EDA技术实用丛书

基于VHDL的FPGA开发
快速入门·技巧·实例

罗力凡 常春藤 等 编著

书中实例的源文件可到人民邮电出版社网站下载

人民邮电出版社
POSTS & TELECOM PRESS

EDA技术实用丛书

基于VHDL的FPGA开发快速入门·技巧·实例

罗力凡 常春藤 等 编著

人民邮电出版社
北京

图书在版编目（CIP）数据

基于VHDL的FPGA开发快速入门·技巧·实例 / 罗力凡等编著. —北京：人民邮电出版社，2009.5
(EDA技术实用丛书)
ISBN 978-7-115-19685-9

Ⅰ. 基… Ⅱ. 罗… Ⅲ. ①硬件描述语言，VHDL—程序设计②可编程序逻辑器件—系统设计 Ⅳ. TP312 TP332.1

中国版本图书馆CIP数据核字（2009）第011551号

内 容 提 要

本书是一本专门介绍如何快速掌握使用 VHDL 语言开发 FPGA 的方法与技巧的图书。本书从最基本的 VHDL 硬件描述语言讲起，先是通过对 VHDL 概念、语法、基本电路编程方法的讲解，让读者掌握 FPGA 的开发语言；接着通过对 FPGA 开发工具、开发思想、开发技巧的详细阐述，让读者从根本上理解 FPGA 开发的深层内涵；最后通过大量的工程实例，将 FPGA 开发语言、开发工具、开发思想和实际工程实现完美的结合。

本书把读者的实际需求作为内容的切入点，在讲述抽象理论时注重引用实例将理论形象化，在讲述实例时又注重将优秀的设计理念巧妙融合进来。全书讲述清楚，内容由浅入深，书中的大量实例凝结了作者多年的实际开发经验，具有较高的参考意义和实用价值。本书既可作为广大数字电路设计人员的设计指南，也可以作为高等院校电子、通信、计算机类专业的教材和参考书。

EDA 技术实用丛书

基于 VHDL 的 FPGA 开发快速入门·技巧·实例

◆ 编　　著　罗力凡　常春藤　等
责任编辑　王晓明
执行编辑　刘　洋
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市潮河印业有限公司印刷

◆ 开本：787×1092　1/16
印张：19.25
字数：471 千字　　2009 年 5 月第 1 版
印数：1－4 000 册　　2009 年 5 月河北第 1 次印刷

ISBN 978-7-115-19685-9/TN

定价：36.00 元

读者服务热线：(010)67129264　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

本书作者从事数字电路设计工作已有多年，深感基于 VHDL 硬件描述语言的 FPGA 技术在电子信息工程领域的重要性。VHDL 作为当今非常流行的硬件描述语言，已经随着可编程逻辑器件在国内的迅猛发展，深深地吸引了广大电子硬件工程师。用 VHDL 编程实现传统的电路功能已经成为广大电子硬件工程师进行电路设计的首选。用 VHDL 硬件描述语言做电路设计具有开发周期短、设计易于修改、电路简单、成本低廉等优点，对那些外形结构要求小巧的微电子系统，可以直接利用 FPGA 器件的可编程特性来大大减少芯片的使用量，从而提高外形结构设计灵活性及系统可靠性。目前，全球最大的可编程逻辑器件制造商、FPGA 的发明者 Xilinx 公司已经成功推出了 ISE 10.1，这将引导 FPGA 的设计进入一个更快速、更成熟、更具时代前瞻性的崭新阶段。

与此同时，各大公司和高校也纷纷设立了可编程逻辑器件（主要指 FPGA 和 CPLD）设计研究机构，已经有相当一部分电子硬件开发人员和学者在从事与之相关的工作和研究。尤其是在各大高校，可编程逻辑课题组如雨后春笋已经在全国蔓延开来。正是有了这样的现实背景，渴望有一本通俗易懂，有大量实际开发经验的入门指导性书籍已经成为许多在校大学生和在职工程技术人员的迫切愿望。纵观现有书店里的 FPGA 设计指导类书籍，大多注重对设计理论的阐述而对实际的开发经验谈之较少，这对那些需要在短时间内快速掌握 FPGA 开发方法、能尽快上手做设计的初学者来说，无疑是很难在选择参考书上做出果断、准确的选择。

