设计原则和技巧,特别提出了组合逻辑和时序逻辑设计设计时的注意事项,以及代码风格的重要性;第 8 章主要讲述了仿真原理、Testbench 设计以及如何通过采用 Modelsim 软件来实现仿真;第 9 章主要就验证与测试进行了探讨,简要介绍了 HVL(硬件验证语言)和断言的重要性;第 10 章就 CPLD/FPGA 的高级应用(包括 DSP、嵌入式 CPU、嵌入式逻辑分析仪等方面的知识)进行分析阐述,通过相关的实例详细说明了 DSP 和 SoPC 的相关应用;第 11 章是对整个 CPLD/FPGA 设计的总结,重点阐述了 CPLD/FPGA 的系统应用,包括信号完整性、电源完整性、功耗与热设计、PCB 设计等方面的内容。

与其他教材相比,本书的主要特点体现在以下 5 个方面:

① 内容新 知识体系以及技术设计理论都是近年来最新、最常用的,实例所使用的实验软硬件平台也都是最近两年三大公司的主流平台;

前 言

② 技术实用 全书侧重实例讲解以及应用设计,其中的实例都是从工程实践中提取的,读者可直接使用;

③ 适用面广 一方面指读者面广,适用于高校学生和工程技术人员,另一方面是指教材中的程序实例大部分适用于 3 个主流可编程逻辑器件公司的软硬件平台(个别实例是平台相关的,已在书中特别指出);

④ 适合教学 通过加入大量的短小实例使读者及时理解消化所讲的理论知识点,保证知识点教与学的完整性,同时在每章最后一节加入一个较完整的实例,使得读者能对本章内容进行系统性的掌握和应用;

⑤ 知识点全面 将产品开发所需的基础知识、编程技巧、开发调试、验证测试及最终系统设计的整套技术流程都进行了介绍,特别是 CPLD/FPGA 的验证和系统开发方面的知识目前大多数 CPLD/FPGA 书籍中很少涉及,但对于系统设计来说又是必不可少的部分,这也是本书的重点和主要特色之一。

本书另附配套光盘 1 张,提供了本书的多媒体课件以及书中所有示例的设计源代码,供读者参考。

全书由郭利文、邓月明编写,由黄智伟教授主审。林韦成(中国台湾)、王康斌、高芳莉和邱树林等工程师为本书的编写付出了诸多努力,提供了许多详细的建议和第一手工程实践资料;Tomonori Hirai 博士(美国)、Jyhming Zhong 博士(美国)和 Mario Lee 博士(美国)等在信号完整性、高速信号与 PCB 设计、功耗与热设计等方面给予了大量的指导;成都电子科技大学楚佩斯、湖南师范大学胡强、贺洪平、王赞、李青、刘斌、欧阳亚、李慧琳、张谦、袁纯、潘律、唐波以及中南大学肖成等同学也为本书的编写做了大量的工作,从而促成了本书的迅速问世。在此一并表示衷心的感谢。同时还要感谢湖南省师范大学青年基金项目课题组(编号:71003)和教学改革研究项目课题组对本书出版的支持。

在本书的编写过程中,作者参考了大量的国内外著作和资料,听取了多方面的宝贵意见和建议,在此对他们致以衷心的感谢。

由于作者水平有限,书中错误和不足之处在所难免,敬请各位读者批评斧正。

作 者

2011 年 1 月

目 录

第 1 章 概 述	1
1.1 数字电路基础及发展演变	1
1.2 CPLD/FPGA 的介绍	2
1.3 设计语言及其方法的介绍	3
1.4 硬件语言与软件语言的区别	4
1.5 设计与验证流程	5
1.6 CPLD/FPGA 的前景与展望	6
1.7 本章小结	9
1.8 思考与练习	9
第 2 章 CPLD/FPGA 硬件结构	10
2.1 PLD 的分类	10
2.2 乘积项结构的基本原理	12
2.3 查找表结构的基本原理	14
2.4 传统 CPLD 的基本结构	14
2.5 传统 FPGA 的基本结构	18
2.6 最新 CPLD 的基本结构	21
2.7 最新 FPGA 的基本结构	25
2.8 CPLD 与 FPGA 的选择	27
2.9 CPLD/FPGA 的配置	27
2.10 本章小结	28
2.11 思考与练习	30
第 3 章 Verilog HDL 语法基础	31
3.1 Verilog HDL 的特点	31
3.2 Verilog HDL 的描述方式	32
3.3 模块和端口	32
3.4 注 释	34

