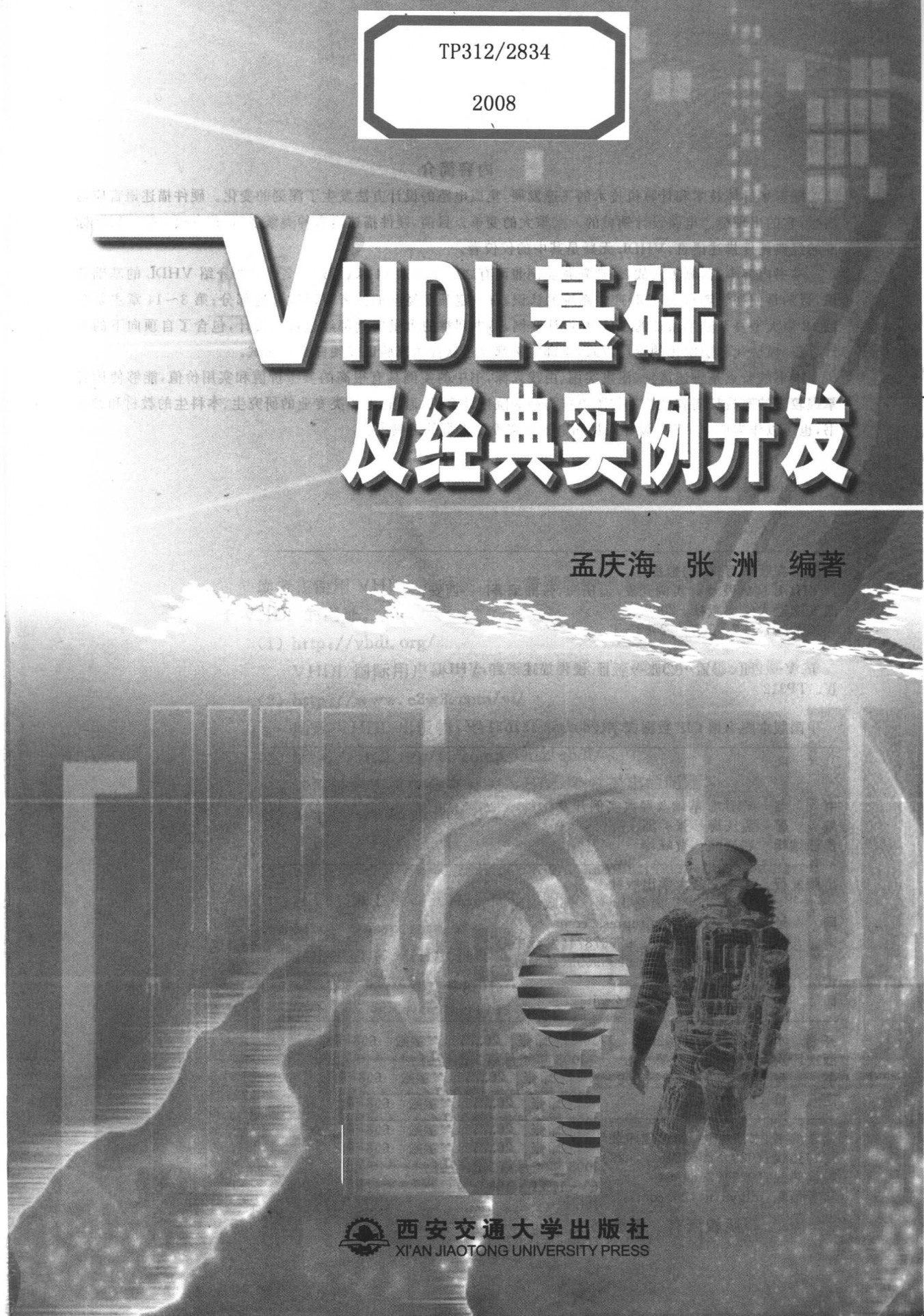
VHDL基础及经典实例开发

孟庆海　张洲　编著

西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

VHDL基础及经典实例开发

孟庆海　张洲　编著



西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

内容简介

随着半导体技术和计算机技术的飞速发展，集成电路的设计方法发生了深刻的变化。硬件描述语言应运而生，它的出现标志电路设计领域的一次重大的变革。目前，硬件描述语言种类繁多，百花齐放。作为国际标准的经典硬件描述语言，VHDL 无疑是其中的佼佼者。

本书内容丰富，侧重实战，经典新颖实例兼而有之。全书共 14 章。第 1～2 章主要介绍 VHDL 的基础知识，目的是使初学者对 VHDL 产生系统的认识，有一定开发经验的读者可以跳过这部分；第 3～14 章主要介绍 12 个大型复杂数字系统的 VHDL 设计实例，书中列举的大量实例都经过精心设计，包含了自顶向下的设计思想，模块化和层次化的设计方式，全部实例都经过软件仿真验证或硬件实际测试。

