Mc Graw Hill Education

VHDL: Programming By Example FOURTH EDITION

VHDL 编程实例

（第四版）

■ [美] Douglas L. Perry 原著
■ 杨承恩 谭克俊 颜德文 译

http://www.phei.com.cn

本书含光盘1张

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

VHDL 编程实例

（第四版）

VHDL: Programming by Example

（Fourth Edition）

[美] Douglas L. Perry 著

杨承恩 谭克俊 颜德文 译

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书从实用的角度出发，用讲解实例的方法，由浅入深地向读者依次介绍了 VHDL 的基本概念、建模的过程、预定义属性和配置等基本内容，并详细地介绍了设计描述、逻辑综合、RTL 仿真、布局布线、VITAL 仿真，以及系统硬件调试这样一个完整的 VHDL 设计过程。这样做目的是希望读者在完成本书的各个例题后，基本能够掌握基于 VHDL 的数字系统设计方法，使其数字系统的设计能力上升到一个新的水平。

本书适合作为高等院校电子及计算机类专业，或相关专业高年级学生或研究生的教材，也可作为工程技术人员的参考用书。

版权贸易合同登记号 图字：01-2007-0720

图书在版编目（CIP）数据

VHDL 编程实例：第 4 版 /（美）佩里著；杨承恩，谭克俊，颜德文译. —北京：电子工业出版社，2009.6
书名原文：VHDL: Programming by Example（Fourth Edition）
ISBN 978-7-121-08725-7

Ⅰ. V… Ⅱ. ①佩… ②杨… ③谭… ④颜… 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（2009）第 065290 号

责任编辑：柴 燕（chaiy@phei.com.cn）
印 刷：北京天宇星印刷厂
装 订：涿州市桃园装订有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1 092 1/16 印张：22.75 字数：582.4 千字
印 次：2009 年 6 月第 1 次印刷
印 数：4 000 册 定价：58.00 元（含光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

原著序

自从 1987 年 VHDL 被 IEEE 作为标准颁布以来，它已经成为电子设计产业的核心语言。近 15 年来，VHDL 已经从设计文档的初始理念，发展成为具有完成设计和验证功能的工具，使得电子设计自动化工业得以迅猛扩展。可以说，VHDL 推进了现代综合技术，使 ASIC 半导体公司得以发展。对于在世界范围内的 VHDL 使用者来说，从本书的第四版中可以获得权威的实际应用信息。

随着半导体器件的体积逐渐缩小，VHDL 的重要性日渐突出，并得到了广泛的应用。在不到 10 年前，用 VHDL 和原理图描述，进行混合设计还是件很普通的事情。但是，随着设计复杂性的增加，工业界抛弃了原理图设计方法而单独垂青于硬件描述语言。本书版本的不断更新就是为了跟上产业发展对 VHDL 需求的步伐。

VHDL 的适应能力在于它的结构的贡献。对 VHDL 的程序包结构的应用，使得设计人员、电子设计自动化公司和半导体厂家能够使用新的语言概念来进行试验，以确保良好的设计工具和数据的互用性。当联合数据类型在 IEEE1164 标准中被认可后，就意味着设计数据的互用性是可能的。

所有这些都是由工业界来推动的，他们依托一个由系统公司、电子设计自动化公司和半导体公司所组成的称为 Accellera 的联盟。

当 ASIC 工业需要一个标准方式来转换 VHDL 中的门级设计数据和时序信息时，Accellera 的前身（VHDL 国际）的发起人之一，IEEE VHDL 团队，建立了一个协作标准，由此提出了 IEEE 1076.4 VITAL（VHDL Initiative Towards ASIC Libraries），并被批准作为提供给设计人员一个单一的由概念到门级流化的语言流程。

20 世纪 90 年代末，Verilog HDL 和 VHDL 工业标准团队在诸如 IEEE 1497 SDF 的公共时序数据、设立寄存器传输级（RTL）标准等方面进行合作，并且更进一步改进硬件描述语言的设计方法和外部连接手段。

但是从一开始，VHDL 团队的领导层就确保对电子设计工程界来说，这是一个开放的和被国际社会所认可的标准。这支队伍的工作成果成为这个领域的基准，因为它的测量开放性使设计师们持续受益，直到今天仍然如此。

如同电子设计自动化界不断地从 VHDL 设计描述中发现可用于工作的新算法以及相关的标准，来再一次推动设计人员的生产积极性一样，设计界也看到了同样的好处。并且作为新一代的可编程逻辑器件的设计人员，把对硬件描述语言的使用作为他们的设计方法学的基础，这样使得 VHDL 的用户数量得到稳步增长。