目 录

3.5	常量、变量与逻辑值	34
3.6	操作符	35
3.7	操作数	38
3.8	参数指令	38
3.9	编译指令	38
3.10	系统任务和系统函数	39
3.11	实例 1: 串并转换程序设计	41
3.12	本章小结	43
3.13	思考与练习	43
第 4 章	Verilog 的描述与参数化设计	45
4.1	数据流描述	45
4.1.1	数据流	46
4.1.2	连续赋值语句	46
4.1.3	延 时	48
4.2	行为级描述	48
4.2.1	initial 赋值语句	48
4.2.2	always 赋值语句	49
4.2.3	时序控制	51
4.2.4	语句块	54
4.2.5	过程赋值语句	55
4.2.6	过程性连续赋值语句	60
4.3	结构化描述	60
4.3.1	实例化门	61
4.3.2	实例化其他模块	61
4.4	高级编程语句	63
4.4.1	if... else 语句	63
4.4.2	case、casex、casez 语句	65
4.4.3	for 语句	69
4.4.4	while 语句	69
4.4.5	forever 语句	71
4.4.6	repeat 语句	71
4.5	参数化设计	72
4.6	混合描述	75
4.7	实例 2: I ² C Slave 控制器的设计	77
4.7.1	I ² C 总线简介	77
4.7.2	I ² C Slave 可综合代码设计	77
4.8	本章小结	87
4.9	思考与练习	87

第 5 章 有限状态机设计	89
5.1 有限状态机的基本概念	89
5.1.1 Moore 型状态机	90
5.1.2 Mealy 型状态机	90
5.1.3 状态机的描述	90
5.2 状态机描述的基本语法	91
5.3 状态编码	93
5.3.1 二进制码(Binary 码)	93
5.3.2 格雷码(Gray 码)	93
5.3.3 独热码(one-hot 码)	94
5.3.4 二—十进制码(BCD 码)	95
5.4 状态初始化	96
5.5 Full Case 与 Parallel Case	97
5.6 状态机的描述	100
5.6.1 一段式状态机	100
5.6.2 两段式状态机	103
5.6.3 三段式状态机	105
5.6.4 小 结	108
5.7 实例 3: PCI Slave 接口设计	109
5.7.1 PCI 协议简介	109
5.7.2 PCI Slave 可综合代码设计	112
5.8 本章小结	119
5.9 思考与练习	119
第 6 章 约束与延时分析	120
6.1 约束的目的	120
6.2 引脚约束及电气标准设定	121
6.2.1 引脚约束文件	121
6.2.2 代码注释约束	122
6.3 时序约束的基本概念	124
6.3.1 路 径	125
6.3.2 时序约束参数	128
6.4 时序约束的本质	130
6.5 静态延时分析	131
6.6 统计静态延时分析	132
6.7 动态延时分析	133
6.8 实例 4: 建立时间和保持时间违例分析	133
6.9 时序违例及解决方式	134