本书的特点是讲述清楚、注重实用、由浅入深，书中的实例具有很高的参考价值和实用价值，能够使读者掌握较多的实战技能和经验。它既可作为高等院校计算机、通信、电子类专业的研究生、本科生的教材和参考书，也可以作为广大 ASIC 设计人员和电子电路设计人员的工具书。

图书在版编目(CIP)数据

VHDL 基础及经典实例开发/孟庆海，张洲编著. —西安：西安交通大学出版社，2008.4

ISBN 978-7-5605-2563-1

Ⅰ.V… Ⅱ.①孟…②张… Ⅲ.硬件描述语言，VHDL Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2007)第 151017 号

书　　名　VHDL 基础及经典实例开发
编　　著　孟庆海　张　洲
责任编辑　屈晓燕　贺峰涛

出版发行　西安交通大学出版社
（西安市兴庆南路 10 号　邮政编码 710049）
网　　址　http://www.xjtupress.com
电　　话　(029)82668357　82667874(发行中心)
(029)82668315　82669096(总编办)
传　　真　(029)82668280
印　　刷　陕西丰源印务有限公司

开　　本　787mm×1092mm　1/16　**印张**　26.75　**字数**　658 千字
版次印次　2008 年 4 月第 1 版　2008 年 4 月第 1 次印刷
书　　号　ISBN 978-7-5605-2563-1/TP·499
定　　价　42.00 元

读者购书、书店添货、如发现印装质量问题，请与本社发行中心联系、调换。
订购热线：(029)82665248　(029)82665249
投稿热线：(029)82664954
读者信箱：jdlgy31@126.com

前　言

人类社会已进入到高度发达的信息社会，信息社会的发展离不开电子产品的进步。现代电子产品在性能提高、复杂度增大的同时，价格却一直呈下降趋势，而且产品更新换代的步伐也越来越快，实现这种进步的主要原因就是生产制造技术和电子设计技术的发展。前者以微细加工技术为代表，目前已进展到深亚微米阶段，可以在几平方厘米的芯片上集成数千万个晶体管；后者的核心就是 EDA 技术。EDA 是指以计算机为工作平台，融合了应用电子技术、计算机技术、智能化技术最新成果而研制成的电子 CAD 通用软件包。没有 EDA 技术的支持，想要完成上述超大规模集成电路的设计是不可想象的。反过来，生产制造技术的不断进步又必将对 EDA 技术提出新的要求。

因此，EDA 技术既是电路设计领域的发展潮流，又是超大规模集成电路设计的攻关方向。与其他核心技术类似，EDA 技术不能没有灵魂，于是，硬件描述语言应运而生，它的出现标志电路设计领域的一次重大的变革。

在上百种硬件描述语言中，VHDL 最早成为国际标准硬件描述语言。VHDL 语法规范，可读性强，易于修改和发现错误。VHDL 是一种全方位的硬件描述语言，包括系统行为级、寄存器传输级和逻辑门级多个设计层次，支持结构、数据流、行为三种描述形式及其混合描述，因此 VHDL 几乎覆盖了以往各种硬件描述语言的功能，整个自顶向下或自底向上的电路设计过程都可以用 VHDL 来完成。另外，VHDL 还具有以下优点：VHDL 的宽范围描述能力使它成为高层次设计的核心，将设计人员的工作重心提高到了系统功能的实现与调试，只需花较少的精力用于物理实现。VHDL 可以用简洁明确的代码描述来进行复杂控制逻辑的设计，灵活且方便，而且也便于设计结果的交流、保存和重用。VHDL 的设计不依赖于特定的器件，方便了工艺的转换。VHDL 是一个国际标准硬件描述语言，为众多的 EDA 厂商所支持，因此移植性好。

随着 VHDL 在我国的广泛应用，无论是 ASIC 设计人员、电子电路设计人员还是高等院校的学生都迫切需要一本除了介绍基本概念和基本语法知识外，更注重实际，实例丰富新颖的参考书。作者编写此书的目的就是让读者了解如何应用 VHDL 完成实际设计工作，进而全面有效地掌握它。

本书分为 14 章。第 1～2 章主要介绍 VHDL 的基础知识，目的是使初学者对 VHDL 形成系统的认识，有一定开发经验的读者可以跳过这部分；第 3～14 章主要介绍 12 个大型复杂数字系统的 VHDL 设计实例，书中列举的大量实例都经过精心设计，包含了自顶向下的设计思想，模块化和层次化的设计方式，全部实例都经过软件仿真验证或硬件实际测试。

