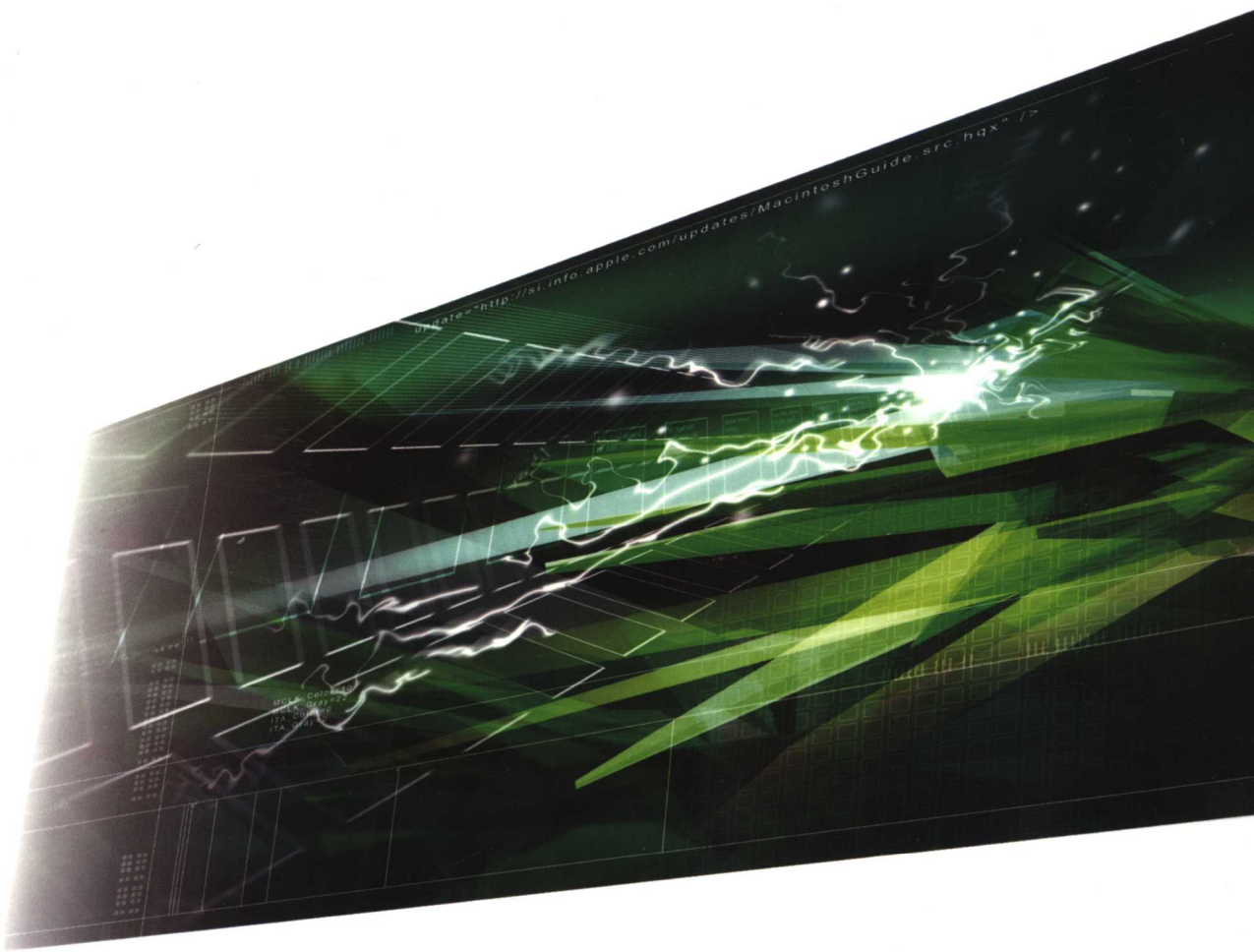
可编程逻辑器件快速进阶丛书

VHDL与硬件实现速成

任勇峰 庄新敏 编著



国防工业出版社
National Defense Industry Press

可编程逻辑器件快速进阶丛书

VHDL 与硬件实现速成

任勇峰　庄新敏　编著

国防工业出版社
·北京·

图书在版编目(CIP)数据

VHDL与硬件实现速成/任勇峰,庄新敏编著.—北京:国防工业出版社,2005.7
(可编程逻辑器件快速进阶丛书)
ISBN 7-118-04004-5

Ⅰ.V… Ⅱ.①任… ②庄… 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(2005)第070156号