基于 FPGA 的电子设计是一个灵活性、实践性非常强的工作，需要有相当丰富的经验才能设计出具有高可靠性的产品，因此缺乏理论与实践结合的学习不仅会对工程的开发有相当大的负面影响，而且随着学习的深入还容易陷入“死胡同”。正是基于这些宝贵的经验教训，本书的作者特别希望摆脱传统的写作方式对学习者思路的束缚，因而将本书的基调定位为“用理论指导实践，用实践验证发展理论”。

本书从最基本的 VHDL 硬件描述语言讲起，先是通过对 VHDL 概念、语法、基本电路编程方法的讲解，让读者掌握 FPGA 的开发语言；接着通过对 FPGA 开发工具、开发思想、开发技巧的详细阐述，让读者从根本上理解 FPGA 开发的深层内涵；最后通过大量的工程实例，将 FPGA 开发语言、开发工具、开发思想和实际工程完美地结合起来。以下是各章的内容概述。

第 1 章：对 VHDL 进行了概述和基本语言结构讲解，内容包括 VHDL 的起源、优势、适用场合、语言结构特点及描述方法等。

第 2 章：对 VHDL 的基本语法进行了全面系统的讲解，内容包括 VHDL 的数据对象、运算操作符、词法描述及 VHDL 语言中常用的库。

第 3 章：对 VHDL 的主要描述语言进行了全面的讲解，内容包括 VHDL 中顺序描述语句、并行描述语句及属性描述语句。

第 4 章：对数字逻辑中的基本逻辑电路的 VHDL 设计方法进行了详细的讲解，内容包括了常用的组合逻辑电路和时序逻辑电路的设计实例。

第 5 章：对常用的小型数字逻辑电路的设计方法、设计思路、设计步骤进行了细致的讲解，内容包括 7 段数码显示器电路、各类分频器电路、键盘扫描电路、状态机实现电路的详细设计过程。

第 6 章：对 FPGA 的原理及当前发展状况进行了概述，在讲述 FPGA 原理的基础上对当今最主流的 3 个公司的 FPGA 器件的型号和使用特点进行了整体描述。

第 7 章：对 FPGA 的开发工具和开发流程进行了细致完整的讲解，选用 Xilinx 公司的软件集成环境（ISE 9.1），以一个简单的计数器的 FPGA 设计为例全面展示了 FPGA 开发的整个流程。

第 8 章：对 FPGA 常用的设计思想和技巧方法进行了系统全面的阐述，在强调理解概念、灵活选择设计方法的基础上列举了大量的设计技巧和设计注意点，是作者工作经验的总结，具有较大的实际参考意义。

第 9 章：列举了几个较大的实际工程设计的例子，这些例子都是经过了作者的验证并用到了实际工程中。这些例子既包含了常用的计算机接口（如串行通信口、I^2C 接口、PS/2 接口）的设计，也包含了常用的数据存储器（如 FIFO）的设计。除此之外，作者还引用了当前极度火热的数据通信的实例，通过介绍一个话音通信实例和一个通信协议实现的实例，给读者在实际开发中提供了具有借鉴意义的开发范例。

与现有的书相比，本书具有以下主要特点。

（1）注重知识完整性。本书对 VHDL 硬件描述语言进行详细的、系统的讲解；对实际中容易出错的细节进行详细的阐述和例化；对 FPGA 开发理论进行完整系统的讲解；对讲述的所有实例进行了详细的注解。

（2）注重抽象理论形象化。本书用了大量实例、图表来形象生动地对理论进行阐述，将理论巧妙地与工程实践结合，具有便于理解、便于借鉴参考的优点。

（3）注重实例工程的普适性和可实现性。本书所用的所有实例都是作者在实际工作中经过反复推敲，融入了作者实际工作经验并成功用于开发项目的正确实例。所讲述的实例具有风格一致、移植简单、适用性强的特点。读者可以把这些实例方便地移植到自己的实际工程中，这对提高读者的 FPGA 开发速度都将起到较好的作用。

在本书的编写过程中，得到了第一作者的妻子宋丽娜的莫大帮助。可以这么说，如果没有她的支持和鼓励，就没有本书的顺利完成。另外，人民邮电出版社的编辑老师们也在本书的出版过程中做了大量工作，在此向他们表示深深的感谢！

作者力求把本书内容写得准确无误，但鉴于时间仓促和作者水平有限，书中难免存在错误和不足之处，对于本书的任何意见和建议，读者可以通过电子邮件的方式与我们取得联系，作者的邮箱地址是 luosongg@sina.com，本书编辑的邮箱地址是 liuyang@ptpress.com.cn，我们在此表示诚挚的谢意！

