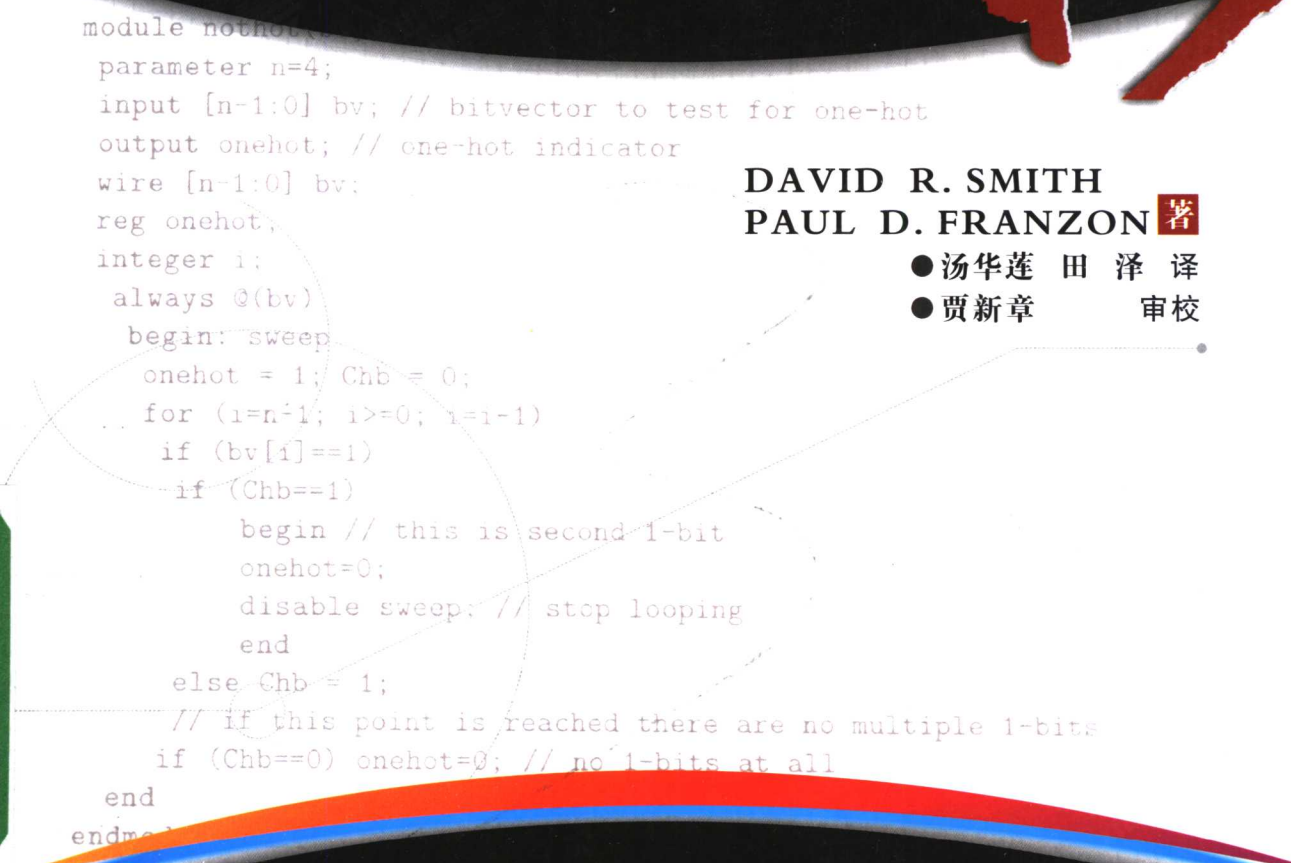
PEARSON
Prentice
Hall

Verilog Styles for
Synthesis of
Digital Systems

面向数字系统综合的Verilog编码风格



DAVID R. SMITH
PAUL D. FRANZON 著

●汤华莲 田 泽 译
●贾新章 审校

西安电子科技大学出版社
http://www.xduph.com

PEARSON Education

面向数字系统综合的 Verilog 编码风格

Verilog Styles for Synthesis of Digital Systems

DAVID R. SMITH　PAUL D. FRANZON　著

汤华莲　田泽　译

贾新章　审校

西安电子科技大学出版社

2007

图书在版编目(CIP)数据

面向数字系统综合的 Verilog 编码风格/(美)史密斯(Smith, D.R.)，(美)弗兰曾(Franzon, P.D.)著；
汤华莲，田泽译. —西安：西安电子科技大学出版社，2007.11
书名原文：Verilog Styles for Synthesis of Digital Systems
ISBN 978-7-5606-1870-8
Ⅰ. 面… Ⅱ. ① 史… ② 弗… ③ 汤… ④ 田…
Ⅲ. 硬件描述语言，Verilog HDL—程序设计 Ⅳ. TP312
中国版本图书馆 CIP 数据核字(2007)第 166578 号

策　　划　臧延新
责任编辑　阎　彬　臧延新
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　　邮　编　710071
http: //www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com
经　　销　新华书店
印刷单位　陕西天意印务有限责任公司
版　　次　2007 年 11 月第 1 版　2007 年 11 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印　张 17.5
字　　数　405 千字
印　　数　1～4000 册
定　　价　30.00 元
ISBN 978-7-5606-1870-8/TP・0973
XDUP 2162001-1

