●高等学校教材

VHDL语言及其应用

付永庆

高等教育出版社
HIGHER EDUCATION PRESS

高等学校教材

VHDL语言及其应用

付永庆

高等教育出版社

内容简介

本书是在作者历时七年为通信与信息系统、信号与信息处理专业研究生讲授VHDL语言及其应用课程的教学实践基础上编写而成的。全书共分15章，以教授完整的VHDL语言体系及其用于系统设计的方法为目的，由浅入深、精炼简洁地介绍VHDL语言学习基础、语法规则和模型结构、深入了解VHDL语言、VHDL描述风格和应用系统设计范例等内容，书中有结合实际应用的程序举例，并配有一定数量的习题。

本书内容包括了教授电类专业硕士研究生或高年级本科生VHDL语言知识所需要的全部资料。既可作为研究生教材，也可经适当取舍后作为电子信息、通信、计算机、自动化专业高年级本科生教材，同时还可作为工程技术人员的自学参考书。

图书在版编目(CIP)数据

VHDL语言及其应用/付永庆. —北京：高等教育出版社，2005.5

ISBN 7-04-017195-3

Ⅰ.V... Ⅱ.付... Ⅲ.硬件描述语言，VHDL Ⅳ.TP312

中国版本图书馆CIP数据核字(2005)第029570号

出版发行	高等教育出版社	购书热线	010-58581118
社　　址	北京市西城区德外大街4号	免费咨询	800-810-0598
邮政编码	100011	网　　址	http://www.hep.edu.cn
总　　机	010-58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
经　　销	北京蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印　　刷	北京鑫海金澳胶印有限公司		
开　　本	787×960　1/16	版　　次	2005年5月第1版
印　　张	22.5	印　　次	2005年5月第1次印刷
字　　数	420 000	定　　价	28.10元

物料号　17195-00

前　言

VHDL语言作为一种国际标准化的硬件描述语言，自1987年获得IEEE批准以来，经过了1993年和2001年两次修改，至今已被众多的国际知名电子设计自动化(EDA)工具研发商所采用，并随同EDA设计工具一起广泛地进入了数字系统设计与研发领域，目前已成为电子业界普遍接受的一种硬件设计技术。

VHDL语言用于数字系统设计的主要优点是：

(1)允许用软件描述系统的硬件结构，即描述系统怎样分解为子系统和子系统间怎样互连。

(2)允许使用类似常用编程语言形式的系统功能指标。

(3)允许对系统设计在制造前以低廉的花费进行性能模拟验证。

(4)允许设计的详细结构从更抽象的性能指标出发沿自顶向下的路线分层次地进行综合。

(5)允许设计重用和在可编程ASIC器件上生成设计芯片。

这些优点使得其应用于实际工程设计时，例如：单芯片系统(SOC)的设计、IP_core开发、网上可重配置系统和嵌入式计算等领域，能使设计者把注意力更多地集中在制定设计对策上，并能大大地节省产品投放市场的时间。

随着VHDL的应用领域不断扩大，当前的数字系统设计手段和研发环境已经发生了根本性的改变。过去普遍使用的基于中、小规模集成电路进行板级系统集成的传统设计技术正在被快速发展的基于可编程ASIC器件进行芯片级系统集成的软硬件联合设计技术所取代。受其影响，电子业界对人才的需求也在朝着要求设计工程师具有VHDL及其相关设计工具背景知识的方向调整。由此可见，VHDL语言不仅推动了数字系统设计手段的变革，而且对电类专业的人才培养也将产生重要的影响，未来的电子设计工程师需要掌握VHDL语言知识已是大势所趋。

为了把当前国际上广泛采用的这种数字系统设计技术介绍给国内的研究生们，作者在总结七年多为哈尔滨工程大学通信与信息系统、信号与信息处理专业硕士研究生讲授VHDL语言及其应用课程的教学实践经验的基础上撰写了这本研究生教材。希望此书能同时兼顾为电类专业高年级本科生开设VHDL语言课程的教学需要，并能通过教学过程达到培养和增强他们工程研发能力的目标。

全书共分15章，以教授完整的VHDL语言体系及其用于系统设计的方法为目的，由浅入深、精炼简洁地介绍VHDL基本概念、VHDL语言学习基础、顺序语句、VHDL的模型结构、说明的可见性、决断信号、假名、生成语句、属性和组、寻址类型与链接数据结构、文件与模拟测试、信号延迟的描述、进一步了解VHDL、VHDL描述风格、应用系统设计范例等内容。书中有结合实际应用的程序举例，并配有一定数量的习题。为配合VHDL语言学习和实践，在附录中还对Altera公司的MAX+PLUS Ⅱ VHDL设计工具和作者设计的PLD-1型可编程器件实验箱的使用方法进行了介绍。