作者在书中列举的大量实例都从读者的实际需要出发，注重实效，拒绝过时、老套，但不放弃经典；摒弃冷僻、乏味，但力求创新。经典的实例，诸如“数字钟”，是无数 ASIC 设计师的成长佳品；新颖的实例，诸如“虚拟逻辑分析仪”，可谓踏破铁鞋无觅处；更有实用价值的实例，诸如“异步 FIFO”、“SCI 串行通信接口”、“I^2C 接口”、“通用串并乘法器”……另外，提供了本书

中的12个大型实例的源代码，读者在需要时可以参考，方便移植，网址：http://ligongxjtupress.com。

本书的特点是讲述清楚、注重实用、由浅入深，书中的实例具有很高的参考价值和实用价值，能够使读者掌握较多的实战技能和经验。它既可作为高等院校计算机、通信、电子类专业的研究生、本科生的教材和参考书，也可以作为广大ASIC设计人员和电子电路设计人员的工具书。

本书主要由孟庆海编写，书中包含着作者多年来使用VHDL设计硬件电路过程中的经验总结。在本书的编写过程中，张帅、徐皓、张鹏、孟宪武、张海涛、高淑芹、郭健收集了大量的资料，张洲、程显奎、张文龙、李晓凯、范博、朱磊对其中的VHDL代码进行了调试工作，尤晓丽、曹霖、周美荣、陈智强完成了全书的文字校对工作；另外作者还得到了许多专家、学者和同行的宝贵建议，在此一并表示诚挚的感谢！

作者的家人在本书编写过程中，一直给予作者莫大的支持和鼓励，在此，向他们表示深深的感谢！

鉴于作者的水平有限，书中难免存在错误和不足之处，望读者不吝批评指正。

作　者

目　录

第1章 VHDL概述

本章首先介绍硬件描述语言的基本知识，包括硬件描述语言的产生以及硬件描述语言的种类。而后，重点介绍作为经典硬件描述语言和全书描述对象的 VHDL 的特点及设计流程，使读者对 VHDL 产生系统的认识。

1.1 硬件描述语言

如前言所述，硬件描述语言是集成电路设计的发展方向。本小节将重点介绍硬件描述语言的产生及主流硬件描述语言。

1.1.1 硬件描述语言的产生

随着半导体技术和计算机技术的飞速发展，集成电路的设计方法发生了深刻的变化。从计算机辅助设计(Computer Aided Design，CAD)、计算机辅助工程(Computer Aided Engineering，CAE)到电子系统设计自动化(Electronic System Design Automation，ESDA)，设计的自动化程度越来越高，系统也越来越庞大，越来越复杂。硬件描述语言(Hardware Description Language，HDL)应运而生，它的出现标志着电路设计领域的一次重大的变革。

早期，传统的硬件电路设计方法主要是采用电路原理图方法，这种方法所完成的设计文件是几十张、几百张甚至几千张电路原理图，每张图中标明了构成硬件电路的各种电子器件的名称和连接线关系。对于大规模复杂电路设计，这种方法的缺点是显而易见的，譬如无法在设计前期仿真，造成后期的被动以至于项目失败，同时数以千计的电路原理图给设计的归档、阅读、修改和使用带来极大不便。随着大规模集成电路的发展，这种传统的设计方法已经无法满足要求，渐渐地被硬件描述语言所替代。

硬件描述语言 HDL 是一种用形式化方法描述硬件电路系统的语言。利用这种语言，硬件电路系统的设计可以从上层到下层(从抽象到具体)逐层描述设计思想，用一系列分层次的

模块来表示复杂的系统。然后，利用电子设计自动化(EDA)工具，逐层进行仿真验证，再将其变为实际电路的模块组合，经过自动综合工具转换到门级电路网表。最后，用专用集成电路ASIC或现场可编程门阵列FPGA自动布局布线工具，把网表转换为要实现的具体电路布线结构。

采用硬件描述语言设计硬件电路可以增加设计的自由度和灵活度，节省人力和物力，缩短开发周期，与传统的原理图设计方法相比，有如下优势。

(1) 采用自顶向下的设计方法。与传统的自底向上的设计方法不同，采用硬件描述语言设计电路，从系统总体要求出发，先确定顶层模块，进行顶层模块的设计，再按照不同的功能，将顶层模块划分为若干子模块，子模块还可以被继续划分为更简单和易于实现的模块，然后进行具体设计，最后完成整个系统的设计。

(2) 早期仿真。系统的总体仿真是设计的重要环节，这时的设计与工艺无关。由于设计的仿真和调试是在高层次完成的，所以能够在早期发现结构设计上的错误，提高设计的成功率。