内 容 简 介

Verilog HDL 是当今国际上一种主流的标准化硬件描述语言，目前已出版有多本详细介绍该语言语法和结构的教材。本书的不同之处在于其重点介绍的并不是语法本身，而是以电路综合为目标，通过大量实例来说明具有不同特点的可综合的编码风格。全书共分 17 章，覆盖了 Verilog 基本语法、仿真测试、面向 FPGA 和标准单元的逻辑综合、可综合的代码风格和 VLSI 设计方法学等关键内容，最后还简要描述了混合技术的设计。

本书是一本实用性很强的针对 Verilog HDL 综合的教材，适用于计算机和电子类相关专业的高年级本科生和研究生，同时也可作为从事数字电路设计人员的参考书。

版权贸易合同登记号 陕版出图字：25—2007—045

前　言

近几年，数字电路的硬件设计正在经历着一场变革。一方面，使用电路图输入工具的手工设计已经被高层次描述的综合所代替；另一方面，在许多应用中，定制集成电路正在被现场可编程门阵列所取代。本书将着力于帮助读者理解这些新的环境所带来的变化。

硬件综合很复杂，不像软件编译那样简单，称职的硬件设计师可能要花一年或者更多的时间才会熟悉这一技术。现在人们广泛使用 Verilog 和 VHDL 这两种硬件描述语言，即使对于最简单的模块，这两种语言也允许有各种不同的描述，可以任意使用结构化的和行为化的混合构造方式，但每一种不同的描述都会产生不同的结果，也会出现不同的综合问题。在 VLSI 技术发展的早期，Mead 和 Conway 提出的设计方法使相当广泛的用户群从中受益，并很快熟悉 VLSI 技术。现在我们需要的正是像 Mead 和 Conway 那样的标准化和精简的设计方法。本书就试图通过选择简单的 Verilog 语言和标准化的方法来实现这一目标，使高年级的本科生和研一的学生在一个学期内就可以通过门级仿真，理解中等规模到更加复杂的电路设计。本书既不是对 Verilog 语言的完整描述，也不是对仿真、综合和 FPGA 芯片配置工具的完整介绍，读者可以查阅相应参考手册来了解这些内容。然而通过使用本书并配合一些实例，学生就能够在一个学期之内完成一些有趣的项目。当然，这需要认真地学习一个学期。可以预计，以后工业界对具有这方面专门知识人才的需求甚至会超过对那些具有系统软件专业知识人才的需求。通过本书的学习，读者将具有驾驭完整设计过程的自信，可以完成更加复杂的设计并且可以有更多的闲暇时间来查阅各种手册。