考虑到并行性是VHDL语句的主要特点，书中没有把并行语句集中放到一起来讲解。因为如果把并行语句放到一起集中讲解的话，将不利于按直线阅读的方式来组织书中的内容。

需要指出，本书没有涉及与可编程ASIC制造有关的逻辑综合问题。这主要是因为它们太多地依赖IC制造厂商们各自所使用的非标准设计工具，而且忽略这些内容，实际上也并不影响我们深入了解和掌握VHDL语言。

学习本书需要先修数字电路课程。布尔代数、组合逻辑和时序逻辑设计方面的知识是学习本书的基础，如果学习者具有某些计算机硬件与编程方面的背景知识将更加有助于本书的学习。对于研究生或期望通过本教材学习达到较高水平者，还需要有电路与系统、通信与信号处理等方面较为完整的专业知识基础作为支撑。

本书承蒙哈尔滨工程大学的马光胜教授审阅并提出宝贵修改意见，谨此致以衷心的感谢。

作者深知，本书撰写有赖于诸多学者和专家发表的有关文章和著作为作者提供了创作的思想源泉，为此，作者特别向他们表示深切的谢意。

此外，还要感谢我的多名研究生协助我验证了部分设计范例和打印了部分书稿。

关于本书的撰写，作者在主观上倾注了极大的精力，力求严谨和紧密结合工程实际，但由于学识与教学经验的限制，不足与疏漏之处仍恐难免，恳请同行专家和广大读者批评指正。意见请寄：哈尔滨工程大学信息与通信工程学院，邮编150001。

作者

2004年12月31日于哈尔滨

目　录

第 1 章　VHDL 基本概念

本章简要介绍了数字系统建模的概念和设计过程中建模与模拟的重要性，初步描述了 VHDL 程序的基本结构，最后介绍了 VHDL 的设计流程和研发工具情况。

1.1　数字系统建模

若要讨论数字系统建模问题，首先要搞清什么是一个数字系统。工程师们对这个问题的答案经常会因他们工作的领域和背景不同而有很大差别。有些人会把一块集成电路看成是一个自含数字系统，而另一些人则会把整台个人计算机看成是一个数字系统。为便于本书讨论，我们倾向于把能处理或储存信息的任一数字电路定义为一个数字系统（digital system）。于是，这个电路既可作为整体被看成一个数字系统，又可被看成组成数字系统的部件。因此，本书的讨论将涵盖从组成元件的低级门电路到顶层功能单元的宽广领域。

把上述定义的数字系统作为一个整体对其进行研究是极其复杂的，或许我们也不具备直接理解如此复杂问题的能力。因此，需要借助一些有效的方法来化解其复杂性，以便能够设计出满足要求的元件（component）和系统。

解决上述问题的最有效方法是系统化的设计方法。一般该方法总是从一个设计要求开始。首先，通过设计一个抽象结构来满足设计要求；然后，把该结构分解为一系列元件并通过互连来完成相同的功能；接着，每一个元件再被分解直至找到已有的能完成与最底层元件同样功能的图元（primitive element）为止；最终，所获得的结果恰好是一个以图元建造的分层复合系统（composed system）。

这种方法的优点是每一个元件或称子系统（subsystem）都能够被独立地设计。并且在使用子系统时，可以把它只作为一个抽象结构看待而不必考虑它的细节。因此，在设计进程的任一阶段，仅需关注与当前设计有关的少量信息，这就避免了陷入处理大量细节信息的麻烦。

下面用模型（model）这个术语来表示我们对一个系统的理解，即用模型代表与系统相关联的信息和掩盖具体细节后的抽象行为。这意味着同一系统可能

有几种形式的模型，并且每一模型都表示了不同方面的系统特点，例如，模型1用于表示系统的行为；而模型2用于表示由子系统互连组成该系统的方式，即互连结构。

有许多理由要求我们来规范这个模型的概念：

其一，当要求设计一个数字系统时，设计要求必须是具体的。设计者的任务就是要设计一个满足给定要求的系统。为了使设计者顺利完成这项任务，给定的设计要求应该既清晰又易于理解，并希望以此方式使设计者能自由地探索硬件实现方案和根据某一准则选择其中最优的一个。但实际上经常遇到的一个问题是给定的设计要求可能是不完善和模棱两可的，因此，将导致用户和设计者对设计要求理解不一致的情况。这个问题恰好可以通过使用一段标准格式的文本模型说明设计要求的办法来解决。