新一代的电子设计人员和当代的复杂系统的设计人员与 ASIC 一起，将会同意第一代的 VHDL 用户们对本书第一版的观点，发现本书是无价的。目前对标准使用的更新，将使得人们在今后若干年中从成功地运用 VHDL 语言进行电子设计中获益。

Dennis B. Brophy
Accellera主席

致 谢

如果没有许多人的帮助，本书不可能完成，在此我愿意向他们表达我的感激之情。Rod Farrow, Cary Ussery, Alec Stanculescu 和 Ken Scott 回答了我大量的关于 VHDL 的奇怪问题。Mark Beardslee 和 Derek Palmer 评阅了本书的第三版的部分章节。他们的评论具有深刻的见解，极有帮助。Paul Krol 提供了第 7 章中关于类属的表格。Keith Irwin 帮助确定了某些章节的风格。Hoa Dinh 和 David Emrich 回答了许多关于 FPGA 综合的问题。感谢 John Ott 和 Dennis Brophy 提供了在写作中使用的和在 CD 中的 ModelSim 和 Leonardo Spectrum 软件。感谢 Altera 的 Derek Palmer 和 Robert Blake 提供了 MaxPlus II 软件并回答了一些问题。最后感谢 Endric Schubert、Mark Beardslee、Gernot Koch、Olaf Poeppe、Matt Hall、Michael Eitelwein、Ewald Detjens 和 William Vancleemput，感谢他们在与 Bridges2Silicon 公司合作期间的努力工作。

前　言

这是本书的第四版，现在这个版本不仅提供了 VHDL 语言体系，而且还提供了设计方法学的知识。这一版将通过创建一个 VHDL 设计，按照对设计进行仿真、综合，设计布局与布线，使用 VITAL 仿真来验证最终结果这么一个步骤来指导读者。快速（At-Speed）调试新技术可以提供极其快捷的设计验证。在这一版中的设计实例已经被更新以及时反映设计方法学的最新关注点。

本书旨在帮助硬件设计工程师学习怎样编写一个完善的 VHDL 设计描述。其目标是提供足够的 VHDL 和设计方法学的知识，使得设计人员能够快速地完成一个优良的 VHDL 设计，并且能够验证这个设计。本书也将试图使一个对 VHDL 完全不了解或仅了解一点点的设计人员通过阅读本书达到写出复杂的 VHDL 描述的水平。本书并不打算给出在每一个可能的应用领域中的每一个可能的 VHDL 结构，而是让设计人员了解怎样简单、高效、正确地写出硬件设计的 VHDL 描述。

本书在内容编排上分为三个逻辑部分。第一部分介绍了 VHDL 语言；第二部分贯穿了 VHDL 的基本设计步骤，其中包括仿真、综合、布局与布线，以及 VITAL 仿真；第三部分贯穿介绍了一个小型 CPU 设计的例子，包括从 VHDL 的获取直到最终的门级实现，以及快速调试。在本书的最后附有附录，介绍了许多关于这门语言的有用的知识和本书所使用的例子。

在第一部分中不止一次地介绍了 VHDL 的特性。在介绍每一个特性时，都引入了一个或多个实例来表明怎样应用这些特性。第一部分由第 1～8 章组成，在每一章中都介绍了 VHDL 的一些描述性能。第 1 章讨论了 VHDL 设计是怎样与原理图设计相关联的，并且介绍了这门语言的基本术语。第 2 章描述了 VHDL 的一些基本概念，包括不同的延时机制、怎样使用例化的特殊数据和定义 VHDL 驱动器。第 2 章讨论了并行语句，而在第 3 章中向读者介绍了顺序语句。在第 4 章中谈到了在 VHDL 中类型使用的有效范围，对每一个类型都给出了一些例子，用以表明在实际的例子中如何使用它们。在第 5 章中引入了子程序和程序包的概念，给出了函数的不同用法，以及在 VHDL 程序包中可以引用的特性。

第 6 章中介绍了 VHDL 的 5 种属性，每种属性都通过举例说明怎样使用这些特定的属性来使设计人员得到最大的便利。通过所给出的例子说明了每一个属性的作用。

第 7 章和第 8 章向读者介绍了一些更加高级的 VHDL 特性。第 7 章讨论了怎样使用 VHDL 的配置来构造和管理复杂的 VHDL 设计，对每一个不同配置方式与表明它们的用法的例子一起进行了讨论。第 8 章介绍了更多的关于讨论重载、用户定义属性、生成语句和 TextIO 的 VHDL 高级主题。