在这个领域中还有另外一些比较好的书，下面我们介绍已经出版的几本：

《Application Specific Integrated Circuits》[1]是一本很好的“百科全书”，它几乎涉及了所有 ASIC 的相关内容，该书强调电路图设计、版图以及采用 VHDL 或 Verilog 描述所支持的器件物理特性。《Verilog HDL: A Guide to Digital Design and Synthesis》[2]、《Verilog Digital Computer Design》[3]和《Modelling，Synthesis，and Rapid Prototyping with the Verilog HDL》[4]这几本书则专门集中讨论 Verilog 语言自身。在上述所有的著作中，首先都是介绍用于仿真的所有 Verilog 语言结构，然后提出综合对 Verilog 语言的限制。本书则从一开始就介绍综合问题，并将综合贯穿于本书的始终，这样可使读者能尽早地通过综合来实现设计。

下面介绍的几本著作适合于已经熟练掌握高级设计方法的读者，它们更适合作为专业参考书。《Logic Synthesis using Synopsys (2nd ed.)》[5]的特点是包括了大量的 VHDL/Verilog 和 Syonpsys 解决方案提示和用例；《Re-Use Methodology Manual》[6]包含了工具的使用和作者推荐的描述风格，特别针对在知识产权(IP)市场上重复使用的模块，但是它没有具体的工程例子。因此这两本书更适合于作为其他一些基本教材的补充。

我们同样应该介绍 Gajski 和 Micheli 所著的一系列关于综合的书，它们是属于另一种类型的书，注重于综合领域的研究，因此适合于将要从事综合器设计的一些人，而不是将来要使用综合器来从事实际设计的大多数读者。

至今仍没有一本专门面向综合来论述编码风格的书，这种书应该用实际工作中的例子来说明可综合的编码风格，从而可更适用于在这一领域刚开始工作的读者。这是一个需要填补的空白领域，我们希望本书能够做到这一点。本书中所有的例子和每一章末尾的练习都已经过综合，并且成功地通过了门级仿真验证。这些内容已经在教学中使用三年多了，主要对象是那些具有计算机组织结构和 UNIX 与 X Windows 工作原理知识的高年级本科生以及研一的学生。虽然现在的工具可以在 NT 和 Linux 环境上运行，但是工业界许多从事设计的工程师仍然喜欢 UNIX 和 X Windows 环境。

对于高校来说，在厂商的“大学计划”支持下，只要象征性地支付维护费用就可以得到综合和配置工具，有时仿真器和工具会成套打包。本书中的练习题使用 VCS 编译仿真器，来自 Synopsys 的 FPGA Compiler 和 Behavial Compiler，以及来自 Altera 和 Xilinx 的 FPGA 的工具包。当然这具有一定的专用性，有些读者可能采用的是其他工具。我们很愿意比较其他厂商所提供的综合器，但实际上很难做到这一点，因为即便是以高校购买的价格，也没有一家(包括我们)可以付得起所有不同工具的费用，更不要说涉及到的时间和资源了。我们现在所能做的就是选择一些当前可用的最好工具，并且相信与使用者在其他环境下所面临的许多问题是类似的。例如，尽管 Verilog 和 VHDL 有很大的差别，但是当前主要厂商所提供的用来综合的子集却是非常相似的，这就是为什么可以使用自动翻译器的原因。由厂商提供的与本书内容相关(包括仿真、综合和 FPGA)的不同领域的完整工具列表，将定期被列举在商业期刊杂志上，包括 Integrated System Design(www.isdmag.com)。有兴趣的读者可以访问厂家的代理商或其主页来了解各种工具的相对优点，同时可知道通过大学计划能够获得什么工具。

本书适合对布尔代数和计算机组织结构有初步了解的高年级本科生或研一的学生。对于本书中的练习题，需要有 UNIX 和 X Windows 的操作经验。如果具有 C 或 Java 编程语言的知识就更好了，但这并不是必需的。通常我们发现，具有计算机科学背景的学生学习这门课时往往带有更多“软件”方面的意识，而具有电子工程背景的学生学习这门课时往往对“门级网络”有更好的理解，因此这两类读者在开始学习本课程时具有同样的潜能。