目　录

第 1 章　VHDL 语言概述及基本结构

硬件描述语言 HDL 是一种用形式化方法描述数字电路和系统的语言。利用这种语言，设计人员可以在数字电路系统的设计中从上层到下层逐层描述自己的设计思想。在设计过程中，设计人员首先用一系列分层次的模块来表示极其复杂的数字系统，然后，利用电子设计自动化（EDA）工具，逐层进行仿真验证，再把其中需要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。通过这种高层次设计（high-level-design）的方法，设计思想也一步一步从抽象走向了具体。

目前，这种逻辑性强、易于洞察设计每个细节的新方法已被广泛采用。据统计，目前在美国硅谷约有 85%以上的 ASIC 和 FPGA 采用硬件描述语言进行设计。

硬件描述语言 HDL 的发展至今已有 20 多年的历史，并成功地应用于设计的各个阶段：建模、仿真、验证和综合等。到 20 世纪 80 年代，已出现了上百种硬件描述语言，对设计自动化曾起到了极大的促进和推动作用。但是，这些语言一般各自面向特定的设计领域和层次，而且众多的语言使用户无所适从。因此，急需一种面向设计的多领域、多层次并得到普遍认同的标准硬件描述语言。20 世纪 80 年代后期，VHDL 和 Verilog HDL 语言适应了这种趋势的要求，先后成为 IEEE 标准，并得到了广泛的应用。本书将以 VHDL 语言为基础，在详细讨论 VHDL 语言用法的基础上，结合实例探索利用 FPGA 进行电子系统设计的方法。

本章将从 VHDL 的概述谈起，在结合与 Verilog HDL 的比较中带读者走向用 VHDL 语言进行电路设计的世界。

1.1　VHDL 语言概述

1.1.1　VHDL 语言的产生历史

任何一种新事物的产生都有其特定的历史背景，VHDL 硬件描述语言也不例外。20 世纪 50 年代中后期，电子技术在美国国防工业得到了广泛应用。随着电子集成工艺的日新月异和电子设计方法的不断推陈出新，20 世纪 70 年代末至 80 年代初美国国防部提出了著名的 VHSIC（Very High Speed Intergrated Circuit）计划，VHSIC 计划的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完成 10 万门级以上的设计建立一项新的描述方法。

1981 年末，美国国防部又提出了一种新的硬件描述语言（HDL），称为“超高速集成电路硬件描述语言”，简称 VHDL（VHSIC Hardware Description Language）语言。当这个语言被首次开发出来时，其目标只是使电路文本化的一种标准，主要是为了使采用文本描述的设

计能够为其他人所理解，同时也用作模型语言，能采用软件进行模拟。VHDL 语言的结构和设计方法受到了 ADA 语言的影响，并吸收了其他硬件描述语言的优点。1986 年，IEEE 致力于 VHDL 语言的标准化工作，为此成立了 VHDL 语言标准化小组。经历了多次的修改与扩充，直到 1987 年 12 月 VHDL 语言才被接纳为 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 语言来进行描述。1993 年，IEEE 1076 标准被修订，更新为新的 VHDL 语言标准 IEEE 1164。1996 年，IEEE 1076.3 成为 VHDL 语言的综合标准。

1995 年我国国家技术监督局制定的《CAD 通用技术规范》推荐 VHDL 语言作为我国电子设计自动化硬件描述语言的国家标准。至此，VHDL 语言在我国迅速普及，现在这门语言已经成为从事硬件电路设计的开发人员所必须掌握的一项技术。

1.1.2 用 VHDL 语言进行硬件设计的主要优势

时至今日，VHDL 语言已经成为数字电路和系统的描述、建模和综合的工业标准。在电子产业界，无论是系统设计人员，还是 ASIC 设计人员或者各大中院校的学生，都应该通过学习 VHDL 语言来提高他们的工作效率。由于 VHDL 语言的通用性，它现在已经成为支持不同层次设计者要求的一种标准硬件描述语言。VHDL 语言能够成为标准并且获得广泛的应用，是与其拥有强大的优势分不开的。

（1）功能强大灵活性好

VHDL 语言具有功能强大的语言结构，可以用简洁明确的程序来描述复杂的逻辑控制。为了有效控制设计的实现，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；而且它还支持阶层设计和提供模块设计的创建。同时，VHDL 语言还支持同步电路、异步电路和随机电路的设计，其超强的系统级抽象功能是其他的硬件描述语言所难以媲美的。