目 录

6.10	实例 5: 四角测试中的时序分析	135
6.11	实例 6: LPC Slave 接口设计	136
6.11.1	LPC 协议简介	136
6.11.2	LPC Slave 可综合性代码设计	139
6.11.3	LPC 协议约束设置	144
6.12	本章小结	144
6.13	思考与练习	144
第 7 章	RTL 设计原则及技巧	146
7.1	RTL 设计的主要原则	146
7.1.1	硬件原则	146
7.1.2	面积与速度原则	147
7.1.3	系统原则	147
7.1.4	同步原则	148
7.2	RTL 设计的主要技巧	148
7.2.1	乒乓操作	148
7.2.2	流水线操作	149
7.2.3	资源共享操作	150
7.2.4	逻辑复用操作	153
7.2.5	串并转换操作	153
7.2.6	异步时钟域数据同步化操作	153
7.2.7	复位操作	154
7.3	组合逻辑设计	157
7.3.1	锁存器	157
7.3.2	组合逻辑反馈环路	161
7.3.3	脉冲产生电路	162
7.4	时序逻辑设计	162
7.4.1	门控时钟	162
7.4.2	异步计数器	162
7.4.3	次级时钟的产生	163
7.4.4	亚稳态	163
7.4.5	实例 7: T_{∞} 引起的亚稳态分析	163
7.5	代码风格	165
7.6	实例 8: 信号消抖时的亚稳态及解决方案	165
7.6.1	信号消抖基本介绍	165
7.6.2	基于 CPLD/FPGA 的信号消抖设计	168
7.7	本章小结	170
7.8	思考与练习	170

第 8 章 仿真与 Testbench 设计	171
8.1 仿真概述	171
8.1.1 周期驱动	171
8.1.2 事件驱动	172
8.1.3 混合语言仿真	172
8.2 仿真器的选择	173
8.3 Modelsim 简介与仿真	173
8.3.1 Modelsim 简介	173
8.3.2 功能仿真	174
8.3.3 时序仿真	180
8.4 Testbench 设计	182
8.4.1 时 钟	182
8.4.2 值序列	184
8.4.3 复 位	186
8.4.4 任 务	187
8.4.5 函 数	188
8.4.6 事 件	188
8.4.7 并行激励	189
8.4.8 系统任务和系统函数	189
8.5 Testbench 结构化	189
8.6 实例 9: 基于 Modelsim 的 I ² C Slave Testbench 设计	191
8.7 实例 10: 基于 Modelsim 的 LPC Slave 接口仿真设计	195
8.8 实例 11: 基于 Modelsim 的信号消抖程序仿真设计	201
8.9 本章小结	203
8.10 思考与练习	203
第 9 章 CPLD/FPGA 的验证方法学	204
9.1 验证与仿真	204
9.2 验证与测试	205
9.3 验证的期望	205
9.4 验证的语言	206
9.4.1 e 语言	207
9.4.2 SystemVerilog	207
9.4.3 SystemC	207
9.4.4 验证语言的分类	208
9.5 断 言	208
9.6 验证的分类	208
9.6.1 形式验证	209

目 录

9.6.2 功能验证	210
9.7 代码覆盖	211
9.8 验证工具	212
9.9 验证计划	212
9.10 DFT	213
9.11 版本控制	214
9.12 实例 12: 基于 FSM 的 SVA 断言验证设计	214
9.12.1 SVA 简介	214
9.12.2 基于 FSM 的 SVA 断言设计	215
9.13 本章小结	221
9.14 思考与练习	221
第 10 章 CPLD/FPGA 的高级应用	222
10.1 基于 DSP 的 FPGA 设计	222
10.1.1 DSP 的发展及解决方案	224
10.1.2 基于 DSP 的 FPGA 设计	225
10.1.3 实例 13: 基于 DDS 的正弦波信号发生器的设计	229
10.2 基于嵌入式处理器的 FPGA 设计	232
10.2.1 硬核、固核与软核	233
10.2.2 基于嵌入式处理器的 FPGA 设计流程	234
10.2.3 基于嵌入式处理器的 FPGA 设计应用	235
10.3 典型的 SOPC 运用: Nios II 简介及应用	237
10.3.1 Nios II 简介	237
10.3.2 实例 14: 基于 Nios II 软核处理器 PWM 控制器设计	242
10.4 基于 HardCopy 技术的 FPGA 设计	248
10.4.1 HardCopy 简介	248
10.4.2 基于 HardCopy 技术的 FPGA 设计流程	249
10.5 嵌入式逻辑分析仪	250
10.6 本章小结	252
10.7 思考与练习	252
第 11 章 CPLD/FPGA 系统设计	254
11.1 常用电平标准及其接口设计	254
11.1.1 常用电平标准	254
11.1.2 接口设计	256
11.1.3 接口设计的抗干扰措施	256
11.1.4 OC/OD 门	257
11.1.5 三态门	257
11.2 信号完整性概述	258