我们需要对章节的顺序做一些解释，它们初看起来有点特殊，原因之一是为了学生可以边做边学。因此，我们的方法是在最早可实践的地方先介绍仿真器的使用，然后再介绍综合器。这样，学生可以加快对工具的学习，大约经过半个学期就可以结合一个小型设计项目，把高级仿真、设计编译器、FPGA、适配和门级仿真整个过程都实践一遍。在掌握这些内容后，在剩余的半个学期，他们将会有信心来迎接更大的独立项目、行为编译器，或者下载并测试真实的硬件。这样，我们选择的顺序可能会打断设计语言和例子描述的连续性。如果本书的读者不满意这样的体系结构，也可以重新排列章节，例如可以将第 4 章关于仿真的介绍放在第 7 章之后，或者将关于描述的章节(第 3 章、第 5～9 章)合并在一起，或者将关于测试和调试的章节(即第 4 章、第 7 章、第 15 章)合并在一起，或者将关于设计编译器和行为编译器(第 12 章、第 13 章和第 16 章)的内容合起来放在 FPGA 综合的章节的前面。

为了方便读者对例子进行实验，可查阅 Prentice Hall 网站上的主页：
http://www.prenhall.com/smith/franzon

参 考 文 献

[1] Smith，Michael J S. Application Specific Integrated Circuits. Addison-Wesley，1997.

[2] Palnitkar，Samir. Verilog HDL：A Guide to Digital Design and Synthesis.Sunsoft Press，1996.

[3] Arnold M G. Verilog Digital Computer Design. Prentice Hall，1999.

[4] Ciletti M D. Modelling，Synthesis，and Rapid Prototyping with theVerilog HDL. 1999.

[5] Karup，Pran，Taker Abbasi. Logic Synthesis using Synopsys (2nd ed).Kluwer，1997.

[6] Keating，Michael，Pierre Bricaud. Re-Use Methodology Manual. Kluwer，1998.

目　录

第 1 章 概 论

目前在一个芯片上可以集成 5000 万个以上的有源器件，而且芯片的复杂度还在不断提高，因此设计者已经不可能用基于电路图的方法来设计硬件了。现在集成电路设计几乎完全来自于高级描述及综合。尽管这一领域的方法学还没有很好的建立，但是为了能跟上芯片复杂度迅猛提高的步伐，设计工具也在快速的发展中。

为了加深对这个演变过程的认识，很有必要回顾一下随着早期器件工艺的变化，设计方法学是如何发展的。在 1980 以前的大规模集成电路(LSI)时代，芯片设计工程师“用手和膝盖趴在地上”来裁剪和粘贴放大的芯片版图。或许这就是为什么把芯片版图设计称为平面布置(floorplan)的缘故。随着超大规模集成电路(VLSI，被定义为芯片上的晶体管数超过 10 万个的集成电路)的出现，以前的那种设计方法已经不可行了，伴随着出现了一些诸如原理图输入、版图编辑、参数提取以及仿真工具等计算机辅助设计工具，设计工程师能够采用更多高级的方法进行设计。此时，通过对版图规则进行抽象和简化[1]，使得这些技术易于被在校大学生掌握。随着器件制造工艺继续发展到深亚微米级线宽时代，芯片上可以集成 1 亿个甚至更多有源器件并且时钟可达 1 GHz。这时很难在版图和门级上来理解相应的电路。因此，高级的文本描述(语言)及综合取代了电路原理图输入方式，芯片复杂度现在由这些工具报告中所给出的门数来定义①。在主流的动态 CMOS 工艺中，每个这样的门大概对应 5 个有源器件(晶体管)，目前芯片集成度在 20 万至 1000 万个门范围内。本书中的设计实例当然比这个规模小得多。

这种演变与上一代软件工程学的发展非常类似。人们不愿意阅读由软件编译器生成的汇编代码，而是更喜欢在源代码中进行调试。同样地，对于一个像“闹钟电路”这样简单的描述，综合器将会把不到两页的高级源代码转变为超过 40 页的门级电路图，那么我们更愿意对高级源代码而不是对门级电路图进行调试。