（2）独立于器件的设计

设计人员采用 VHDL 语言进行硬件电路的设计时，并不需要首先选择完成此项设计的逻辑器件。这样设计人员可以集中时间来进行硬件电路系统的具体设计，而不需要考虑其他的问题。当采用 VHDL 语言完成硬件电路系统的功能描述后，可以使用不同的逻辑器件来实现其功能。如果需要对设计进行资源利用和性能方面的优化，也并不要求设计人员非常熟悉器件的内部结构。这样，设计人员就可以集中精力来进行设计的构思。

（3）具有程序移植能力

VHDL 语言的移植能力是允许设计人员对需要综合的设计描述进行模拟的，在综合前对一个数千门的设计描述进行模拟可以节约大量可观的时间。由于 VHDL 语言是一种标准化的硬件描述语言，因此同一个设计的 VHDL 语言描述可以被不同的 EDA 工具支持，从而使得 VHDL 语言程序的移植成为可能。VHDL 语言的移植能力，就是指同一个设计的 VHDL 语言描述可以从一个模拟工具移植到另一个模拟工具，从一个综合工具移植到另一个综合工具，或者从一个工作平台移植到另一个工作平台。

（4）强大的性能测估能力

独立于器件的设计和可进行程序移植，允许设计人员采用不同的器件结构和综合工具，来对自己的设计进行评估。在设计人员开始具体的设计之前，他们并不需要了解将采用何种

逻辑器。设计人员可以进行一个完整的 VHDL 语言描述，并且可以对它进行综合，生成选定的器件结构的逻辑功能，然后再对设计结果进行测试、评估，最后选用最适合该设计的逻辑器件。同样为了衡量综合的质量，设计人员可以采用不同的综合工具对设计进行综合，然后再对综合结果进行分析和评估。

（5）易于 ASIC 移植

VHDL 语言效率高的重要体现之一，就是如果设计人员的设计是被综合到一个 CPLD 器件或 FPGA 器件，那么就可以使设计的产品以最快的速度上市。当产品的数量达相当的规模时，采用 VHDL 语言能够很容易地帮助设计人员实现转成 ASIC 的设计。有时用于 PLD 的程序可以直接用于 ASIC，并且由于 VHDL 语言是一种 IEEE 的工业标准硬件描述语言，所以用 VHDL 语言设计可以确保 ASIC 厂商生产高质量的器件产品。

（6）语法严谨、规范性强，易于共享和复用

VHDL 语言的语法规范、标准，可读性强。用 VHDL 语言书写的代码文件既是程序，又是文档；既是设计人员进行设计成果交流的交流文件，也可作为合同签约者之间的合同文本。另一方面，由于 VHDL 语言是一种 IEEE 的工业标准硬件描述语言，具有严格的语法规范和统一的标准，因此它可以使设计成果在设计人员之间进行交流和共享。反过来，就可以进一步推动 VHDL 语言的发展和完善。

1.1.3 用 VHDL 语言设计的基本流程

为了保证用 VHDL 语言设计硬件电路系统的可靠性、正确性，设计人员在编写 VHDL 语言程序对硬件电路系统进行设计之前，必须对硬件电路系统的设计目的和设计要求有一个非常明确的认识才行。例如，硬件电路系统的设计功能是什么？硬件电路系统中的信号建立时间、信号保持时间和最大工作频率等技术指标的要求是什么？只有对硬件电路系统的设计目的和设计要求有了明确的认识以后，设计人员才可以选择适当的 VHDL 语言设计方式和实现功能的逻辑器件。

设计人员通常遵循以下设计流程。

（1）按照硬件电路系统设计的要求，抽象出硬件设计的总体模型，并根据硬件系统的宏观内外连接关系，对其定义。这部分主要是抽象出各级模块，并用实体对它们的连线接口进行描述。

（2）分析硬件系统的具体功能，按照功能要求编写能实现硬件系统功能的 VHDL 语言程序。这部分主要就是抽象出不同的结构体进行描述。

（3）对上述两步流程中所设计的 VHDL 语言程序进行功能仿真，并根据功能的实现情况，对程序做修改和优化。这就是通常所说的前仿真。

（4）VHDL 语言程序是对硬件电路的具体描述，当在验证了所编程序能正确实现电路功能之后就应该分析 VHDL 程序的执行效率，这就是通常所说的 VHDL 程序的综合、优化和布局布线。