目 录

11.2.1	信号完整性的基本原则	259
11.2.2	传输线的基本理论	259
11.2.3	反射与阻抗匹配	261
11.2.4	串 扰	263
11.2.5	EMI	264
11.2.6	芯片封装	264
11.2.7	信号完整性的工具	265
11.3	高速设计与 SERDES	265
11.3.1	高速设计的基本原则和注意事项	265
11.3.2	SerDes	266
11.4	电源完整性概述	268
11.4.1	电源噪声	268
11.4.2	PCB PDS 设计技巧与挑战	268
11.4.3	电源完整性的基本原则和注意事项	278
11.4.4	实例 15: 采用 Altera PDN 工具的电源耦合电容设计	279
11.5	功耗与热设计	284
11.5.1	功耗设计	284
11.5.2	热设计	286
11.6	PCB 设计与 CPLD/FPGA 系统设计	288
11.7	实例 16: 基于 $\mu\text{C}/\text{OS}-\text{II}$ 的 FPGA 系统设计	292
11.7.1	$\mu\text{C}/\text{OS}-\text{II}$ 简介	292
11.7.2	系统设计要求简介	292
11.7.3	设计思路及步骤	292
11.8	本章小结	305
11.9	思考与练习	305
	参考文献	307

第 1 章

概 述

本章重点介绍数字电路设计的建模以及可编程逻辑器件的一些基本概念,简单介绍 CPLD/FPGA 的设计理念及其验证流程。

本章主要内容如下:

- 数字电路的发展演变;
- CPLD/FPGA 的介绍;
- 设计语言及其方法的介绍;
- 硬件语言与软件语言的区别;
- 设计与验证流程;
- CPLD/FPGA 的前景与展望。

1.1 数字电路基础及发展演变

1904 年英国科学家弗莱明为自己发明的电子管弗莱明“阀”申请了专利,人类历史上第一只电子管诞生了,世界从此迈入了电子时代。图 1-1 就是当时较为流行的一款电子管。1906 年德福雷斯特在此基础上发明了三极管,从此三极管成为了被广泛应用的电子器件。而今随着电子技术的不断发展,数字电路相继从晶体管、中小规模集成电路、超大规模集成电路(VLSIC)发展到今天的专用集成电路(ASIC),数字逻辑器件也由从前简单的逻辑门发展到了复杂的 SOC(System On Chip)。

随着科学技术及其工艺的发展,体积小、功耗低、集成度高成为了 IC 设计所追求的目标。系统的逻辑复杂度、规模与日俱增,日新月异。20 世纪 60 年代,Gordon Moore 预测一块芯片上集成的晶体管的数量将呈指数规律增长,即所谓的“摩尔定律”。他认为芯片上的晶体管的集成度每 18 个月将会翻一番,换一句话来说,就是在单位面积上逻辑门的数量每 18 个月翻一番,即呈指数级数增长,累计到现在已经超过 2^{30} 。到目前为止,这个定律依然正确。目前一个