然而在技术发展的现阶段，硬件编译与软件编译并不完全相同，可以通过图 1.1 和图 1.2 做比较。

在软件处理过程中，语法和类型错误在编译阶段被发现，而一些设计错误通常在运行阶段仍然存在，这时需要返回到源代码进行必要的改正。作为选择，可采用一个符号调试器来将运行错误和对应的源代码段联系在一起，同时也可以使用剖析器(profiler)去寻找那些频繁使用的代码段，对这些代码段的优化是提高程序运行效率的最有效方法。

① 遗憾的是并非所有情况下复杂度都可以简单地用门数表示(见第 12 章)。

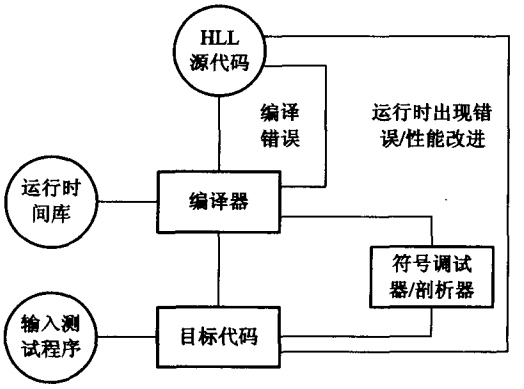


图 1.1 软件编译

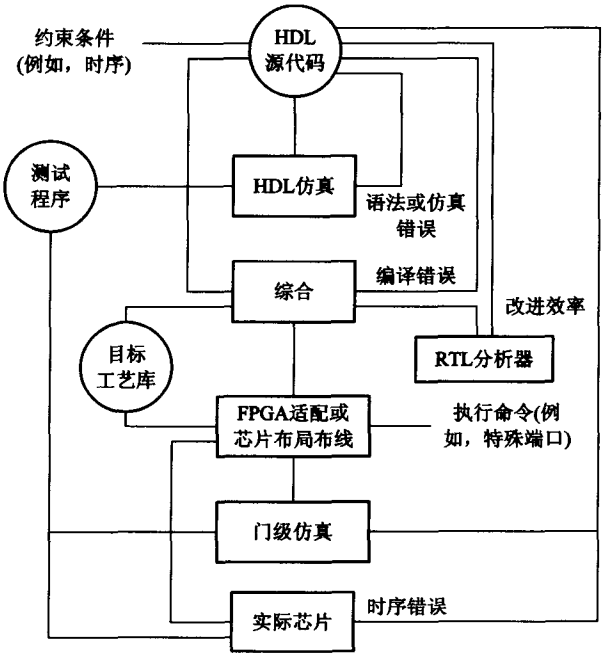


图 1.2 硬件综合

在硬件综合中，需要更多的处理阶段，在每一个阶段都可能暴露出一些更深层次的错误。第一个阶段是高级源代码描述的仿真，以决定整体设计是否正确(第 4 章)。当设计通过此阶段后，源代码被提交到综合器的解析(analysis)和推断(elaboration)阶段，在这里会发现源代码中更深层次的错误(第 12 章)。可综合语言结构是完整 HDL 语言的一个子集，它被仿真器所支持。在目前的技术状况下，被仿真器支持的源代码很有可能不被综合器支持，偶尔也会发生相反的情况。作为选择，可以使用 profiler 工具来得到设计中哪些部分占用的芯片面积最大以及产生的时间延迟最长等信息。接下来，设计提交到综合器编译阶段(第 12 章)。只有在此之后，当设计被放置在芯片上时(第 13、14 章)，才能知道电路和连线的实际延时。此时需要将设计再次提交仿真或使用形式验证工具进行验证(第 7 章)。只有当设计被送到制造厂进行“流片”时，才能真正期望设计中已不再存在任何错误。然而，不像软件可以完全正确地复制，在将设计“复制”为芯片的制造过程中仍然有可能发生错误，因此需要

通过芯片内部或外部设备进行另一层次的测试(第 15 章)。如图 1.2 所示，如果只从外部管脚上对相关的电路节点进行激励和读取，就可以在所有测试阶段上使用同一个测试平台程序。