（5）在进行综合、优化和逻辑布局布线过程中，由于开发软件的一些不可预见性，或是系统连接时序的不一致性，或是缺少必要的人为的约束，容易导致设计功能的异常性。所以再一次的系统级仿真是必要的，这就是通常所说的后仿真。

（6）在完成以上步骤以后，最后的工作就是程序的固化，也就是将设计好的 VHDL 程序

下载到目标器件中。

以上6个方面只是简略描述了用VHDL语言进行硬件开发的一般步骤，具体的实施方法本书将在以后的章节中结合FPGA开发的具体要求做详细阐述。

1.1.4 VHDL语言与Verilog HDL语言的比较

从前面的讲述中可以看出，VHDL语言已经成为一种成功的硬件描述语言，并得到了广大电子硬件设计人员的青睐。本章引言部分讲到了Verilog HDL硬件描述语言，它同样受到了当今电子工程师们的追捧，在20世纪90年代中期它也成为了IEEE标准。下面就这两种常用硬件描述语言的区别加以阐述，希望读者能通过比较，选择出适合自己设计需要的语言。

（1）共同特点

Verilog HDL和VHDL作为描述硬件电路设计的语言，其共同的特点在于：能形式化地抽象表示电路的结构和行为，支持逻辑设计中层次与领域的描述，可借用高级语言的精巧结构来简化电路的描述。此外，它们都具有电路仿真与验证机制以保证设计的正确性，支持电路描述由高层到低层的综合转换，硬件描述与实现工艺无关（有关工艺参数可通过语言提供的属性包括进去），并且都便于文档管理、易于理解和设计重用。正是基于这些共同点，很多设计人员在设计的时候同时采用这两种高级语言进行混合编程，这也从一定程度上融合了两种语言各自的优点。

（2）各自的特点

由于VHDL早在1980年美国国防部就开始用来对电子系统进行设计，并于1987年就被IEEE（Institute of Electrical and Electro nics Engineers）制定为标准，至今已有将近30年的应用历史，因而VHDL拥有广泛的设计群体，成熟的资源也远比Verilog HDL丰富。与Verilog HDL相比，VHDL的最大优点是：它是一种系统级抽象非常强的硬件描述语言，能通过严谨的语法规范地描述硬件设计的整个过程。由于其强烈的硬件相关性，对于特大型的系统级数字电路设计，VHDL的优势更为明显。Verilog HDL语言也是一种实用性很强的硬件描述语言，它的主要特点是：Verilog HDL的语法要求比较灵活，有很强的软件特性，如果初学者有一定的C语言编程基础，就能在较短的时间内掌握其编程方法；另外Verilog HDL语言的直观性较强，在门级开关电路描述方面比VHDL有优势。当然，这两种语言还在不断的完善过程中，单从语言优劣的角度看，其实二者是旗鼓相当的，都是当今业界广泛应用的硬件描述语言。

（3）二者的使用范围

Verilog HDL和VHDL都是被电子设计人员广泛采用的硬件语言，二者都可以用形式化方法描述数字电路和系统。一般说来，两种语言有着共同的使用范围，换句话说，在能使用VHDL设计的地方可以使用Verilog HDL，在能使用Verilog HDL设计的地方也一定能使用VHDL。不仅如此，两种语言在设计的时候还能混合使用，充分发挥二者在描述不同电路时各自的优势。对于初学者来说，建议根据自己当前背景进行选择。如果先前是以硬件为主的，有较强的硬件设计功底可以先学VHDL；如果先前有一定的软件基础比如C编程基础，可以选择先学Verilog HDL。由于两种语言都是基于硬件设计的，它们有很强的相似性，学会一种语言后一般来说很容易掌握另一种语言，所以初学者没有必要在两种语言的选择上大费周折，一旦选择了学其中一种语言就应该持之以恒熟练掌握。

1.2 VHDL 语言程序的基本模型结构

1.2.1 VHDL 语言程序的基本结构单元

通常 VHDL 语言程序是对一个设计单元进行描述的，这个设计单元就是所谓的设计实体。一个基本设计单元只能唯一地对应一个设计单元或者说是一个基本设计实体，它可以是一个复杂的数字电子系统，也可以是一个数字单元或芯片，甚至还可以是一个简单的基本门电路。但是，不管是复杂的数字电子系统，还是简单的门电路，它们的基本构成都是一样的。它们都是由实体说明和结构体两部分组成的。虽然说一个实体说明可以对应多个结构体，但是通常要求一个独立的 VHDL 语言程序文件只能由一个实体说明和一个结构体组成，这称为实体结构体对。