(3) 降低设计难度。硬件描述语言具有多层次描述系统功能的能力，从系统的数学模型到门级电路。将高层次行为描述和低层次的寄存器传输描述以及结构化描述结合起来，使对硬件电路的描述更加准确。利用模块化设计，可以实现资源共享，极大地减少了重复劳动。上述这些特点，大大提高了设计人员的工作效率，降低了硬件系统的设计难度。

(4) 提高设计文件可读性。前面提到，采用传统的电路设计方法，设计文件是几十张、几百张甚至几千张电路原理图；而采用硬件描述语言时，设计文件是采用硬件描述语言编写的程序，给阅读、归档、修改和使用带来极大的方便。

(5) 大量采用ASIC。ASIC芯片与硬件描述语言的关系十分密切，二者相辅相成，相互促进。众多的ASIC生产厂商的工具软件都支持硬件描述语言。这样，设计人员在设计硬件电路时，就不会受到专用芯片的限制，而是根据硬件电路系统设计的需要来选择ASIC芯片，方便修改设计，增加灵活度，缩短开发周期。

1.1.2 硬件描述语言的种类

硬件描述语言经过几十年的发展，种类繁多。20世纪80年代初，硬件描述语言已达上百种，它们对电子设计自动化起到了促进和推动作用。但是，这些语言一般面向各自的领域，并没有得到普遍认同而成为标准硬件描述语言。20世纪80年代后期，硬件描述语言朝着标准化、集成化的方向发展。

目前，比较有代表性的硬件描述语言有VHDL、Verilog HDL、Superlog和System C等。前二者已经成为IEEE标准，在我国十分流行；后二者则是后起之秀，具有良好的发展前景，国内对此研究较少，值得关注。

1. VHDL

有关VHDL的内容将在下一节详细介绍。

2. Verilog HDL

Verilog HDL语言最早是由GDA(Gateway Design Automation)公司的设计师Phil Moorby在1983年开发出来的。在1984～1985年间，Phil Moorby成功设计了Verilog-XL仿

真器;1986 年,他对 HDL 的发展做出了另一个重大贡献,提出了快速门级仿真的 XL 算法,这使得 Verilog HDL 语言变得更加丰富和完善。1989 年,CADENCE 公司收购了 GDA 公司,Verilog HDL 语言从此成为 CADENCE 公司 EDA 开发环境中的硬件描述语言。1990 年,CADENCE 公司公开发表了 Verilog HDL 语言,并且成立 LVI 组织以促使 Verilog HDL 语言成为 IEEE 标准,即 IEEE Standard 1364 - 1995。2001 年发布了 Verilog HDL 1364 - 2001 标准。在这个标准中,加入了 Verilog HDL—A 标准,使 Verilog HDL 有了模拟设计描述的能力。

3. Superlog 语言

开发一种新的硬件设计语言,总是有些冒险,而且未必能够利用原先对硬件开发的经验。能不能在原有硬件描述语言的基础上,结合高级语言 C、C++甚至 Java 等语言的特点进行扩展,达到一种新的系统级设计语言标准呢? Superlog 就是在这样的背景下研制开发的系统级硬件描述语言。Verilog HDL 语言的首创者 Phil Moorby 和 Peter Flake 等硬件描述语言专家,在一家叫 Co-Design Automation 的 EDA 公司进行合作,开始对 Verilog HDL 进行扩展研究。1999 年,Co-Design 公司发布了 SUPERLOGTM 系统设计语言,同时发布了两个开发工具:SYSTEMSIMTM 和 SYSTEMEXTM,一个用于系统级开发,一个用于高级验证。2001 年,Co-Design 公司向电子产业标准化组织 Accellera 提交了 Superlog 扩展综合子集 ESS,这样它就可以在今天 Verilog HDL 语言的 RTL 级综合子集的基础上,提供更多级别的硬件综合,为各种系统级的 EDA 软件工具所利用。

至今为止,已超过 15 家芯片设计公司用 Superlog 来进行芯片设计和硬件开发。Superlog 是一种具有良好前景的系统级硬件描述语言。但是不久前,由于整个 IT 产业的滑坡,EDA 公司进行大的整合,Co-Design 公司被 Synopsys 公司兼并,形势又变得扑朔迷离。

4. System C 语言

随着半导体技术的迅猛发展,SoC 已经成为当今集成电路设计的发展方向。在系统芯片的各个设计中,像系统定义、软硬件划分、设计实现等,集成电路设计界一直在考虑如何满足 SoC 的设计要求,一直在寻找一种能同时实现较高层次的软件和硬件描述的系统级设计语言。System C 正是在这种情况下,由 Synopsys 公司和 CoWare 公司积极响应目前各方对系统级设计语言的需求而合作开发的。1999 年 9 月 27 日,40 多家世界著名的 EDA 公司、IP 公司、半导体公司和嵌入式软件公司宣布成立"开放式 System C 联盟"。Cadence 公司也于 2001 年加入了 System C 联盟,使 System C 跨向业界标准的进程大大加快。System C 从 1999 年 9 月联盟建立初期的 0.9 版本开始更新,从 1.0 版到 1.1 版,2001 年 10 月推出了 2.0 版。