本书的第二部分包含了第 9～11 章。第 9 章和第 10 章讨论了综合的方法，怎样编写可综合的设计。这两章描述了综合过程的基础，其中包含怎样编写一个可综合的 VHDL，什么是工艺库，综合的过程是怎样的，什么是它的约束和属性以及最优化的过程是怎样进

行的。第 11 章讨论了从 VHDL 获取直到 VITAL 仿真的完整的高水平设计流程。

本书的第三部分贯穿介绍了一个小型 CPU 设计的全过程，从 VHDL 的编程到仿真、综合，布局与布线以及 VITAL 仿真。第 12 章从逻辑功能的角度描述了 CPU 的顶层设计。在第 13 章中，展现了 CPU 的 RTL 描述并且从综合的角度进行了讨论。第 14 章讨论 VHDL 的测试平台和怎样使用这些平台来进行功能验证，在叙述了 CPU 设计的仿真后结束了第 14 章的讨论。在第 15 章中，验证过的设计被综合成为目标工艺。第 16 章得到综合后的设计，并且对这个设计进行布局与布线，使之成为目标器件。第 17 章以对 VITAL 的讨论开始，以对完成了布局和布线的 CPU 设计进行 VITAL 仿真而结束。第 18 章是一个新增的章节，讨论了快速调试新技术，这一章深入细致地告诉读者这个 CPU 设计的硬件实现怎样能够帮助加速验证。

最后，在本书的结尾有 4 个附录提供了参考信息。附录 A 是贯穿本书所使用的一个 IEEE 1164 STD_LOGIC 程序包的程序清单。附录 B 是一组非常有用的表格，它把本书中其余的有用信息汇总成几个快速参考表。最终，附录 C 告诉我们怎样阅读在 VHDL 语言参考手册（VHDL Language Reference Manual）中使用的 Bachus-Naur 描述模式（BNF）。附录 D 中列举了 VHDL93 更新的内容。我谨希望读者能和我在写作本书时获得乐趣一样，从本书的阅读中和在使用 VHDL 的工作中获得许多乐趣。

译者序

非常高兴能受到电子工业出版社的邀请，将 Douglas L. Perry 先生的《VHDL Programming by Example (Fourth Edition)》一书介绍给广大的中国读者。作为一种通用的硬件描述语言，VHDL 目前已被广泛地应用在数字系统的设计当中。Douglas 先生的《VHDL Programming by Example》一经问世，就在 VHDL 的读者群中产生了巨大的影响。此书写作风格独特，描述方式与一般的计算机语言类图书不同。作者不是依惯例由语句、语法、编程、举例这种逐步描述方式，而是从实用的角度出发，用实例来说明问题。本书通过大量简繁难易程度不同的实例，由浅入深地向读者依次介绍了 VHDL 的基本概念，建模过程以及预定义属性和配置等基本内容，然后介绍了 VHDL 的高级特性以及它的综合工具。在书中的第三部分，作者以一个 CPU 模块的设计为例，详细介绍了设计描述、逻辑综合、RTL 仿真、布局布线、VITAL 仿真以及在系统硬件调试的 VHDL 设计全过程。这也正是本书的独到之处。如果读者能依本书的例子完成 CPU 的设计，就可以基本掌握基于 VHDL 的数字系统设计方法。

由于本书的这种别具一格的写作特点，读者在阅读本书时消除了通常阅读计算机语言类图书时所产生的乏味现象，因而本书深受读者的欢迎，应读者的要求现已出版到第四版。在这一版中，作者依据调试技术的发展增加了关于快速调试新技术的章节，使得本书能够跟上技术发展的潮流，让读者及时掌握科技发展的最新动态。

本书的序、前言以及第 1～5 章由杨承恩翻译，第 6～12 章由颜德文翻译，第 13～18 章及附录由谭克俊翻译，最后由杨承恩进行了统稿。需要说明的是，原书的章节除了章有序号外，所有的节与小节均无序号，在统稿时，考虑到国人的阅读习惯，译者依个人理解，为节与小节添加了序号，并在个别地方做了细微的调整。