国防工业出版社出版发行
(北京市海淀区紫竹院南路23号)
(邮政编码 100044)
国防工业出版社印刷厂印刷
新华书店经售
*
开本 787×1092 1/16 印张 13 300千字
2005年7月第1版 2005年7月北京第1次印刷
印数:1—3000册 定价:23.00元

(本书如有印装错误,我社负责调换)
国防书店:**(010)68428422** 发行邮购:**(010)68414474**
发行传真:**(010)68411535** 发行业务:**(010)68472764**

前　言

随着 CPLD/FPGA 等超大规模可编程数字集成电路与 VHDL/Verilog HDL 等硬件描述语言的飞速发展，数字系统的设计从电路原理图设计转向硬件描述语言设计，从传统的硬件设计方法转向计算机软硬件协同设计的方法。如 Altera 公司和 Xilinx 公司的 EDA 软件环境，可以辅助用户完成从简单的数字逻辑电路到复杂的数字逻辑系统的设计，而且可以随时更改设计。CPLD 和 FPGA 的厂商免费提供 EDA 软件设计环境，学习者可以充分利用它们开发出复杂的数字系统，所以，VHDL、CPLD/FPGA 等技术近年来持续火热。

学过计算机语言的人都有这样的体会，要掌握一门计算机语言，单纯钻研书本中的语法和例子的学习效果并不好，而边学习语法和实例边上机实践则有事半功倍的效果。一方面，上机实践过程中每个实例的调试和仿真成功能产生成就感，激发学习者的兴趣；另一方面，上机实践过程可以修正和完善读者对语言的认识，并进一步巩固知识。学习 VHDL 也是这样，最好读者边看书学习边上机练习。本书正是以此作为指导思想，希望读者通过这种方法快速、初步掌握 VHDL。书中完整的实例都有仿真结果，便于读者提高学习效率，尽快了解从编程、编译、仿真、选配器件到数据下载的完整过程。

在硬件设计中，HDL 和 CPLD/FPGA 等器件的发明使硬件设计工作的效率得到极大地提高，也大大降低了硬件设计的难度。因此，VHDL 及 CPLD/FPGA 受到广大工程技术人员和学生的青睐。但是 CPLD/FPGA 技术仅仅降低了硬件设计的入门门槛，初学者入门后，会发现在具体的硬件实践中仍然有一系列问题待解决。因此本书针对以 CPLD 和 FPGA 器件为主的硬件电路实现过程中，可能遇到的常见问题，简洁明了地介绍了相关的知识和解决办法。

在数字系统设计中，竞争和冒险是困扰广大设计人员的顽疾，本书注意到了这一点，并在最后一章进行了讨论，同时给出了一些有效的解决办法。

衷心感谢研究生刘鑫同学和徐文强同学。本书大部分实例的软件仿真与验证是由刘鑫完成的，徐文强对第 7 章的图形、表格和公式进行了编辑。同时还要感谢在编写此书过程中给予了热情帮助的所有老师和同学。

由于作者水平有限，书中出现错误和疏漏在所难免，希望广大读者予以批评指正。欢迎通过 renyongfeng@nuc.edu.cn 和作者交流。

作　者

2005.5

目　录

第1章 概 述

随着电子技术的迅猛发展，电子电路设计正朝着速度快、容量大、体积小、重量轻、微功耗的方向发展。而推动它的正是日趋完善的CPLD、FPGA和ASIC这些可编程逻辑器件(PLD)设计技术。在单块芯片上实现一个数字系统的全部功能，即系统级芯片(System Level IC，SLIC；或System-on-Chip，SoC)的设计制造与集成电路芯片设计相结合，使得设计行业能更快、更好地设计出性能更优良、功能更完善的PLD产品。而硬件描述语言HDL(Hardware Description Language)在数字系统的硬件设计中成为电子设计自动化(EDA)的一种关键技术。作为IEEE标准的硬件描述语言VHDL(VHSIC Hardware Description Language)已在设计中得到了广泛的应用，且影响日益深远。