Levin 超大规模的集成电路芯片,集成度一般达到了十几亿门。而这样高的集成度也要求人

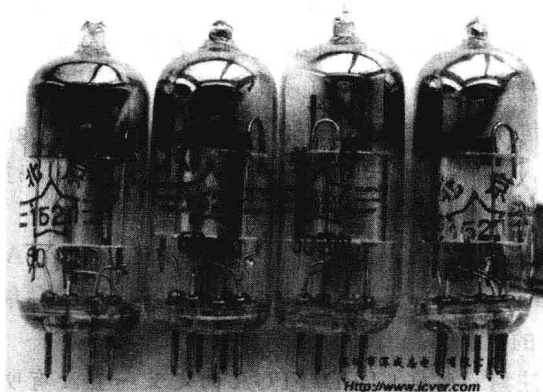


图 1-1 电子管

第1章 概述

们思考:该把芯片设计成固定功能,还是设计成可以灵活地选择功能呢?于是集成电路芯片便朝着两个方向并行发展,一个是专用集成芯片(Application Specific Integrated Circuit, ASIC),另一个是可编程逻辑器件。ASIC一般应用于固定场合。人们设计线路时,需要按照它的要求进行设计,这就要求电子工程师具有一定的技术和经验。另外,有些ASIC功能复杂且繁多,而实际上具体应用只需要其中的一部分,这样就造成了逻辑资源的浪费。而可编程逻辑器件功能强大,人们可以通过编程来实现自己想要的功能,去除不需要的冗余功能。只要了解编程软件和硬件的编程语言就可以实现对芯片的设计,这样可以加速芯片设计并缩短其上市时间(Time To Market, TTM)。

另一方面,这样复杂的集成度也就意味着传统的设计方法和设计理念已经不能适应芯片的产业发展,这就要求人们去变革和更新现有的设计理念以及设计方法。在早期简单的逻辑门设计阶段,电子辅助设计(EDA)的作用并不明显,人们往往习惯于使用卡诺图来简化设计,然后通过单板等试验进行系统设计验证。计算机的出现及发展带动和加速了软件产业的发展,从而使利用软件编程方法完成硬件芯片电路设计成为现实。这时,工程师开始使用EDA工具来对原理图进行仿真分析化简,分析其性能特征,可是因为原理图的可移植性差、维护起来费时费力的缺点,它只适用于一些比较简单的集成电路芯片,而当集成度复杂到ASIC和可编程逻辑器件时,一种抽象性更高、应用起来更简单的设计语言——硬件描述语言(Hardware Design Language, HDL)便应运而生。硬件描述语言的出现加速了芯片设计的速度,缩短了上市时间,而硬件描述语言本身也在日益丰富与壮大。

1.2 CPLD/FPGA 的介绍

CPLD/FPGA 是用户根据需要自行构造逻辑功能的数字集成电路。CPLD/FPGA 基本设计方法就是借助集成开发软件平台,用画原理图或者硬件描述语言编程的方法生成相应的网表文件,然后通过布局布线以及时序约束下载到目标芯片中,实现设计的数字系统。

CPLD/FPGA 的产生是集成电路的不断发展带动了计算机的高速发展,从而带动了软件技术更新、EDA 技术飞速发展的结果。历史上,第一个可编程逻辑阵列器件 PLA(Programmable Logic Array)出现在 20 世纪 70 年代中期,它的主要结构是由“与或”阵列组成的,与阵列和或阵列都可以同时编程。随着技术的发展,可编程逻辑器件已经从 PLA 发展到 PAL(Programmable Array Logic)再到 GAL(Generic Array Logic),目前已发展到 CPLD/FPGA。

传统的 CPLD(Complex Programmable Logic Device)发展来自于典型的 PAL、GAL 器件结构,任何一个组合逻辑可以用“与或”表达式来描述,能够实现大量的组合逻辑功能,适合于中小规模通用数字集成电路。

传统的 FPGA(Field Programmable Gate Array)基于查找表结构,绝大多数都是基于 SRAM 的结构,相对而言集成度比 CPLD 要高,能够快速实现许多复杂的逻辑功能,甚至用于 SOC 的设计。

注意:目前 CPLD/FPGA 的定义一直没有明确,不同厂家的叫法也不尽相同。Xilinx 公司把基于查找表技术、SRAM 技术和要外挂配置用的 E²PROM 的 PLD 叫作 FPGA;把基于乘积项技术、Flash 工艺的 PLD 叫作 CPLD。Altera 公司把自己的 PLD 产品——MAX 系列(乘积项技术、Flash 工艺)、FLEX 系列(查找表技术,SRAM 工艺)都叫作 CPLD。事实上,随着技