1.2　VHDL 硬件描述语言

20 世纪 70 年代末和 80 年代初,美国国防部提出了 VHSIC(Very High Speed Integrated Circuit)计划,VHSIC 计划的目标是为下一代集成电路的生产、实践阶段性的工艺极限以及完成 10 万门级以上的电路设计,建立一种新的描述方法。

许多高科技半导体公司为美国国防部设计芯片,这些公司几乎都拥有自己的硬件描述语言。这些硬件描述语言一般仅支持门级描述,而不支持大规模电路描述,各家公司的设计人员

难以相互协作。在这种背景下，来自 IBM、Texas Instruments 和 Intermetrics 公司的专家在美国国防部的资助和组织下进行合作，开发了一种强大的硬件描述语言。1985 年，公布了第一个对外版本，VHDL7.2。1986 年，IEEE 开始致力于 VHDL 的标准化工作，并成立了一个 VHDL 标准化小组。经过反复的修改与扩充，直到 1987 年 12 月，VHDL 才被接纳为 IEEE 1076 标准，称为 VHDL-87 标准。1988 年，Milstd454 规定所有为美国国防部设计的 ASIC 产品必须采用 VHDL 来描述。1993 年，VHDL—87 标准被修订，更新为 VHDL—93 标准。

1995 年，我国国家技术监督局制定的《CAD 通用技术规范》推荐 VHDL 作为我国电子设计自动化硬件描述语言国家标准。至今，VHDL 在我国迅速普及，成为广大硬件工程师必须掌握的一项技术。

1.2.1 VHDL 的特点

VHDL 能够成为标准化的硬件描述语言并获得广泛应用，正是因为它具有如下的优点。

(1) 功能强大和设计灵活。VHDL 拥有强大的语言结构，可以用简洁的程序描述复杂的逻辑控制。为了有效地控制设计的实现，它具有多层次的设计描述功能，支持设计库和可重复使用的元件生成；支持层次化设计和模块化设计，同时，VHDL 还支持同步、异步和随机电路设计，这是其他硬件描述语言难以比拟的。

(2) 与具体器件无关。设计人员采用 VHDL 设计硬件电路时，并不需要首先确定设计采用哪种器件，也不需要特别熟悉器件的内部结构。这样做的好处是设计人员可以集中精力进行系统设计。当设计完成后，再根据消耗的资源，量体裁衣，选择合适的器件。

(3) 很强的移植能力。VHDL 的移植能力非常强，因为它是一种标准的硬件描述语言。同一个设计的程序可以被不同的工具所支持，包括综合工具、仿真工具、系统平台等。

(4) 强大的硬件描述能力。VHDL 既能够描述系统级电路，又可以描述门级电路。描述方式既可以采用行为描述、寄存器传输描述或者结构描述，也可以用其混合描述方式。同时，VHDL 也支持惯性延迟和传输延迟，以便准确地建立硬件电路模型。

VHDL 的描述能力还体现在具有丰富的数据类型，既支持标准定义的数据类型，又支持用户自己定义的数据类型，增加了自由度。

(5) 语法规范、易于共享。VHDL 的语法非常规范，可读性极强。用 VHDL 编写的代码文件既是程序，也是文档；既可以作为设计人员之间交流的内容，又可以作为签约双方的合同文本。另一方面，作为一种工业标准，VHDL 易于共享，适合大规模协作开发。同时，这些特点也促进了 VHDL 的发展和完善。

综上所述，VHDL 的确具有其他硬件描述语言所不具有的优点。但是，VHDL 也并不是一种完美的硬件描述语言，自身存在一些缺点，主要体现在 3 个方面。

(1) 对设计者的硬件电路功底要求较高。采用 VHDL 描述硬件电路需要设计人员具有较多的硬件电路知识，甚至芯片结构方面的知识。应该摆脱一般的高级语言程序设计思路，因为电路世界里的事件往往是并行发生的。硬件电路系统内部的模块可以是互相独立的，也可以是互为因果的。先构思电路，然后才能描述。

(2) 系统抽象描述能力较差。虽然 VHDL 能够描述系统级电路，但系统的抽象性不能太强，否则 EDA 工具无法综合。目前，VHDL 中的一部分运算和数据类型只能适用于系统建模，如果要综合成实际的硬件电路，存在很大困难。

(3) 不具备描述模拟电路能力。对于模拟电路而言，VHDL 并不是一种理想的硬件描述语言。现在，IEEE 正致力于设计 VHDL 的超级 VHDL-AMS，这种语言将能够对模拟电路和数模混合电路进行描述，可以预见，支持模拟电路和数模混合电路描述将是硬件描述语言的发展方向。