由于译者水平有限，译文中定有许多错误与不妥之处，欢迎读者批评指正。

译　者

2009 年于大连

目　录

第1章 VHDL介绍

VHSIC 硬件描述语言（VHDL）是一门工业标准语言，用来描述从抽象层次直到具体层次的硬件。VHDL 起源于在 20 世纪 70 年代和 80 年代初美国国防部的工作，它是由 ADA 语言发展起来的，至今都可以在 VHDL 的全部结构以及其他的 VHDL 的语句中看到它的影子。

VHDL 的用户自它诞生以来迅速增加，目前在全球已有成千上万的工程师用它来创造复杂的电子产品。本章将以由浅入深的方式把读者带进复杂的 VHDL 世界。VHDL 是一门强有力的语言，具有若干个语言构造，用这些构造可以描述非常复杂的行为。了解 VHDL 的所有特性不是一项简单的任务，本章首先以一个简单的方式介绍它的复杂特性，然后再描述它的更加复杂的用法。

1986 年，VHDL 被提议作为 IEEE 的一个标准。在经历了若干次的更改和修订之后，终于在 1987 年 12 月被接纳作为 IEEE 1076 标准。IEEE 1076-1987 标准 VHDL 是本书所使用的 VHDL。（附录 D 中简短地描述了 VHDL 1076-1993）。所有的例子都是用 IEEE 1076 VHDL 描述的，用 Model Technology 公司的 VHDL 仿真环境进行了编译和仿真。综合的例子是用 Exemplar Logic 公司的综合工具完成的。

1.1 VHDL 术语

在进一步阅读本书之前，让我们先定义一些贯穿全书所使用的术语。这些是构筑 VHDL 的最基本的构件，出现在几乎所有的描述中，它们和某些在 VHDL 中重新定义的术语一样与通常设计人员所了解的含义是有差别。

- 实体（Entity）。所有的设计都是用实体来表达的，实体是设计的最基本模块。设计的最高层次是顶层实体。如果设计是分等级的，那么顶层的描述将包含较低层次的描述。这些较低层次的描述将是被包含在顶层实体描述中的较低层次的实体。
- 结构体（Architecture）。所有的能被仿真的实体都有一个结构体描述，这个结构体描述了实体的行为。一个单一的实体可以有多个结构体，一个结构体可能是一种行为，而另一个结构体可能是一个设计的结构描述。
- 配置（Configuration）。配置语句用来把一个元件实例绑定到实体—结构体对上。一个配置可以认为是设计的元件列表，它描述了某种行为适用于哪个实体，极其像一个元件列表的描述，这描述告诉我们哪一部分适用于设计中的哪一个部分。
- 程序包（Package）。程序包是一个在设计中常用的数据类型和子程序的集合。可以把程序包看做是一个含有各种用于构建设计的工具的工具箱。
- 驱动器（Driver）。这是一个信号源，如果一个信号由两个源驱动，那么当这两个源都起作用时，这个信号就有两个驱动器。

- 总线（Bus）。总线这个术语带给我们的印象通常都是一组信号或者是一种在硬件设计中使用的独特的通信方式。在 VHDL 中，总线是一种特殊类型的信号，这种信号可以使自身的驱动器关断。
- 属性（Attribute）。属性是一类附着在 VHDL 对象上的数据，或者预定义的关于 VHDL 对象的数据，如当前缓冲器的驱动能力或器件的最高运行温度等。
- 类属（Generic）。类属是 VHDL 的一个参数的术语，这个参数把信息传递到一个实体。例如，如果一个实体是一个具有上升和下降延时的门级模型，用类属就可以把上升和下降延时的值传递到实体中。
- 进程（Process）。进程是 VHDL 中执行的基本单元。所有的在对 VHDL 描述进行仿真中所执行的操作都可被分解为单进程或多进程。

1.2 在 VHDL 中描述硬件

VHDL 描述由初步设计单元和二次设计单元组成。初步设计单元指的是实体和程序包，二次设计单元指的是结构体和程序包体。二次设计单元总是与初步设计单元相互关联的。库中收集了许多初步设计和二次设计单元，一个典型的设计通常都含有一个或多个设计单元库。

1.3 Entity 实体

一个 VHDL 实体指定了实体的名称、实体的端口和实体所关联的信息。所有的设计都是用一个或多个实体创建起来的。

下面看一个简单的实体的例子。

```
ENTITY mux IS
   PORT ( a, b, c, d : IN BIT;
          s0, s1 : IN BIT;
          x, : OUT BIT);
END mux;
```