1.1 硬件描述语言

硬件描述语言(Hardware Description Language，HDL)，是描述硬件电路的功能、信号连接关系及时序关系的语言。硬件描述语言比电路原理图更能有效地表示硬件电路的特性。硬件描述语言的最大特点是可以借鉴高级编程语言的功能特性，对硬件电路的行为和结构进行高度抽象化和规范化的描述，同时还可以对硬件电路的设计进行不同层次、不同领域的模拟验证和综合优化处理，从而使硬件电路的设计达到高度自动化。

随着大规模专用集成电路ASIC(Application-specific IC)以及CPLD、FPGA的开发和研制，为了提高开发的效率、增加已有成果的可继承性，各生产厂家相继开发了用于各自领域的硬件描述语言。其中最有代表性的是美国国防部开发的VHDL、Viewlogic公司开发的Verilog HDL以及Altera公司开发的AHDL。VHDL在1987年被接纳为IEEE 1076标准，并且在1993年进行了扩展，修订为新的VHDL标准IEEE 1164。1996年，IEEE 1076.3成为VHDL的综合标准。1995年，中国国家技术监督局发布的《CAD通用技术规范》(GB/T17304)中也明确推荐VHDL作为我国电子设计自动化硬件描述语言的国家标准。

1.2 VHDL简介

VHDL是VHSIC(Very High Speed Integrated Circuit)Hardware Descriptions Language的缩写，即超高速集成电路的硬件描述语言。VHDL能够描述硬件电路的结构、行为与功能。虽然其硬件的相关语法与形式类似于一般程序语言，但是涉及许多与硬件关联的语法构造。VHDL设计的层次性，即自上而下的结构式设计方法，适合大型设计工程的分工合作。在主要的系统结构、元件及相互间的连接方式确定以后，就可以将工作分包下去，各自独立进行，例如使用主程序外的元件(component)、函数(function)以及程序内的模块(block)程序。

VHDL 借用 Pascal 和 Ada 等软件编程语言的结构化电路的描述，具有电路模拟与验证机制，确保设计的正确性，支持电路描述由高层向低层的综合变换，易于理解和重用。VHDL 还是一种与实现技术相独立的语言，既不束缚于某一特定的模拟程序或数字装置上，也不把设计方法强加于设计者。它允许设计者在其使用范围内选择工艺和方法。为了适应未来的数字硬件技术的发展，VHDL 还提供了便于将新技术引入现有设计的潜力。VHDL 的最大特点是描述能力极强，可覆盖逻辑设计的诸多领域和层次，并支持众多的硬件模型。VHDL 的特点包括如下几个方面。

(1) 可以分层次设计。

(2) 每个设计单元，既有定义好的接口(以便连接其他元件时使用)，又有明确的行为规范(用来仿真)。

(3) 用算法或者实际硬件结构来定义一个元件操作的行为规范。例如，一个元件最初可以用算法来定义，在高层次设计、检验时使用，仿真通过之后，可以用硬件结构代替算法定义，以实现实际电路的设计。

(4) 并发性：用硬件描述语言所描述的实际系统，其许多操作是并发执行的。

(5) 逻辑操作和设计的时序行为都能仿真。

因此，VHDL 作为一种文件和模块语言，允许明确地指定和仿真数字逻辑系统的行为。

由于商用 VHDL 综合工具的发展，VHDL 的使用也随着仿真环境的大大改善而更加广泛。有些 VHDL 程序，可以直接从 VHDL 的行为描述中创建逻辑电路结构。使用 VHDL 可以在一个芯片上设计、仿真和综合任何从简单到复杂的电路系统。

1.3 VHDL 设计硬件电路的优点

当前使用最广泛且被公认的硬件描述语言是美国国防部开发的 VHDL，VHDL 与其他 HDL 相比有如下一些优点。

1. 设计技术齐全，方法灵活，支持广泛

VHDL 具有强大的语言结构，可以用简洁明确的程序描述复杂的逻辑功能。设计也非常灵活，既支持自顶向下和基于库(library-based)的设计方法，也支持自底向上的设计，而且还支持同步电路、异步电路以及其他随机逻辑电路的设计。其范围之广是其他 HDL 无法比拟的。另外，由于 VHDL 早在 1987 年 12 月就已作为 IEEE 1076 标准公开发布，因此，目前大多数 EDA 工具几乎在不同程度上都支持 VHDL。这样就给 VHDL 的进一步推广和应用创造了良好的环境。从设计层方面来说，VHDL 既支持模块化设计，也支持层次化设计。