术的发展,最新的 CPLD 和 FPGA 结构已经越来越相似,特别是 Lattice 公司的 XO 系列和 Altera 公司的 MAXII 系列问世后,人们很难用一个明确的定义去区分 CPLD 和 FPGA。

经过几十年的发展,许多公司都开发出了 CPLD/FPGA 可编程逻辑器件,而经过行业的调整和整合,目前 Altera、Lattice 和 Xilinx 公司的 CPLD/FPGA 产品已经占到了市场份额的 80% 以上。

Altera 公司在 20 世纪 90 年代以后发展迅猛,是最大的可编程器件供应商之一。在 CPLD 产品方面连续推出了 MAX3000/7000、FLEX10、MAXII、MAXIIIZ 系列,而在 FPGA 产品方面也不断推陈出新,推出了 Cyclone、Stratix、Arria 等系列。

Xilinx 公司是 FPGA 的发明者,是老牌的 FPGA 公司,也是最大的可编程逻辑器件供应商之一,其产品丰富,种类齐全,主要有 XC9500、Coolrunner、CoolrunnerII 等系列的 CPLD 产品以及 Spartan、Virtex 系列的 FPGA 产品,目前最新款的 FPGA 是 Artix-7、Kinex-7、Virtex-7 系列的产品。

Lattice 公司是 ISP(In System Program,在系统编程)技术的发明者,而这一项技术极大地促进了 PLD 产品的发展。在 20 世纪 90 年代末、21 世纪初,相继收购了 Vantis(原 AMD 子公司)和 Agere 公司(原 Lucent 微电子部),从而成为了第三大可编程逻辑器件供应商。它可以提供从 PAL 到 FPGA 的一系列产品,甚至包括模拟器件,其中 CPLD 比较有特色,特别是 XO 系列。目前,该公司的主要的产品有 ispMACH4000 系列、XO 系列 CPLD、EC 系列、XP 系列 FPGA,以及可编程模拟器件等。

通常而言,在欧美市场使用 Xilinx 的人较多,而在亚太地区使用 Altera 的人相对较多。全球 60% 的 CPLD/FPGA 产品都是由 Altera/Xilinx 公司提供的。

除此之外,目前世界上还有几家比较有影响的 CPLD/FPGA 公司,如 Actel、Qlogic 等,另外 2008 年新成立了一家 CPLD/FPGA 公司——SiliconBlue 公司。不过短时间内,它们还无法与上述三家公司匹敌。

1.3 设计语言及其方法的介绍

随着计算机技术和 EDA 技术的不断发展,各家可编程逻辑器件公司开发了各种硬件描述语言,如 AHDL(Analog Hardware Description Language,模拟硬件描述语言,Altera 公司为自家产品而开发的)、ABEL(Advanced Boolean Equation Language 即高级布尔方程语言,Lattice 公司专为其产品而开发的)。

1981 年美国国防部提出了一种新的标准化语言——VHDL(Very high speed integrated circuit Hardware Description Language,超高速集成电路硬件描述语言)。它的语法丰富,语句类型繁多,功能强大,结构严谨,并且能够在高层次上进行仿真描述,成为了最受欢迎的硬件描述语言之一,于 1987 年成为了 IEEE 标准。

1983 年,美国 GDA(Gateway Design Automatic)公司的 Phi Moordy 创立了 Verilog HDL 语言,并在 1984~1985 年期间设计出了仿真器 VerilogXL,之后又于 1986 年提出了用于快速门级仿真的 XL 算法。1989 年 GDA 公司被 Cadence 公司收购,1990 年 Cadence 公司决定开发 Verilog HDL 语言,并成立了 OVI(Open Verilog International)组织来促进 Verilog HDL 语言的发展。它吸收了 VHDL 和 C 语言的长处,有着类似于 C 语言的语法体系,设计