其二，要求能把系统行为的说明传递给用户。因为设计者不可能预测到系统可能被使用的每一种方式，故不能枚举出所有可能的系统行为。如果设计者能提供一个标准化的模型，那么用户就能以任一给定的输入信号组合来检查和确定系统的性能。由此可见，标准化的模型是一个支持用文本描述系统的理想工具。

其三，要求允许通过模拟进行设计测试和验证。如果设计是从一个表达某系统性能的设计要求开始，那么就可以使用测试输入来模拟其对应模型的行为，同时记录相应的输出结果。于是，可以使用若干子系统来设计一个电路，其中每一个子系统都将具有自己的行为模型。接着，再使用相同的测试输入来模拟这个复合系统，并把这个复合系统的输出结果与前面对设计要求进行模拟得到的记录结果相比较。如果结果相同，便可得知这个复合系统满足设计要求，即能表达所要系统的性能；否则，就需要对设计做出某些修改。可以重复上述过程直至达到元件特性已知的设计底层。这样，当元件被制造时，源于模拟的测试输入和输出还能用于验证实际电路功能的正确与否，并且，这种全程测试与验证方法的优点是能够保证测试输入覆盖最终电路被使用的所有工作环境。

其四，要求允许对一个设计的正确性进行形式验证。形式验证要求对系统性能给出一个数学说明，而说明本身可以用一种标准逻辑系统符号来表示，比如，时序逻辑电路符号。形式验证还需要一个能表示建模语言语义或表示标准逻辑系统符号含义的数学定义。验证过程包含应用逻辑系统推理规则来证明设计隐含着被要求的性能的内容。尽管形式验证至今也没有得到实际应用，但当前它是一个十分活跃的研究领域。

最后，要求允许自动综合电路。如果能形式化地指定被要求系统的性能，理论上，就有可能把设计指标转换成为一个能满足设计要求的电路。这种方法的

优点是特别节省人力投入，能使设计者自由地探索可行的设计方案而不必陷入处理设计细节的琐碎之中。此外，这种方法几乎没有引入设计错误也不需要设计检查，因此，自动综合电路的正确程度会更高。

综上所述，我们有一个共同的目的就是想要在设计进程中用最少的花费和最少的时间获得最大的可靠性。因此，需要设计指标既清楚具体又易于理解、子系统能正确地被使用以及设计能满足性能指标要求。否则，在生产后纠正一个设计错误将会以巨额花费为代价。通过避免错误和为设计进程提供更好的工具，设计花费和延迟就能够被限制在要求的范围内。

1.2 建模的域和级

在1.1节中，曾指出一个系统可以有不同的模型，每一个模型都以不同的角度来描述这个系统。通常，这些模型可以被划分为三个领域，即行为领域(function)、结构领域(structure)和几何领域(geometry)。行为领域与系统完成的运算有关，从某种意义上说，这是最抽象的描述领域，因为它没有提示怎样进行实现。结构领域与系统怎样由互连的子系统组成有关。几何领域与系统怎样布放在物理空间内有关。

每一个领域也都可以被划分为抽象的级(level)，不同的级也称为层。在顶层，只考虑行为、结构和几何领域的粗略视图；而在较低层次，它们的更为精确的细节被依次引入。根据文献[1]，这里每一个领域都被划分为四个级，即四个层次。图1-1给出了领域与级的详细划分。

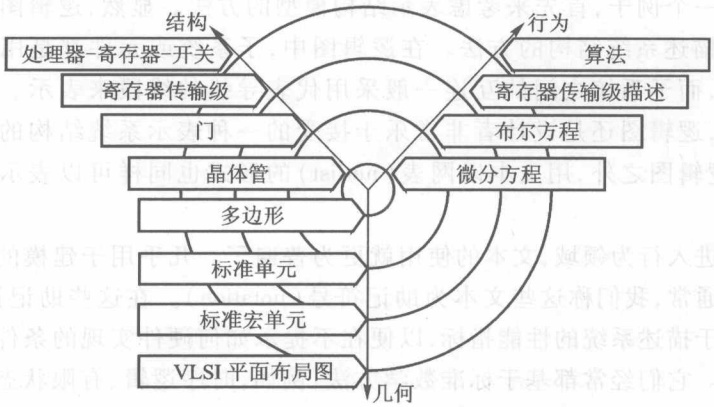

图1-1 领域与级的详细划分图

图中，三个独立的轴线分别表示不同的领域，而同心圆与每一个轴线的交点分别表示不同的级。

行为领域描述一个设计的基本功能，也就是一个系统的输入和输出关系。例如，布尔方程就是组合逻辑电路的行为描述。在行为领域中，我们把对各层行为基于算法的描述称为行为模型（behavioral model）。