2. 系统硬件描述能力强

VHDL 具有多层次描述系统硬件的能力，可以从系统的数学模型直到门级电路。并且，高层次的行为描述可以与低层次的寄存器传输语言描述和门级描述混合使用。另外，VHDL 还可以自定义数据类型，给设计人员以很大的自由度。

3. VHDL 可以与工艺无关地进行编程

设计人员用 VHDL 进行硬件电路设计时，没有嵌入与工艺相关的信息(当然，这些信息也是可以用 VHDL 来编写的)。与大多数 HDL 的不同之处是，当 VHDL 的门级或门级以

上层次的描述通过模拟验证之后，再用相应的工具将设计映射成不同的工艺(如 MOS、CMOS 等)。这样，在工艺更新时，就不用修改设计的程序，只要改变相应的工艺映射工具即可。

4. 语言标准、规范，易于共享和复用

VHDL 已作为一种 IEEE 的工业标准，设计的成果便于重用和交流，反过来又能进一步推动 VHDL 的推广和完善。另外，VHDL 具有严格的语法规范，给编译、阅读及使用都带来极大的方便。

5. 设计周期短，投资风险小

由于 VHDL 设计的数字系统可以仿真验证以及可做最佳化选择，可以缩短设计周期，并减少投资风险。

1.4 VHDL 与计算机软件语言的比较

VHDL 与计算机软件语言虽然都属于软件语言，但二者各有特点。软件是相对于硬件而言的。计算机软件包括机器运行所需要的各种程序及其有关资料，例如汇编程序、编译程序、操作系统、诊断程序、控制程序、专用程序包、程序库程序、数据管理系统、各种维护使用手册、程序说明和框图等。软件是计算机在日常工作时不可缺少的，它可以扩大计算机的功能和提高计算机的效率，是计算机系统的组成部分。而 VHDL 是为数字电路的建模和模拟(simulation)而制定的，是一种面向模拟、针对硬件的语言。它的语法中有许多方面均考虑到模拟与硬件的因素，包括 VHDL 的硬件相关结构、并发特征和混合级描述以及混合级模拟。

1. VHDL 中的硬件相关结构

VHDL 具有许多与数字电路结构直接相关的概念，其中最主要的是元件(component)，它是数字硬件结构——“黑盒”或“模块”的抽象。VHDL 中的元件由实体和结构体两部分共同描述完成。其中实体描述元件与外部环境的接口，其内部行为及结构是隐蔽的。实体的功能定义在称为结构体的单元中，结构体规定实体电路的输入、输出以及相互之间的行为与功能。一个实体可以存在多个对应的结构体，分别以行为风格、数据流风格、结构化风格以及各种风格混合的描述方法来实现。元件的存在使 VHDL 脱离普通程序语言的范畴，成为描述数字电路的专用硬件设计语言。

2. VHDL 的并发性

计算机软件程序一般按书写的顺序依次执行，而 VHDL 却具有并发性。VHDL 的并发性体现在两个方面，首先在使用 VHDL 进行数字电路设计时存在并发性，即 VHDL 支持设计分解，可使被分解的各个子部分的设计并行完成。一个模型的设计主要由 3 部分组成：元件库部分——USE 说明区；实体部分——确立模型与环境的接口；结构体部分——描述元件的行为或功能，为模型生成测试向量，并捕获模型输出信号状态以供分析。在设计流程方面，在系统分析阶段，系统分析者可将设计对象分为若干独立的子元件，交给若干设计小组实现。在此阶段，系统设计者严格定义元件接口，并将元件之间的相互作用以文档形式提供给设计小组。然后，各设计小组可独立并行地对子元件进行详细设计，并模拟验证子元件，确保正确无误。最终，系统设计者集成各子元件形成完整的

设计，对整个设计进行模拟验证。设计的并发性可极大地加快整体设计进程并提高设计质量。

其次，VHDL之所以称为硬件描述语言，很重要的一点是因为它在模拟执行上具有并发性，这一点很适于描述硬件电路活动的并发性特点，是其他程序设计语言所不具备的。VHDL中的进程类似于UNIX操作系统的进程概念，它们的挂起、活动均是独立的。并发性使得VHDL的设计模拟可在并行机上进行，大大提高了模拟效率，是解决模拟时间瓶颈的方法之一。在并行模拟中，VHDL源描述程序经编译后，除了完成通常的功能外，还将每个进程静态地映射到特定的处理器上，计算的加载、通信频带的估算均来自于暂时性的分析及波形传播分析，以帮助获得合理的平衡的静态映射。