下面以一个简单的 2 选 1 电路的 VHDL 语言程序为例来简述一下 VHDL 语言程序的基本结构。

【例 1-1】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY mux IS
GENERIC(m:TIME : =1ns);
      PORT(d0 :IN BIT;
           d1 :IN BIT;
           sel: IN BIT;
           q :OUT BIT);
END mux;
ARCHITECTURE rtl OF mux IS
SIGNAL tmp : BIT;
BEGIN
      PROCESS(do,d1,sel)
      VARIABLE tmp1,tmp2,tmp3: BIT;
      BEGIN
          tmp1:=d0 AND sel;
          tmp2:=d1AND (NOT sel);
          tmp3:=tmp1 OR tmp2;
          tmp<=tmp3;
          q<=tmp AFTER m;
      END PROCESS;
END rtl;
```

在说明上面 VHDL 语言程序的结构之前，先介绍一下保留字的概念。所谓保留字就是指一些在 VHDL 语言中具有专门定义的特殊字符。一般来讲，读者可以根据自己的习惯来确定保留字的书写格式，本书中保留字采用大写形式。

1. 实体

在上述程序（例 1-1）中的第 3 行到第 9 行是 2 选 1 电路单元的实体说明部分，其格式是采用保留字 ENTITY 起始于第 3 行，终止于第 9 行。实体说明部分给出了 2 选 1 电路的实体名和输入输出端口：第 3 行给出了实体名 mux；第 4 行是类属说明；第 5 行至第 8 行给出了 2 选 1 电路的输入输出端口说明；第 9 行是实体说明部分的结束行。下面就实体的这几个方面做具体说明。

实体说明的格式为：

```
ENTITY<实体名>IS
[类属说明;]
[端口说明;]
[实体申明部分;]
[实体语句部分;]
END [ENTITY] <实体名>;
```

实体说明部分从“ENTITY<实体名>IS”开始至“END [ENTITY]<实体名>”结束，其中[ENTITY]选项是 93 版 VHDL 标准语法的要求。需要说明的是，“<>”中的部分表示是必选项，不可缺省；“[]”中的部分表示是可选项，如无特殊声明，可以不用。此项说明适合于全书，以后的描述中均遵循此项说明，请读者留意。

（1）类属说明

类属说明用于为设计实体和其他外部环境通信的静态信息提供通道，可以定义端口大小、实体中元件的数目以及实体的定时特性等。它是实体说明的一个可选项。

类属说明的基本格式为：

```
GENERIC （[CONSTANT]参数名：[IN]子类型标识[:=静态表达式]; ……）;
```

如例 1-1 中的第 4 行，这个类属说明是指在编写 VHDL 语言程序时，设计实体的结构体内参数 m 的值为 1ns。

类属说明语句的主要作用是可以增强 VHDL 程序的通用性，避免程序的重复书写。

（2）端口说明

端口说明用于为设计实体和其外部环境通信的动态信息提供通道，它在实体说明中也是一个可选项。端口说明描述的是设计实体与外部的接口，其功能相当于电路图符号中的一个引脚。实体说明中的每一个 I/O 信号被称为一个端口，一个端口就是一个数据对象。端口可以被赋值，也可以被当作变量用在逻辑表达式中。

定义于一个实体的一组端口称为端口说明。端口说明的基本书写格式为：