结构领域描述一个系统的逻辑结构，也就是说描述一个设计的抽象实现。这个抽象实现实质上就是按功能对系统内部进行划分后所获得的各个子结构之间的互连网表。例如，在寄存器传输级，子结构可能是 ALU、ROM、MUX、寄存器等功能模块单元，抽象实现就是它们之间的互连关系。在结构领域中，我们把对各层互连关系的描述称为结构模型（structural model）。

几何领域描述一个设计的物理实现，或者说描述怎样把结构领域中的抽象元件布放在硅晶片（silicon die）上。通常，芯片速度、面积和功耗均属于几何领域的一部分。

需要指出，在实际工程中，无论哪个领域，各级之间的边界都是难于界定的，因为它们经常是重叠的。实际上，行为领域与结构领域之间有时也存在着模型混合使用的情况。不过，任何设计的最终实现都将是一个物理实现。

1.3 建模语言

通过前面的讨论可知，不同的模型可以用来表达一个系统的行为、结构或几何领域中的不同层次。同样，也有不同的方法可以用来表示这些模型。

作为一个例子，首先来考虑表示结构模型的方法。显然，逻辑图是一种被广泛用于描述系统结构的方法。在逻辑图中，子系统或元件都是用图形符号来表示的，而子系统之间的互连一般采用代表导线的线段来表示。就使用广泛性而言，逻辑图还是设计者非常乐于接受的一种表示系统结构的方法。但是，除了逻辑图之外，用文本以网表（netlist）的形式也同样可以表示系统的结构信息。

一旦进入行为领域，文本的使用就更为普遍了。几乎用于建模的语言都是文本的。通常，我们称这些文本为助记符号（notation）。在这些助记符号中，一部分被用于描述系统的性能指标，以便在不提示如何硬件实现的条件下来表示系统运算。它们经常都基于标准数学方法，例如，时序逻辑、有限状态机等。而另一部分主要用于模拟以达到测试和验证系统功能的目的，它们类似于传统的编程语言。此外，还有一些符号是面向硬件综合的。为便于建模，对它们施加了更多的限制，因为某些编程语言结构难于转换为硬件。

早期的硬件描述语言(hardware description language)简称 HDL 语言,兼有上述提到的用文本描述系统性能的诸多特点,比传统的基于逻辑图和布尔方程描述硬件的设计方法更容易控制不断增加的门级逻辑规模,因此,更适合复杂系统设计和自顶向下的系统设计要求。同时,HDL 语言也为在希望的抽象层次上对设计进行精细和简洁的描述提供了便利。故此,在 VHDL 语言产生之前,HDL 语言为 IC 芯片制造厂商所采用。由于 HDL 的标准仅为厂商所专有,而非国际化,因此,使设计者和 IC 制造厂商之间的信息交换、设计重用乃至设计维护都存在着困难。

VHDL 语言是能够解决早期 HDL 语言缺点的两种国际标准化硬件描述语言之一。它源于美国国防部发起的 VHSIC(very high speed integrated circuits)计划。1987 年 12 月由 IEEE 批准为标准 HDL(IEEE 1076 标准),称为 VHDL'87 版本,1993 年首次被修订为 VHDL'93 版本,2001 年被修订为 VHDL2001 版本。

除了直接定义 VHDL 语言语义的 IEEE 1076 标准之外,为拓展 VHDL 的应用领域和硬件综合能力,自 1987 年以后,还相继有 IEEE 1076.1、IEEE 1076.2、IEEE 1076.3、IEEE 1076.4、IEEE Standard 1164 等标准被 IEEE 正式批准。有关这些 IEEE 标准的简单介绍放在了本书附录 1 和附录 2 之中,希望了解更多信息的读者也可以直接登录 www.eda.org 网站查询。

本书的目的是要以 VHDL 作为建模语言,因为它包含有自顶向下直至门级描述行为和结构的工具,也提供有一个能用几何领域信息注释模型的属性机制。VHDL 原本就是作为一种建模语言用于说明系统性能指标和模拟之目的的。如果限定在能够自动转换为硬件的 VHDL 语言子集上使用它,就能够对其进行硬件综合。由于 VHDL 设计是文本化的,因此也便于文档管理和设计重用。