在所有的处理阶段都有可能出现错误，这就需要返回到最初的源代码中进行修改。在整个处理过程中，高级源代码是理解和改正设计的惟一参考点。因此，源代码的描述风格和最终综合结果的质量有很大关系。

除了所包含的步骤增加外，硬件综合器还有很多编译指令和编译程序开关(在本文中它们没有全部出现)。一个能干的硬件设计师需要一年甚至更多的时间才能熟悉和掌握这种技术。另外，目前广泛基于硬件设计描述语言的用法允许种种不同的描述，包括从纯行为到纯结构化，或者介于两者之间的任何级别的描述。对同一事件的不同描述方法就是我们所说的编码风格。对编码风格的讨论从第 3 章开始一直贯穿本书。通常，在设计底层的基本模块时采用行为描述风格，而在上层采用结构和行为的混合描述风格。对于一个给定的设计，不同的描述风格会产生不同的结果，同时也会引发不同的综合问题。

那么综合到底是什么呢？

- 综合是一个来自被称为开关理论的组合和时序状态机最小算法的自动集合；
- 综合是一种从行为描述到纯粹的结构描述(网表)的翻译；
- 综合是一种从粗略描述(高层)到细致描述(底层)的转化。

这些都是综合，但是这些观点过分简化，因为综合过程中还涉及到优化和选择。

今天通常使用两种硬件描述语言：VHDL 和 Verilog。看起来 VHDL 颇受美国政府部门、学术界以及欧洲的欢迎，而 Verilog 受到绝大多数美国和日本商业设计者的欢迎。Verilog 流行的一个原因可能是因为许多商业用户习惯于它的语法，Verilog 语法源于人们已经很熟悉的 C 语言，而 VHDL 语法是 ADA 语言的派生；另一个原因可能是 Verilog 是一种更小、更简单的语言。尽管有时候这会限制一些设计描述，但是同样也有一些证据可以证明它能加速一个工作样机的设计[2]。由于我们的目标是进一步简化这个过程，因此我们选择了 Verilog。

不管是 Verilog 还是 VHDL，都可以用很多的方法来描述硬件，甚至是那些最简单的硬件模块。本书覆盖了传统描述风格，它渗透在第 8 章的间接描述风格中。第 9 章中我们将介绍一种被称为控制点风格的常规描述形式，这是对设计重用和套用设计的初始入门。这种风格继承了传统风格可以直接被工业仿真器和综合器支持的优点，同时又简单、经济，这主要是因为它大量使用了分层结构和库单元，同时依靠综合器来推断控制状态机。就上述最后一点而言，它与 Synopsys Behavioral Compiler 风格相似(第 16 章)，但是对第 5 章到第 15 章中的例子，我们只是用更容易得到的 Synopsys Design Compiler 和 FPGA Compiler 来处理。我们发现在设计中用这种风格产生的描述比适用于相同硬件的其他描述几乎缩短了一个数量级，并且没有增加速度和面积的负担。读者没有必要一定跟随这种风格，可以随意使用所讨论的其他实用风格。但是在所有的例子中通过选择和遵循一致的风格，读者将发现它明显减少了每一个例子从开始分析到达到一个工作状态所需要的时间。一般我们认为描述越短，工作样机的速度就会越快。

最后必须注意的是，Verilog 和 VHDL 语言起初都是为仿真而不是为综合设计的。不像图 1.1 所示的软件编译，综合器仅支持被仿真器支持的描述语言的一部分子集。通常的教学方法是首先讲述仿真器支持的所有结构，然后作为一个附件给出综合的一些限制。相比之

下，本书在随后各部分内容的介绍中都会涉及到综合问题，目的是帮助读者能更好地理解和使用众多厂商提供的手册资料。同时，教学经验表明，仅通过本教材就可以让学生能第一次成功地完成一个适中规模的完整设计过程。

参 考 文 献

[1] Meade C，Conway L. Introduction to VLSI Systems. Addison-Wesley，1980.

[2] Cooley J. EDA consumer advocate diary. Integrated System Design，1995(7)：56-60.