1.2.2　VHDL 设计流程

采用 VHDL 设计硬件电路系统的设计流程一般可以分为以下几个步骤。

(1) 确定电路具体功能。通常情况下，开发前期先设计总体方案，但总体方案相对比较抽象，使用 VHDL 的设计人员必须分析电路所要实现的具体功能。

(2) 设计输入。利用自顶向下的方法，将设计划分为不同的功能模块。每个模块完成一定的逻辑功能。模块划分是设计过程中的一个重要环节，这一步要花费较多的时间和精力完成，从而保证整体最优。

编写每个模块的程序，然后将各个模块的程序组合在一起，完成整个系统的 VHDL 描述。

(3) 功能仿真。在功能仿真阶段主要对所设计的电路进行功能验证，通过功能仿真，发现设计存在的缺陷。例如，输入输出是否有矛盾，有无未加处理的输入信号，是否允许使能等。通过功能仿真，在设计前期纠正缺陷和错误，可以节省后期的时间，缩短整体开发周期。

(4) 综合、优化和布局布线。综合的作用是将较高层次的 VHDL 抽象描述转化为较低级别抽象，或者说实际的硬件电路。

优化的作用是将电路设计的时延缩到最小和有效利用资源。几乎所有的高级 VHDL 综合工具都可以利用约束条件对电路设计进行优化。一般情况下，常用的约束条件主要包括时间约束和面积约束。

布局布线的作用是将通过综合和优化所得到的逻辑，安放到一个逻辑器件之中的过程。一个较好的布局布线过程就是将电路的相关部分放置在一起，以消除布线延迟。

(5) 时序仿真。时序仿真是接近于真实器件运行特性的仿真，仿真文件中包含了器件硬件特性参数，因而仿真精度高。时序仿真的文件必须来自针对具体器件的综合器与适配器。

(6) 编程下载。编程下载指将 VHDL 程序经过综合、优化和布局布线后生成的编程数据写入具体的可编程器件中。

(7) 硬件测试。最后，将写入编程数据的硬件系统进行实际测试，以便检验设计的运行情况。

第2章

VHDL硬件描述语言

和其他高级语言类似，VHDL也是由各种基本元素和基本结构组成，如常量、变量、运算符、语句、进程以及函数，等等。读者在学习硬件描述语言时，首先要掌握这些基本元素的用法，熟悉基本结构，同时总结硬件描述语言与其他高级程序语言的不同之处，为以后设计复杂数字系统打下坚实基础。

2.1 VHDL的基本元素

标识符、数据对象、数据类型及运算符和操作符是组成VHDL的积木，每一个VHDL程序都离不开这些基本元素，掌握它们对正确书写VHDL程序至关重要。

2.1.1 标识符

VHDL—87标准中有关标识符的语法规范被VHDL—93标准全部接受并加以扩展。通常，称VHDL—87标准中的标识符为短标识符，而将VHDL--93标准中的标识符称为扩展标识符。下面对这种标识符的命名规则分别进行介绍。

1. 短标识符

在VHDL中，VHDL—87标准中的短标识符应该遵循以下命名规则：

(1)短标识符必须由英文字母、数字以及下划线组成；

(2)短标识符必须以英文字母开头；

(3)短标识符不区分大小写；

(4)短标识符最后一个字符不能是下划线；

(5)短标识符中不能含有两个连续的下划线；

(6)VHDL中的保留字不能作为短标识符来使用。

下面是几个符合规范的短标识符。

Freq
Clk_8module
ROM_DP
下面是几个不符合规范的标识符。

_Freq	标识符必须以英文字母开头
8Clk_module	标识符必须以英文字母开头
ROM__DP	标识符中不能含有两个连续的下划线
ROMDP_	标识符的最后一个字符不能是下划线
ENTITY	标识符不能是保留字

2. 扩展标识符

在 VHDL 中，VHDL—93 标准中的扩展标识符应该遵循以下命名规则：

(1)扩展标识符用反斜杠来界定，例如：\data_bus\；

(2)扩展标识符中可以包含图形符号和空格等，例如：\data&_bus\；

(3)扩展标识符的两个反斜杠之间可以使用保留字，例如：\ENTITY\；

(4)扩展标识符的两个反斜杠之间可以用数字开头，例如：\8_data_bus\；

(5)扩展标识符中允许多个下划线相连，例如：\data_ _ _bus\；

(6)同名的扩展标识符和短标识符是不同的，例如：\data_bus\和 data_bus 不同；

(7)扩展标识符中如果含有一个反斜杠，这时则应该用两个相邻的反斜杠来代替，例如：如果扩展标识符的名称为 data\bus，那么此时的扩展标识符应该表示为\data\\bus\；