本书介绍的 VHDL 语言主要以 IEEE 1076 标准为准,内容将涵盖VHDL'87、VHDL'93 和 VHDL2001 三个版本。为了便于讲解,书内提供的所有说明均以 VHDL'93 版本的 VHDL 语言编写。有关 VHDL 语言版本之间的差别,书内主要以提示注意的方式加以解释和说明。

1.4 VHDL 建模的概念

前面几节已经讨论了行为、结构和几何领域中的建模问题。本节要初步介绍一下每个域中的基本建模概念及其对应的 VHDL 基本单元,以便为以后各章的讨论奠定知识基础。

首先以 4 位寄存器为例来讨论 VHDL 描述方式。如图 1-2 所示,模块 reg4

代表着一个 4 位寄存器。

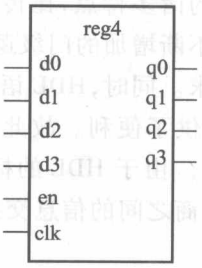

图 1 - 2　4 位寄存器

若用 VHDL 术语,则模块 reg4 被称为设计实体(entity),它的输入和输出称为端口(port)。例 1 - 1 给出了该实体外部界面的 VHDL 描述。

例 1 - 1

```
entity reg4 is
    port (d0, d1, d2, d3, en, clk: in bit;
          q0, q1, q2, q3: out bit);
end entity reg4;
```

实际上,这是一个实体说明(entity description)的示例。它定义了实体的名字,并列表给出了输入和输出端口名以及数据入出实体的方向和位值(**0** 或 **1**)。由此可见,一个实体说明所描述的是实体的外部视图。这也是国外有关 VHDL 的参考书中把 VHDL 设计都看作图而不是文本程序的直接原因。

● 基本行为单元

在 VHDL 中,一个实体的内部实现称为构造体(architecture)。同一实体的行为允许用多种不同描述形式的构造体来实现。一种描述实体实现的方法是建立模型的输入输出关系,即实现模型的算法。行为构造体(behavioral architecture)就是一种以抽象的形式(算法)描述实体内部功能的构造体。通常,行为构造体都是由能按序执行一系列命令的进程语句(process statement)或与简单进程相等效的并行信号赋值语句所组成。这里命令代表着顺序语句(sequential statement)。顺序语句与常见的编程语言非常相似,它们的类型包括计算表达式、变量赋值、条件执行、循环执行和子程序调用等。此外,还包括一条信号赋值语句(signal assignment statement),这是进程中包含的唯一能硬件建模的语句。除了不是在赋值的同时而是要在未来的某个时刻更新信号值之外,信号赋值类

似于变量赋值。

为说明上述概念，现在来研究例 1－2 给出的实体 reg4 的构造体。

例 1－2

```
architecture behav of reg4 is
begin
    storage: process is
        variable stored_d0, stored_d1, stored_d2, stored_d3: bit;
    begin
        if en ='1' and clk ='1' then
            stored_d0 := d0;
            stored_d1 := d1;
            stored_d2 := d2;
            stored_d3 := d3;
        end if;
        q0 <= stored_d0 after 5ns;
        q1 <= stored_d1 after 5ns;
        q2 <= stored_d2 after 5ns;
        q3 <= stored_d3 after 5ns;
        wait on d0, d1, d2, d3, en, clk;
    end process storage;
end architecture behav;
```

在构造体 behav 中，第 1 个保留字 **begin** 之后部分包含一个描述寄存器功能的进程语句。它从名字 storage 开始，至保留字 **end process** 结束。这里，进程语句定义了当系统被模拟运行时要发生的动作序列。正是这些动作控制着实体端口值随时间变化的规律，也就是说，它们控制了实体的行为。进程通过使用信号赋值语句来修改实体的端口值。

进程 storage 的工作方式为：首先，当模拟阶段开始时，信号被置为 **0**，而进程同时被激活。于是，列在保留字 variable 后的进程变量被初始化为 **0**。然后，进程中的语句按顺序执行。第 1 条语句是条件测试以确定信号 en 和 clk 是否同时为 **1**。如果是，则执行保留字 **then** 和 **end if** 间的语句，并使用输入信号的值更新进程变量。在条件 **if** 语句之后，有 4 条信号赋值语句，它们将在 5 ns 以后更新输出信号。

当进程中的所有语句都被执行以后，进程到达等待语句（wait statement）处并被悬挂，也就是说，转为不激活，直至它敏感的信号，即在 **wait on** 列表中的信号 d0、d1、d2、d3、en 和 clk 之一改变数值，进程才会被再次激活，并从保留字