3. VHDL与C++的类比

模拟算法可分为解释型和编译型两种。解释型模拟器中存在一个模拟核心，它不断读取VHDL源描述或其编译后的中间格式数据，并对每一条语句解释性地执行。其优点是概念清晰、关系明确，并且编程实现要简便一些；其明显的缺点是对每一条语句都要重新解释并执行，会带来很多不必要的预处理，效率很低。编译型模拟器将VHDL源描述直接转化为功能等价的可执行的二进制代码，这样，在每一条语句的模拟执行过程中消除了多余的预处理，极大地提高了模拟效率并改善了系统性能。

编译型模拟器将VHDL源描述转换为功能等价的C++源代码，这种策略主要是基于对VHDL和C++语言语法特点的比较后得出的。从语言层次上讲，这两种语言都属于高级语言，在许多方面均具有相似性，可以从这种相似性出发，完成它们之间的转化。这些相似性主要体现在如下几个方面。

(1) VHDL中的实体(entity)与C++中的类(class)概念类似。对于VHDL中的任一实体，均可以将它翻译成C++中的一个类。而与实体对应的结构体(architecture)则可以从前面的实体类中派生，这样它就可以共享在实体类中定义的所有数据。

(2) VHDL中的进程(process)继续从相对应的结构体类中派生，这样它就可共享所有的结构体类中定义的数据。当一个结构体包含多个进程时，每一个进程都是结构体的派生类，为了避免同一结构体对象的重复定义，可以通过虚(virtual)基类的方式进行派生。

(3) VHDL中的端口概念可以映射到C++中的函数(包括构造函数)参量，类属则可以映射为函数的默认参量。

(4) 可以为一些VHDL特有的数据类型使用C++定义类的方式定义一个新的数据类型。例如，VHDL中的信号(signal)可为其定义一个类Qsignal，在类中保存信号的属性值，例如上次信号变化的时间及值等。这样，对于信号类型对象赋值的特殊性就可以通过等号运算符的重载来解决。此外，信号的一些预定义属性的运算也可以转化为对类属对象的某个成员函数的调用求解过程。

(5) VHDL的层次关系的实现方式主要由元件说明、例示和组装完成。可以对应于C++语言中的成员类对象的概念。对于不同结构体的组装，则可以在定义成员类对象时，通过给出不同构造函数的参量选取不同的结构体。

(6) VHDL行为描述中的几种顺序语句：变量赋值语句、if语句、case语句、loop语句、next语句、过程调用语句等，在C++语言中均有类似的语法或库函数，做少量的修改即可进行转换。

1.5　设 计 流 程

在使用 VHDL 设计之前，有必要先了解整体 VHDL 的设计过程。在 VHDL 的基本设计过程中，有几个步骤通常叫做设计流程。这些步骤适用于任何硬件描述语言的基本设计过程，用框图示于图 1-1 中。

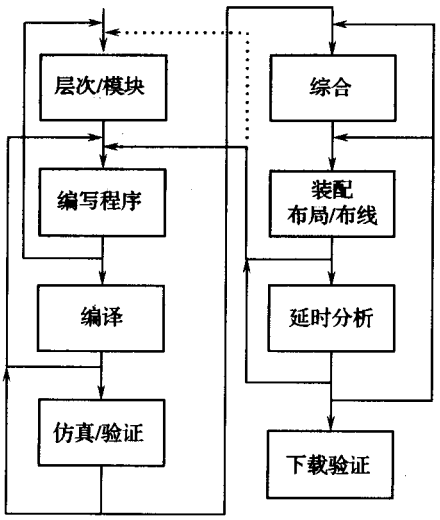


图 1-1　VHDL 设计流程

整体设计流程分为“前期”工程和“后期”工程两步。

所谓“前期”步骤又分为 3 个步骤，开始于提出基本方法和建立在框图层次上的模块。大型的逻辑设计通常是分层次的。VHDL 有好用的框架来定义模块及其接口，还有随后填加的实体细节和它们的内部结构体细节。