第1章 概述

人员只要有 C 语言基础就可以入门。1995 年,IEEE 制定了 Verilog HDL 的 IEEE 标准即 Verilog HDL1364 - 1995,之后又于 2001 年发布了 Verilog1364 - 2001 标准。

Verilog HDL 和 VHDL 作为最流行的 HDL,两者之间孰优孰劣一直是设计工程师不断争论的话题。二者在设计上都能胜任数字电路设计的任务,目前几乎所有的 CPLD/FPGA 设计综合和仿真软件平台都支持这两种语言。

硬件描述语言一直在这两种语言的基础上发展和演化着,之后又出现了许多更为抽象的硬件描述语言,如 SystemVerilog、SystemC、Superlog 和 CoWare C 等语言。这些高级 HDL 的语法结构更加丰富,更适合系统级、功能级等高层级的设计描述与仿真。图 1-2 所示为 HDL 适用层次示意图。

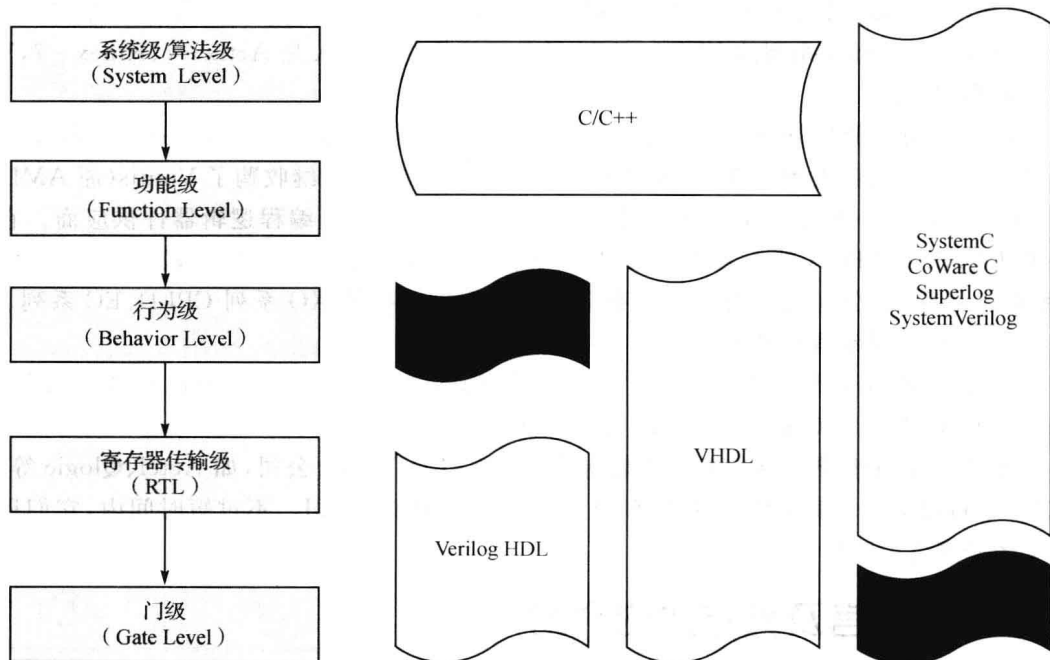


图 1-2 设计语言与设计层次的关系

1.4 硬件语言与软件语言的区别

硬件语言与软件语言有着本质的区别。Verilog 和 VHDL 语言作为硬件语言,其本质是在描述硬件。虽然从某种意义上来说,Verilog HDL 借鉴了许多 C 语言的要素,可是它描述出来的结果就是芯片内部实际的电路,所以一段 HDL 代码的优劣就在于它转化成所描述电路性能是否合理和流畅,而非代码是否简洁。

硬件语言与软件语言相比较,最大的区别在于软件语言缺乏了以下 3 个基本概念。

① 互连(Connectivity): 互连是硬件系统中的一个基本概念,而软件语言并没有这样的描述。Verilog HDL 中的 wire 变量可以很好地表达这样的功能。