```
PORT ([SIGNAL]端口名：[模式]子类型标识[:=静态表达式]; ……);
```

如例 1-1 中的第 5 行至第 8 行，这个端口说明描述了 2 选 1 电路的输入输出端口，它是与外部进行信息交换的窗口。由此也可以看出，端口说明的组织结构必须有一个名字、一个通信模式和一个数据类型。其中，端口名字就是指定端口的标识符；通信模式用来说明数据、信号通过该端口时的流动方向；数据类型标识用来说明经过该端口的数据、信号的数据类型。

通常端口名字是由字母、数字和下划线组成的。虽然没有严格的端口命名规则，但命名习惯最好遵循惯例，同时端口的名字也最好能反映实际意义，这样会增强程序的可读性，例如时钟信号端口名字一般采用 CLK，复位信号端口名字一般采用 RESET。需要注意的是，端口名字在一个设计实体中必须是唯一的，不能重复。端口说明中的通信模式有 5 种，即输

入通信模式、输出通信模式、缓冲通信模式、双向通信模式和链接通信模式。

① 输入通信模式。输入通信模式采用保留字 IN 来表示，它表示数据或信号只能从实体外部通过实体的端口流向实体内部，而不能从实体端口流出。一般情况下，设为输入通信模式的端口信号有时钟信号、复位信号、使能信号、数据控制信号、地址输入信号以及单向的数据输入信号等。

② 输出通信模式。输出通信模式采用保留字 OUT 来表示，它表示数据或信号只能从设计实体内部通过实体的端口流向实体的外部。一般情况下，设为输出通信模式的端口信号有计数器输出信号、寄存器输出信号、单向数据输出信号以及控制其他单元的信号等。

③ 缓冲通信模式。缓冲通信模式采用保留字 BUFFER 来表示，它表示数据或信号既可以从设计实体内部通过实体的端口流向实体的外部，同时也可以将流出实体端口的信号或数据引回到设计实体，从而将该数据或信号用于内部反馈。

内部反馈的实现方法是将设计实体的一个端口设定为缓冲模式，同时在实体内部建立内部结点。当设计实体的某一端口既需要输出又需要反馈时，该相应端口就应该设为缓冲通信模式。设为缓冲通信模式的端口信号驱动源来自于设计实体的内部，或者来自其他设定为缓冲通信模式的端口。

④ 双向通信模式。双向通信模式采用保留字 INOUT 来表示，它表示数据或信号既可以从设计实体内部通过实体的端口流向实体的外部，同时也可以从实体外部通过实体的端口流向实体内部。需要注意的是，虽然双向通信模式可以代替输入通信模式、输出通信模式和缓冲通信模式，但是建议读者们不要这样做。双向模式通常在具有双向传输数据功能的设计实体说明中使用，例如含有双向数据总线的设计单元。

⑤ 链接通信模式。链接通信模式采用保留字 LINKAGE 来表示，它表示实体端口无指定方向，可以与任意方向的数据或信号相连。

VHDL 语言具有丰富的数据类型，除了预定义的数据类型外，设计人员还可以自己定义数据类型，这种规定将给 VHDL 语言程序的设计带来极大的方便。常见的数据类型主要有 BIT、BIT_VECTOR、BOOLEAN、INTEGER、枚举类型和物理类型等。而在 VHDL 语言中经常使用的数据类型是由 IEEE STD_LOGIC_1164 程序包提供的 STD_LOGIC、STD_LOGIC_VECTOR。这里需要注意的是，设计人员使用 STD_LOGIC_1164 程序包提供的数据类型时，必须要在实体说明部分加上如下两条语句：

```
LIBRARY IEEE;
USE IEEE STD_LOGIC_1164.ALL;
```

这是 VHDL 语言的库文件，在下面的章节中将进行详细介绍。

2. 结构体

如例 1-1 所知，程序中的第 10 行到第 22 行是 2 选 1 电路单元的结构体部分，它的格式是采用保留字 ARCHITECTURE 起始于第 10 行，终止于第 22 行。结构体部分描述了 2 选 1 电路的逻辑电路关系：第 10 行给出了结构体名 rtl，并给出了与实体说明的匹配关系，表示结构体 rtl 属于实体 mux；第 11 行是在结构体内定义的一个 BIT 型内部信号 tmp；第 12 行是结构体的开始部分，用保留字 BEGIN 表示；第 13 行至第 21 行用一个进程描述语句描述了 2 选 1 电路的逻辑功能；第 22 行是结构体部分的结束行。

这里需要特别提醒读者的是，VHDL 语言程序中的字符是不区分大小写的，这与一些高

级编程语言是有区别的，但是在实际编程中为了增加程序的可读性，通常用大写形式表示保留字，其他部分用小写形式表示。下面就结构体的几个方面作具体阐述。

结构体说明的格式为：