第 2 步是真正为模块编写 VHDL 程序，包括接口、内部细节。使用专业的 VHDL 文本编辑器使这一步工作变得更容易。这样的编辑器有自动高亮、VHDL 关键字、自动缩进等，对常用的程序结构内部模块的语法进行检查，单击进入编译等功能。

数字系统的设计者写出一些程序就可以编译。VHDL 编译器为了检查语法错误并检查与其他相关模块的相容性，它也创建内部信息，这是后来设计进程中的仿真所需要的。没必要把所有的程序编完了才进行编译，尤其是工程较大时，一次编译一个模块可以避免产生语法错误、名称不一致等问题。

第 3 步是仿真/验证。在没有安装具体电路的情况下，VHDL 仿真器对设计进行仿真，并观察其输出波形。除了能观察到输出波形外，仿真/验证更重要的作用是分析电路是否能按期望的那样工作。在大的工程中，大量的努力都花在这一步中，在这一阶段发现设计错误具有很高的价值，如果错误发现得迟了，可能“后期”步骤都要返工。

在 VHDL 的函数声明里，研究电路和逻辑操作时不考虑延时，即认为门的延时参量是零。而在仿真的延时验证中，研究包括估计延迟时间的电路的操作，并检验上升时间、保持时间和其他延时是否满足要求。由于延时可能会过于依赖综合和适配的结果，前期工作

的延时检验是有限的。可以做一些初步的延时检验获得适合总体设计的方法，但是延时检验的细节必须到最后才能得到。

仿真/验证之后，进行“后期”阶段的工作。“后期”阶段分为 3 个步骤：逻辑综合、装配与布局/布线、延时分析。

在逻辑综合阶段，综合器所要做的工作是检查 VHDL 程序的语法是否正确，再根据厂商提供的器件库，将 VHDL 源程序转换成各种器件的组合，并依据设计者所给出的命令，在各器件之间做适当的布线。综合时的几个要点：VHDL 源程序、厂商提供的库以及用户所执行的命令，称之为约束条件，如面积、速度、功耗、可测性；支持工艺库，如 TTL 库、CMOS 库等；最后一步是进行延时分析，延时分析主要是将做完布局/布线的结果再做一次验证，如验证前级输出信号到本级信号的建立时间及保持时间是否足够、延时限制条件是否满足等。如果时序上有错误就要寻找问题的根源，返回设计中的某个步骤改写设计，这些都是要做延时分析时才能知道的。“前期”和“后期”工作都完成后，就可以对芯片下载编程了。

第 2 章　VHDL 的程序结构

一个完整的 VHDL 程序通常包括实体(Entity)、结构体(Architecture)、配置(Configuration)、程序包集合(Package)和库(Library)5 个部分。前 4 部分是可分别编译的源设计单元。库存放已经编译的实体、结构体、配置和程序包集合。

2.1　VHDL 程序的基本结构

VHDL 的程序结构至少由实体(entity)和结构体(architecture)两部分组成。借用 Pascal 和 Ada 软件编程语言的编程思想，VHDL 也采用结构程序的设计原理，其主要思想是在实体中定义一个芯片(IC)或硬件电路模块的接口，同时隐去它的内部细节。因此，实体是 VHDL 的硬件抽象,它表示具有明确的输入、输出的硬件设计的一部分。同时，结构体指定设计实体输入和输出之间的行为、逻辑关系或功能，并且可以采用行为风格、数据流风格、结构化风格或 3 种风格的混合形式进行描述。图 2-1(a)为 VHDL 程序的基本构成,图 2-1(b)是一个半加器的逻辑图，例 2-1 是用 VHDL 设计的半加器的程序。总之，实体和结构体是一个 VHDL 文件的最基本的部分。