(8)与短标识符不同，扩展标识符是区分大小写的，例如：\a\和\A\是不同的标识符。

下面是 VHDL 中的关键字，即保留字，不能作为一般标识符使用。

ABS	ACCESS	AFTER	ALIAS
ALL	AND	ARCHITECTURE	ARRAY
ASSERT	ATTRIBUTE	BEGIN	BLOCK
BODY	BUFFER	BUS	CASE
COMPONENT	CONFIGURATION	CONSTANT	DISCONNECT
DOWNTO	ELSE	ELSIF	END
ENTITY	EXIT	FILE	FOR
FUNCTION	GENERATE	GENERIC	GROUP
GUARDED	IF	IMPURE	IN
INERTIAL	INOUT	IS	LABEL
LIBRARY	LINKAGE	LITERAL	LOOP
MAP	MOD	NAND	NEW
NEXT	NOR	NOT	NULL
OF	ON	OPEN	OR
OTHERS	OUT	PACKAGE	PORT
POSTPONED	PROCEDURE	PROCESS	PURE
RANGE	RECORD	REGISTER	REJECT
REM	REPORT	RETURN	ROL

ROR	SELECT	SEVERITY	SIGNAL
SHARED	SLA	SLL	SRA
SRL	SUBTYPE	THEN	TO
TRANSPORT	TYPE	UNAFFECTED	UNITS
UNTIL	USE	VARIABLE	WAIT
WHEN	WHILE	WITH	XNOR
XOR			

2.1.2 数据对象

在 VHDL 中，通常把用来保存数据的单元称为对象，有的书中称之为客体。VHDL 的数据对象包括 4 类：常量(CONSTANT)、变量(VARIABLE)、信号(SIGNAL)及文件(FILE)。其中，前 3 个是 VHDL—87 标准通过的，文件(FILE)是 VHDL—93 标准中通过的。

1. 常量(CONSTANT)

常量就是指在设计实体中不会发生变化的值，它可以在很多部分进行说明，并且可以具有任何的数据类型。作为一种硬件描述语言的元素，常量在硬件电路设计中具有一定的物理意义，它通常用来代表硬件电路中的电源或者地线等。

一般来说，采用常量的最大好处是可以使设计人员编写出更好的 VHDL 程序，并且可以使程序的修改变得更加容易。这一点非常类似于其他高级编程语言中的常量。通常，常量说明的格式如下所示：

CONSTANT 常量名[,常量名…]:数据类型 :=表达式;

其中，“CONSTANT”是用来表示常量的保留字，用来声明一个常量；“[]”中的部分表示是可选项，即多个相同数据类型的常量可以在此一起声明；数据类型是对象所具有的类型说明；表达式用来对常量进行赋值，赋值符号为“ :=”。

这里，所谓保留字就是指一些在 VHDL 中具有专门定义的特殊字符，(参见上一小节)。对于保留字来说，一般可以根据自己的设计习惯来确定保留字的书写格式，本书中 VHDL 的保留字均采用大写形式。同时有一点要说明：除前面介绍的长标识符外，一般 EDA 工具对于 VHDL 的大小写是不加区分的。

下面给出几个常量说明的例子，读者会发现常量说明是十分简单的。

```
CONSTANT pi    : real := 3.14;
CONSTANT VCC   : real := 3.3;
CONSTANT GND   : real := 0.0;
CONSTANT delay : time := 10 ns;
```

这里需要**注意**：常量被赋值以后的值将不再改变。例如，上面用来进行运算的常量 pi 被定义为 3.14，那么在 VHDL 程序中 pi 的值就被固定为 3.14。任何试图修改常量 pi 值的操作都将被视为非法操作。

常量说明的范围十分广泛，它既可以在程序包、实体说明和结构体的说明部分进行说明，也可以在语句的说明部分进行说明。在 VHDL 程序中不同部分说明的常量具有不同的作用范围：程序包中说明的常量可以在其所包含的任何实体和结构体中使用；实体说明中说明的常

量可以在其对应的结构体中使用；结构体中说明的常量只能在本结构体中使用；进程语句中说明的常量只能在该进程语句中使用。

对于常量说明有一点要注意，常量所赋的值应该与定义的数据类型一致，否则将出现错误，这一点应该避免。例如：

```
CONSTANT  delay : time := 10.0;
```

常量 delay 定义的数据类型是时间类型 time，而所赋的值却是一个实数 10.0，显然这是一个错误的常量说明。

2. 变量(VARIABLE)

变量主要用于对暂时数据进行局部存储，它是一个局部量，只能在进程语句、过程语句和函数语句的说明部分中加以说明。作为一种硬件描述语言的元素，变量在硬件电路设计中具有一定的物理意义，变量主要用于局部数据的暂时存储，是一种载体。