```
ARCHITECTURE <结构体名> OF <实体名> IS
[定义语句； ]
BEGIN
<并行处理语句>
END [ARCHITECTURE] <结构体名>;
```

从上面的格式可知，结构体部分从“ARCHITECTURE <结构体名> OF <实体名> IS”开始至“END [ARCHITECTURE] <结构体名>”结束。需要说明的是[ARCHITECTURE]选项是93 版 VHDL 标准语法的要求，此项是可选的。

（1）结构体命名规则

结构体名是对本结构体的命名，它是该结构体的唯一名称，通常是由设计人员自由命名的。由上述结构体基本格式所知，OF 后面的实体名表明了该结构体所对应的实体。需要说明的是一个实体可以对应多个结构体，这通常对应了实体不同的功能描述。

尽管结构体的命名可以是设计者随心所欲，但是在实际设计中还是应该遵循一些惯例的，比如，采用行为描述方式的结构体一般命名为 behave；采用数据流描述方式的结构体一般命名为 dataflow 或是 rtl；采用结构体描述方式的结构体一般命名为 structural。通过这样的命名可以使阅读 VHDL 语言程序的人一目了然地看懂程序所采用的描述方式，但是在实际设计中这 3 种描述方式通常也没有明确的区分，有时候还有 3 种描述方式混用的情况，所以单纯只看描述名并不能帮助我们正确地读懂程序，之所以要遵循一些惯例只是告诫初学者在学习的初期就应该重视要养成良好的编程习惯。

（2）定义语句

从结构体部分的基本书写格式可以看出，定义语句位于 ARCHITECTURE 和 BEGIN 之间，如例 1-1 中的第 11 行就是一个定义信号的语句。定义语句通常是对结构体内部所使用的信号、常数、数据类型还有函数等进行定义，这里要特别强调的是这些定义语句的作用范围仅限于所定义的结构体内。举个例子来说，前面讲过一个实体可以对应多个结构体，假如我们在其中的一个结构体内定义了信号、常数，这些信号、常数并不能在另外的结构体中使用。初学者往往容易混淆定义语句的使用范围，这一点一定要引起重视。

（3）并行处理语句

并行处理语句位于结构体描述部分的 BEGIN 和 END 之间，如例 1-1 中第 12 行至第 22 行就是一个并行处理语句，并行处理语句是 VHDL 语言设计的核心，它描述了结构体的行为及连接关系，反映了这个设计的功能。一般来说，结构体中的并行处理语句主要包括块语句、进程语句、信号赋值语句、子程序调用语句和元件例化语句，下面简单介绍这 5 种并行处理语句的基本组成和功能。

块语句是由一系列并行语句构成的结合体，它的功能是将结构体中的多个并行处理语句组成一个或者多个子模块。

进程语句主要用于定义顺序语句模块，用来将从外部获得的信号或者内部数据向其他的信号进行赋值。

信号赋值语句用于将设计实体内的处理结果向定义的信号或者实体说明端口进行赋值。

子程序调用语句可以调用过程或者函数，并将获得的结果赋值于信号。

元件例化语句主要用来对其他的设计实体做元件调用说明，并将调用元件的端口与其他的元件、信号或者是高层实体的端口连接。

关于上述 5 种并行处理语句的具体内容将在后续章节中详细讨论。下面给出一个全加器设计的实例，希望读者通过分析实体和结构体的内容，熟练掌握 VHDL 语言程序的基本结构。

【例 1-2】

```
LIBRARY IEEE;
USE IEEE STD_LOGIC_1164.ALL;
USE IEEE STD_LOGIC_UNSIGNED.ALL;
ENTITY   full_adder IS                                    --实体说明
    PORT (a,b,cin: IN BIT;                                --输入端口说明
          s,cout : OUT BIT);                              --输出端口说明
END full_adder;
ARCHITECTURE behave OF full_adder IS                      --行为型构造体说明
BEGIN
    PROCESS(a,b,cin)                                      --引入进程语句
    VARIABLE ai,bi,ci,si: INTEGER;                        --定义内部变量
      BEGIN
          IF a='0'THEN
             ai:=0;
          ELSE
             ai:=1;
          END IF;
          IF cin='0'THEN
             ci:=0;
          ELSE
             ci:=1;
          END IF;
          Si:=ai+bi+ci;                                   --结构体具体功能描述语句
          CASE si IS
              WHEN   0 => s<='0'; cout<='0';
              WHEN   1 => s<='1'; cout<='0';
              WHEN   2 => s<='0'; cout<='1';
              WHEN   3 => s<='1'; cout<='1';
              WHEN   OTHERS => s<='X'; cout<='1';
          END CASE;
    END PROCESS;
END behave;
```

1.2.2 VHDL 语言结构体的 3 种描述方法

从前面所讲结构体命名内容可知，设计人员通常用 behave、dataflow（rtl）、structural 这 3 个单词来对结构体进行命名，事实上，这 3 个单词代表了结构体描述的 3 种方法。