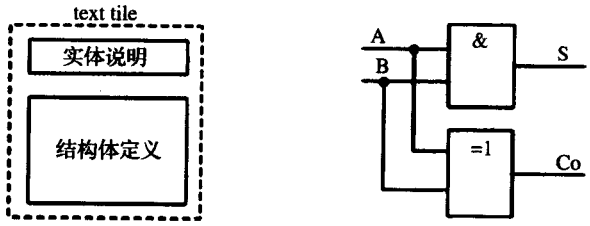


图 2-1　VHDL 程序的基本构成及举例

(a) VHDL 程序的基本构成；　(b) 半加器。

例 2-1　半加器的 VHDL 程序

```
entity adder1 is
    port(
            a:in bit;
            b:in bit;
            s:out bit;
            co:out bit
        );
end adder1;                                    --以上是半加器的实体说明

architecture adder1_arch of adder1 is
```

```
begin
    s<=a xor b;
    co<=a and b;
end adder1_arch;                          --以上是半加器的结构体定义
```

VHDL 的实体说明是结构体的接口，可以被其他设计模块调用，从较高层次隐去低层次结构体的细节。如图 2-2 所示，高层次结构体可以多次使用低层实体，而且多个顶层结构体可以使用同一个低层实体。图 2-2 中，结构体 B、E 和 F 不用其他任何实体，是单独的。

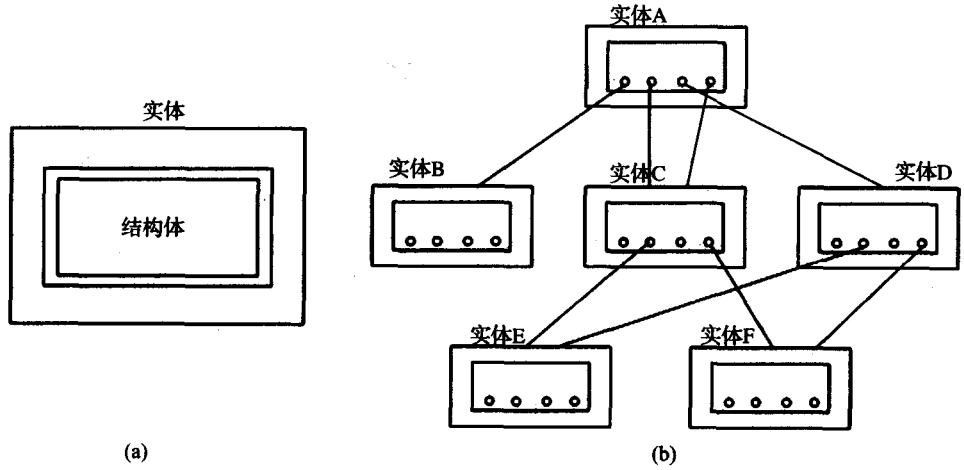


图 2-2 VHDL 程序的基本结构和层次关系

(a) 一个 VHDL 程序的基本结构；(b) 多个 VHDL 程序的层次关系。

VHDL 允许设计者对单个实体定义多个结构体，并提供一个配置管理器，负责管理在特定的编译和仿真期间使用某个结构体，并对仿真的结果进行比较，从中可以选出最佳的结构体，如图 2-3 所示。

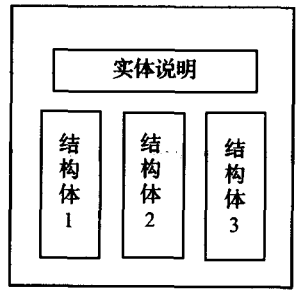


图 2-3 有多个结构体的 VHDL 程序

在 VHDL 的文本程序中，结构体定义紧跟在实体说明之后。例如，例 2-2 是一个简单的二输入与门的 VHDL 程序，实现 z = x and(not y)的逻辑功能，即当 x=1 且 y=0 时，z=1，否则 z=0。其中，inhibit 是实体名，inhabit_arch 是结构体名，都是由设计者来命名的标志符。