通常，变量说明的格式如下所示：

```
VARIABLE  变量名 [,变量名…]:数据类型[ :=表达式];
```

其中，“VARIABLE”是用来表示变量的保留字，用来声明一个变量；[:=表达式]用来对变量进行初始赋值，它是一个可选项，赋值符号为“ :=”。

下面是几个变量说明的例子。

```
VARIABLE  temp : bit := '0';
VARIABLE  a,b : integer := '0';
VARIABLE  count : integer RANGE 0 TO 255 := 0;
VARIABLE  enable : STD_LOGIC;
```

在上面的第三个例子中，“RANGE”是用来表示限制数据范围的保留字，“TO”则是用来表示数据范围的保留字。这条程序语句表示将变量 count 的数据范围限制在从 0～255 的整数范围内。

与信号相同，对变量进行说明时可以对它赋予初始值，也可以不赋初始值。如果在说明变量的时候没有赋予初始值，那么则认为它取默认值，即指定数据类型的最左值或者最小值。例如，在上面第四个变量的说明中，这里并没有赋予初始值，那么此时变量的初始值为'0'，因为数据类型 STD_LOGIC 的初始值为'0'。

这里需要**注意**：变量的赋值是直接的、立即生效的，它在某一时刻仅包含一个值，这一点与信号赋值是不同的。因此在变量赋值语句中，不允许出现附加延时，例如：

```
VARIABLE  temp,a : bit;
temp := a AFTER 10 ns;
```

在上面的例子中，“AFTER”是表示附加延时的保留字，它表示将变量 a 的值延时 10 ns 后赋值另一个变量 temp。事实上，这个程序语句是错误的，因为变量赋值语句中不能出现附加延时。

前面提到过，变量的作用范围仅仅是说明它的进程、过程或是函数，而在程序其他部分是无效的。这样有时便会提出一个问题：如何将一个变量的值带出它的作用范围之外？实际上，解决方法很简单，只需要将变量的值赋给一个相同类型的信号，然后由信号带出变量的作用范围即可。

3. 信号(SIGNAL)

在硬件电路中,用导线将元件或模块连结构成电子系统。在 VHDL 中,信号起着导线的作用,将实体与实体、进程与进程连结起来构成系统。通常,将信号视为实际系统中的一条硬件连接线。

通常,信号说明的格式如下所示:

SIGNAL 信号名[,信号名…]:数据类型 [:=表达式];

其中,"SIGNAL"是用来表示信号的保留字,用来声明一个信号;[:=表达式]用来对信号进行初始赋值,它是一个可选项,赋值符号为":="。这里需要**注意**:在 VHDL 程序中,信号赋值的符号与此不同,它应为"<="。

下面来看几个信号说明的例子。

```
SIGNAL clk : bit := '0';
SIGNAL reset : bit := '1';
SIGNAL address_bus : STD_LOGIC_VECTOR(7 DOWNTO 0);
```

由于信号和变量具有一定的相似性,因此读者在某些场合下会将两者混淆。实际上,信号和变量之间,不论在形式上还是在操作过程上,还是有很大差别的。一般来说,信号和变量的区别主要体现在以下几个方面。

(1)信号赋值是有一定延迟的,而变量赋值是没有延迟的。

(2)对于进程语句来说,进程只对信号敏感,而不对变量敏感。

(3)信号在某一时刻除了具有当前值外,还具有一定的历史信息(保存在预定义属性中);而变量在某一时刻仅包含一个值。

(4)信号可以是多个进程的全局信号;而变量只在定义它的过程、函数和进程中可见。

(5)信号是硬件中连线的抽象描述,其功能是保存变化的数据值和连接子元件,信号在元件的端口连接元件;变量在硬件中没有类似的对应关系,主要应用于高层次的建模中。

对于信号赋值和变量赋值的区别,是设计人员经常容易混淆的地方。在 VHDL 中,信号赋值和变量赋值的格式如下所示:

```
信号 <= 表达式;
变量 := 表达式;
```

可以看出,信号赋值和变量赋值分别使用不同的赋值符号:信号赋值使用符号"<=",而变量赋值则采用符号":=";同时信号和变量之间允许相互赋值,但是要保证两者的数据类型一致。

这里需要特别引起**注意**,对于信号赋值来说,在信号赋值的执行和信号值的更新之间至少要有 δ 延迟,只有延迟过后信号才能得到新值,否则保持原值不变;而对于变量来说,赋值没有延迟,变量在赋值语句执行后立即得到新值。下面通过一个小例子来说明信号赋值和变量赋值的区别。

```
P1: PROCESS
VARIABLE a,b : integer;
BEGIN
    WAIT UNTIL clk = '1';
        a := 10;
```