关键字 ENTITY 表示这是一个实体语句的开始。在本书的所有描述中，语言的关键字和由标准程序包提供的类型都用大写字母表示。例如，在前述的例子里，关键字是 ENTITY、IS、PORT、IN、INOUT 等，提供的标准类型是 BIT。用户创建的对象名称，如上面的例子中的 mux，将用小写字母表示。

实体的名称是 mux。在 PORT 子句中表明这个实体有 7 个端口。6 个端口是 IN 模式，1 个端口是 OUT 模式。4 个数据输入端口（a、b、c、d）是 BIT 类型，两个多路选择器选择输入端口 s0 和 s1 也是 BIT 类型，输出端口同样是 BIT 类型。

实体描述了面向外部世界的接口，它指定了端口的数量、端口的方向和端口的类型。能放置在实体中的信息比在这里表明的要多的多，这里仅提供给我们一个基础，在这个基础上我们能建立更复杂的例子。

1.3.1　结构体

实体描述 VHDL 模型的接口。结构体描述实体的基本功能，并且含有对实体的行为进行建模的语句。一个结构体总是与一个实体和描述这个实体的行为相关联的。上面所描述的计数器件的结构体如下。

```
ARCHITECTURE dataflow OF mux IS
  SIGNAL select : INTEGER;
BEGIN
  select <= 0 WHEN s0 = '0' AND s1 = '0' ELSE
            1 WHEN s0 = '1' AND s1 = '0' ELSE
            2 WHEN s0 = '0' AND s1 = '1' ELSE
            3;
x <= a AFTER 0.5 NS WHEN select = 0 ELSE
     b AFTER 0.5 NS WHEN select = 1 ELSE
     c AFTER 0.5 NS WHEN select = 2 ELSE
     d AFTER 0.5 NS;

END dataflow;
```

关键字 ARCHITECTURE 表示这条语句描述了一个实体的结构体。这个结构体的名称是 dataflow，所描述的实体称为 mux。

在进行结构体和实体之间的连接时，应考虑到一个实体可以有多个描述实体行为的结构体。例如，一个结构体可以是行为描述，而另一个结构体可以是构造描述。

关键字 ARCHITECTURE 和关键字 BEGIN 之间的文字区域是为稍后使用的局部信号和元件声明区域。在这个例子中，声明了信号 select 作为局部信号。

结构体的语句区域由关键字 BEGIN 开始。所有的在 BEGIN 和 END 之间的网表语句被称为并行语句，因为所有的语句在同一时刻被执行。

1.3.2　并行信号赋值

在典型的程序设计语言如 C 或 C++中，每条赋值语句的执行都是以一个指定的次序一条接一条进行的。这个执行的次序是由源文件中语句的次序来确定的。在 VHDL 的结构体中，不存在指定赋值语句的次序，其执行的次序是依据所发生的关于某些信号的事件来唯一指定的。赋值语句对这些信号必须是敏感的。

考察在结构体 behave 之后的第一条赋值语句，如下所示。

```
select <= 0 WHEN s0 = '0' AND s1 = '0' ELSE
          1 WHEN s0 = '1' AND s1 = '0' ELSE
          2 WHEN s0 = '0' AND s1 = '1' ELSE
          3;
```

信号的赋值是用符号<=来识别的。信号 select 将根据 s0 和 s1 的值被赋予不同的数值。无论何时，只要有关于信号 s0 或信号 s1 的事件发生，这条语句就被执行。关于一个信号

的事件是指这个信号的值发生了变化。信号赋值语句对在符号<=右边的任何信号的变化都是敏感的。这条信号赋值语句是对 s0 和 s1 敏感的，dataflow 结构体中其他的信号赋值语句是对信号选择敏感的。

现在来看看这些语句实际上是怎样工作的。假设有一个稳态条件，也就是 s0 和 s1 两者的值都为 0，并且信号 a、b、c 和 d 当前的值也都为 0，信号 x 将会有一个 0 值，因为只要信号 s0 和 s1 两者的值都为 0，它就会被赋予信号 a 的值。现在，假设我们触发了一个关于信号 a 的事件，将它的值改为 1。当这个事件发生，这第一条信号赋值语句将不会执行，因为这条语句对信号 a 的变化不敏感，这是因为信号 a 不在运算符的右边。第二条信号赋值语句会执行，因为它对关于信号 a 的事件是敏感的。当第二条信号赋值语句执行后，a 的新值被赋给信号 x。输出端口 x 现在的值变为 1。