例 2-2　一个与门电路的 VHDL 程序

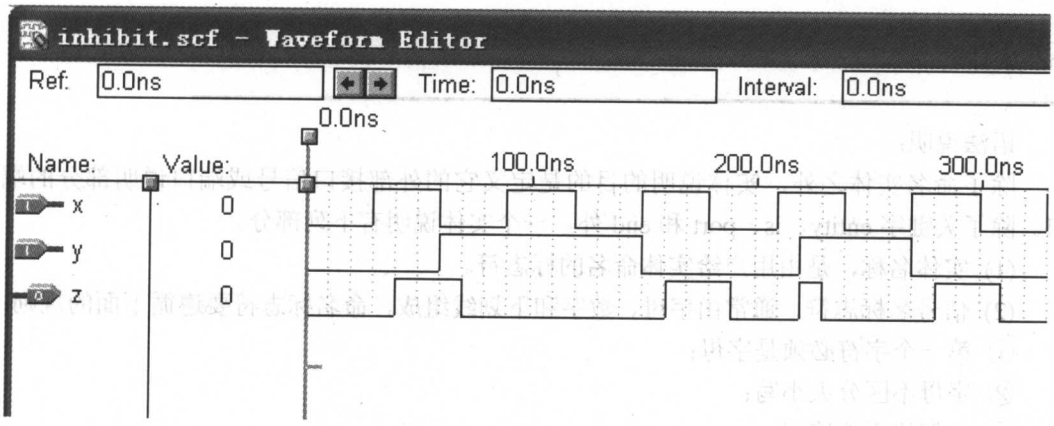
```
library ieee;
use ieee.std_logic_1164.all;                    --打开需要用到的库

entity inhibit is                               --实体(端口)说明
    port(
            x, y:   in BIT;
            z:      out BIT
        );
end inhibit;
architecture inhibit_arch of inhibit is         --结构体定义
begin
    z <= '1' when x='1' and y='0'else
         '0';
end inhibit_arch;
```

程序说明：

由仿真结果可以得出，只有当 x=1，y=0 时，输出 z=1，其余 z=0。在结构体描述中使用了 when-else 语句(参见 3.4 节)。

仿真结果：



在大的工程里，实体和结构体有时定义在独立的文件里，其编译按照它们声明的名字去匹配。

像其他高级编程语言一样，VHDL 也要求程序易读，如用两个横杠“--”表示注释语句的开始、起名字时最好与其功能有关等。

VHDL 定义了很多特殊的字符串，叫做保留字(reserved words)或关键字(key words)。举的例子中包括：entity，port，is，in，out，end，architecture，begin，when，else 和 not。举例中的命名标志符是 Inhibit、x、y、BIT、z 和 inhibit_arch。“BIT”对应一个预定义数据类型(参见 3.1 节)，是一个内在标志符。在 VHDL 中，程序的注释是使用两个连续的横杠“--”，不论是在程序中的任何地方，只要出现连续两个横杠，其后面的语句都会被编译器

(compiler)忽略。注释语句可以放在程序的任何地方。

2.1.1 实体

实体说明是 VHDL 语言的硬件抽象。它表示具有明确的输入、输出的硬件设计的一部分端口。实体定义必须和其所对应的结构体定义存放在同一个库中，一个实体代表整个电路设计的一个完整层级，可以是整个电路，也可以是任何层级的一个模块。

实体说明不但定义设计实体和使用设计实体的环境之间的接口(或端口)，还指定作为设计实体一部分的说明语句。实体说明可以由多个设计实体共享，而每一个设计实体可以具有不同的结构体。因此，实体说明可以潜在地表示具有相同端口的一类设计实体。

VHDL 实体说明的语法格式为

```
entity    实体名    is
    [generic(类属说明)      --定义端口的大小、I/O 引脚的分配
                            --实体中子元件的数目及实体的定时特性]
    port ( 信号名：模式   信号类型;
           信号名：模式   信号类型;
            …
           信号名：模式   信号类型
           );
end 实体名;
```

语法说明：

除了命名实体之外，实体说明的目的是定义它的外部接口信号或端口说明部分的端口。除了关键字 entity、is、port 和 end 外，一个实体说明有下列部分。

(1) 实体名称，是由用户给实体命名的标志符。

(2) 信号名标志符，通常由字母、数字和下划线组成，命名标志符要遵循下面的规则：

① 第一个字符必须是字母；

② 字母不区分大小写；

③ 下划线不能连用；

④ 最后一个字符不能是下划线。

信号名是外部端口(或接口)信号的标志符。信号分为 5 种模式，详细说明信号的方向，这 5 种模式为

in	输入实体的信号。如时钟(clk)、复位(reset)、使能信号、地址输入等信号。
out	输出实体的信号。如计数输出、寄存器输出、控制其他单元的信号等。注意：这类信号不能被实体内部的结构体引用，只能被其他实体使用。
buffer	输出实体的信号，是可读的，即可以被实体内部的结构体引用，形成内部反馈。
inout	输入或输出实体的信号。这种模式经常用于可编程逻辑器件 PLD 中的三态