② 并发(Concurrency): 软件语言天生就是串行的,只有执行了上一行程序才能进入下一行程序,这样与硬件系统的设计理念相违背的,硬件语言基于并行,能够有效地描述硬件系统。

③ 时序(Timing): 软件语言的运行没有一个严格的时间时序概念, 程序运行的快慢取决于处理器本身的性能, 而硬件语言可以通过时间度量与周期的关系来描述信号之间的关系。

但是硬件描述语言在抽象程度上比起软件语言相对差一些, 语法也不如软件语言灵活, 特别是文件的输入/输出。为了克服这些缺陷, PLI(Programmable Language Interface, 编程语言接口)也就应运而生, 这样就可以在仿真器里面实现 C 语言程序和 Verilog HDL 程序的相互通信, 或者是 Verilog HDL 调用 C 语言的函数库, 提高了 Verilog HDL 的灵活性和抽象能力。

1.5 设计与验证流程

当硬件系统进入软件编程设计阶段时, 基本上之前所有传统的设计理念和手段都需要更新和变革。传统上采用的“自下向上(Bottom Up)”的设计方法已经不再适应 CPLD/FPGA 的设计需求。CPLD/FPGA 的设计需要从系统整体要求出发, 自上向下逐步将设计内容细化, 最后完成系统硬件的整体设计, 这就是所谓的“自顶向下(Top Down)”的设计方法, 也就是从抽象到具体的一种设计方法。与传统的设计方法相比较, 自顶向下的方法不仅节约了大量的设计时间, 也大大减少了调试时间, 整体上缩短了系统的设计时间, 节省了大量的人力与物力, 加速了产品上市。图 1-3 所示为 HDL 硬件设计与验证的流程, 有些步骤可以根据系统的复杂程序进行适当的修剪、省略。

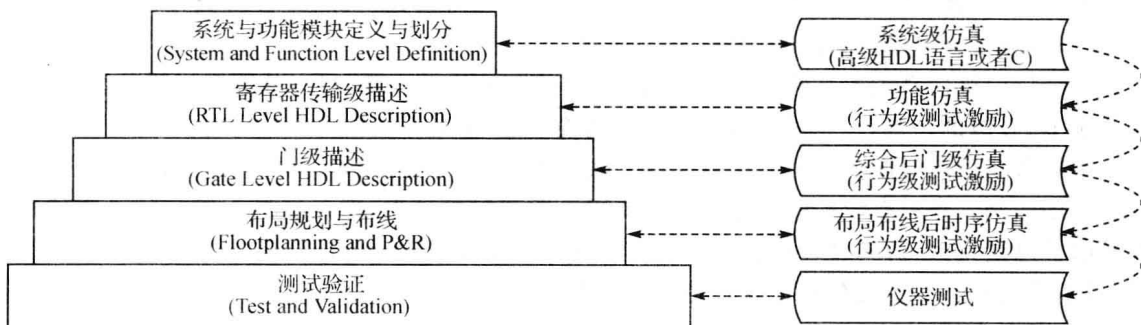


图 1-3 设计层次及任务要求

1. 系统与功能模块定义

在大型系统的设计与视线中, 首先需要对整个系统进行详细的系统规划与描述, 并且通过系统级的仿真来整体权衡和考量系统的性能和功能实现。这时侧重于系统的规划与实现, 而此时使用的仿真语言也多用高级描述语言, 如 C/C++、SystemVerilog、SystemC 等。

当确定了系统的整体功能后, 系统工程师就要对整个系统进行功能划分, 定义接口及其主要模块(如时钟、协议等), 并且通过仿真比较不同的方案所达到的性能优劣, 根据市场和系统性能指标要求从整体上优化实现方案, 以满足设计要求。

2. 寄存器传输级描述

当系统功能模块和接口定义好以后, 逻辑工程师就开始规划描述寄存器到寄存器之间的逻辑功能实现。寄存器传输级描述不同于门级描述, 它比门级更加抽象, 它不关心寄存器与组合逻辑之间的细节关系, 实现的语言也比门级简单。