现在再看看当信号 s0 的值变化时的情况。假设 s0 和 s1 两者都为 0，并且端口 a、b、c 和 d 的值分别为 0、1、0 和 1。现在让信号 s0 的值由 0 变到 1。第一条信号赋值语句对信号 s0 是敏感的，因而会执行。在并行语句执行时，将会使用所有包含在表达式中的信号的当前值来计算表达式值。

当第一条语句执行时，将用符号“<=”右边的信号表达式的当前值来计算应赋给 q 的新值。表达式值的计算会使用表达式中的所有信号的当前值。

在 s0 的值等于 1 且 s1 的值等于 0 时，信号 select 将接收到一个新值 1。这个 select 信号的新值将触发一个针对 select 信号的事件，导致第二条信号赋值语句执行。这条语句利用信号 select 的新值来把端口 b 的值赋给端口 x。这新的赋值将使得端口 x 由 0 变到 1。

1.3.3 事件安排

信号 x 的赋值不会瞬即产生，每一个要赋给信号 x 的值都含在 AFTER 子句中。推迟这个新值生效的机制称为安排一个事件。在这个赋给端口 x 一个新值的语句中，安排在 0.5ns 后发生一个事件，也就是信号 x 含有一个新的值。当这个事件的条件成熟了（0.5ns 后），信号 x 就会收到新值。

1.3.4 语句并行性

第一项赋值是在端口 s0 或 s1 发生事件时执行的唯一一条语句。第二条信号赋值语句不会被执行，除非一个关于信号 select 的事件发生或者是一个关于端口 a、b、c、d 的事件发生。

这两条在结构体 behave 中的信号赋值语句形成一个实体 mux 的行为模型或结构体。dataflow 结构体不含有结构，在这个结构体中没有给出任何元件例化。由于没有更深的层级，这个结构体可以认为是一个在设计的层级中的末节点。

1.3.5 结构设计

编写 mux 设计的另一个方法是例化子元件，子元件用来执行复杂模型中较小的运算。用一个尽可能简单的，我们前面曾经用过的 4 输入多路选择器模型，可以产生一个简

单的门级描述来表明怎样描述和例化元件。下面给出的结构体是对 mux 实体的一个结构描述。

```
ARCHITECTURE netlist OF mux IS
    COMPONENT andgate
        PORT(a, b, c : IN bit; x : OUT BIT);
    END COMPONENT;
    COMPONENT inverter
        PORT(in1 : IN BIT; x : OUT BIT);
    END COMPONENT;
    COMPONENT orgate
        PORT(a, b, c, d : IN bit; x : OUT BIT);
    END COMPONENT;

SIGNAL s0_inv, s1_inv, x1, x2, x3, x4 : BIT;

BEGIN
    U1 : inverter(s0, s0_inv);
    U2 : inverter(s1, s1_inv);
    U3 : andgate(a, s0_inv, s1_inv, x1);
    U4 : andgate(b, s0, s1_inv, x2);
    U5 : andgate(c, s0_inv, s1, x3);
    U6 : andgate(d, s0, s1, x4);
    U7 : orgate(x2 => b, x1 => a, x4 => d, x3 => c, x => x);
END netlist;
```

这个描述使用了许多低层元件来建立 mux 器件的行为模型，其中有一个反相器 inverter 元件，一个与门 andgate 元件和一个或门 orgate 元件。所有这些元件都在结构体的声明区中声明。声明区位于结构体语句和关键字 BEGIN 之间。

例子里用了许多局部信号来连接每一个元件以形成结构体描述，这些局部信号是用 SIGNAL 声明来声明的。

结构体的语句区位于关键字 BEGIN 之后。在这个例子中有许多元件例化语句。这些语句分别标有 U1-U7。语句 U1 是一条元件例化语句，它例化了反相器元件。这条语句把端口 s0 连接到反相器元件的第一个端口，把信号 s0_inv 连接到反相器的第二个端口。其结果是反相器的端口 in1 被连接到 mux 实体的端口 s0，反相器的端口 x 被连接到本地信号 s0_inv。在这段语句中，端口根据它们在语句中出现的顺序依次连接。

请注意元件例化语句 U7。这条语句使用了下列标识：

```
U7 : orgate(x2 => b, x1 => a, x4 => d, x3 => c, x => x);
```

这条语句用名称关联来指定端口与信号之间的相互配对。例如，语句的第一段表明或门 orgate 的端口 x2 连接到实体的 b 端口。最后一个例化子句把 orgate 元件的端口 x 连接到实体的 x 端口。子句的次序不重要。连接的名称和次序可以混杂，但是不推荐这样做。