高 等 院 校 教 材

SystemC 片上系统设计

陈　曦　徐宁仪　编著

周祖成　审定

科 学 出 版 社

北　京

内 容 简 介

SystemC 是被实践证明的优秀的系统设计描述语言，它能够完成从系统到门级、从软件到硬件、从设计到验证的全部描述。SystemC 2.01 已作为一个稳定的版本提交到 IEEE，申请国际标准。

本书为配合清华大学电子工程系 SystemC 相关课程的教学而编写。全书分 9 章，内容包括：硬件描述语言的发展史；SystemC 出现的历史背景和片上系统设计方法学概述；SystemC 的基本语法；SystemC 的寄存器传输级设计和 SystemC 的可综合语言子集，以及根据作者设计经历归结的 RTL 设计准则和经验；接口、端口和通道等 SystemC 行为建模的基础、交易级建模和通信细化；SystemC 的 Master-Slave 通信库；一个 SystemC TLM 建模实例——片上总线系统；SystemC 与 VHDL/Verilog HDL 的比较；SystemC 的验证标准和验证方法学；SystemC 开发工具 SystemC_win、WaveViewer 等，以及使用 MATLAB 进行 SystemC 算法模块的验证。每一章都精心编写了课后习题以配合教学的需要。

本书可作为大学电子设计自动化（EDA）相关课程教材，也可供电子工程技术人员作为 SystemC设计、应用开发的技术参考书。本书丰富的实例源代码特别适合初学者根据内容实际运行、体会，举一反三，以掌握SystemC进行应用系统设计。书中全部源代码可以从 http://www.sciencep.com上获得。

图书在版编目（CIP）数据

SystemC 片上系统设计/陈曦，徐宁仪编著. —北京：科学出版社，2003.10

高等院校教材

ISBN 7-03-012292-5

I. S… Ⅱ.①陈…②徐… Ⅲ. SystemC 语言–程序设计–高等学校–教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2003）第 091861 号

责任编辑 杨 凯 筱 戎　**责任制作** 魏 谨

责任制印 刘士平　**封面设计** 李 力

科学出版社出版

北京东黄城根北街 16 号 邮政编码：100717

http://www.sciencep.com

印刷

北京东方科龙图文有限公司制作

http://www.okbook.com.cn

科学出版社发行 各地新华书店经销

2004 年 1 月第 一 版 开本：B5（720×1000）

2004 年 1 月第一次印刷 印张：20

印数：1～5 000 字数：367 000

定 价：30.00 元

（如有印装质量问题，我社负责调换〈新欣〉）

序

随着集成电路制造技术的发展，人们已经可以将包括存储器、信号采集和转换电路、DSP、CPU 核等模拟、数字和混合电路构成的一个完整的电子系统集成到一个芯片上，从而产生了片上系统（System on Chip）的概念。片上系统的核心技术是深亚微米的甚大规模集成电路制造技术和系统设计技术。

一方面，电子系统越来越复杂；另一方面，其市场寿命也越来越短，一个新产品往往上市几个月就被功能更新、更强大的产品代替了。这给电子设计带来了严峻的挑战。

在传统的设计方法中，首先由系统工程师进行系统规范，进行系统分割，接着硬件与软件分开设计。硬件使用硬件描述语言如 VHDL 或者 Verilog HDL 实现，软件使用 C、C++或者汇编语言实现。软件的设计往往要迟于硬件。这种设计方法的缺点是系统设计、硬件设计和软件设计使用不同的语言，无法进行软硬件协同验证，系统验证成为制约设计效率的主要因素，迫切需要一种通用语言能够完成从系统、软件到硬件、门级各个层次的设计描述和验证，这就是系统设计语言（System Level Description Language）。

当前系统设计语言的发展方向有两个。一是扩展传统的硬件描述语言 VHDL 和 Verilog HDL，使它们支持抽象数据类型，从而具有系统描述能力。这方面比较突出的工作之一是由 Accellera 组织（Open Verilog International 组织和 VHDL International 组织合并成立的硬件描述语言发展组织）的 SystemVerilog。二是扩展传统的软件语言 C 和 C++，使它们支持硬件描述。在 SystemC 出现之前已经有很多这方面的工作比较成功。但这方面的工作最突出的就是 SystemC。SystemC 在 Synopsys 等一批大公司的支持下得到了很大的发展，它已经成为很多 EDA（电子设计自动化）工具除 VHDL 和 Verilog HDL 之外的第三种支持语言。由于其开放性等特点，已经得到全世界工程师的欢迎和认可，许多大公司都推出了 SystemC 的开发工具。

SystemVerilog 可以粗略地理解为是 Verilog 的超集。业界多数人认为，如果它成为 IEEE 标准，很可能导致 VHDL 逐步退出历史舞台。同时业界也普遍认为，SystemC 与 SystemVerilog 刚好构成互补的关系，“SystemVerilog 的发展将会成为 SystemC 的基石”。由于 SystemC 在系统级设计与验证方面没有对手，我们认为它几乎必然会流行开来并成为 IEEE 的标准。

由于开发工具的不断推出，SystemC 在国外已经十分流行。目前已经有国内外一些大公司利用SystemC开发项目成功的案例。我们在深圳清华大学研究院EDA与网络应用重点实验室使用 SystemC 开发的数字音频芯片的流片也是使用SystemC开发成功的案例之一。我们已经在清华大学开始讲授SystemC。

SystemC之所以成功，关键在于:

(1)开放性和广泛的支持。由于SystemC扩展了C++语言，所以无数熟悉C++又对硬件描述感兴趣的设计者可以很容易地掌握。对于那些熟悉硬件描述语言的工程师，学习SystemC也很容易，对于他们来讲，只需找到他们与所熟悉的语言所对应的语法就可以了。SystemC几乎受到了所有EDA公司的支持，这也是它成功的关键。

(2)抽象描述能力。SystemC是在系统级和交易级建模的最佳语言，同时它能够很好地支持寄存器传输级设计和验证。

(3)仿真速度。SystemC描述的系统是可执行的文件，仿真速度远远大于逐行解释的硬件描述语言，这一点对于越来越复杂的电子系统来讲十分重要。

周祖成

2003年9月于清华园

编者序

1 写书背景

到目前为止，成为 IEEE 标准的硬件描述语言有两个: VHDL 与 Verilog HDL。VHDL 的英文全称为 VHSIC Hardware description language，其中 VHSIC 指 Very High Speed Integrated Circuit。VHDL 出现于 1980 年，并于 1987 年开始成为 IEEE 标准，称为 IEEE Std.1176。Verilog HDL 语言出现于 1983 年，并于 1995 年成为 IEEE 标准，称为 IEEE Std 1364—1995。与 VHDL 和 Verilog HDL 同时存在的语言如 Superlog、SpecC 等，由于它们由专门的公司所有，并非开放标准，所以不是很流行。

VHDL 与 Verilog HDL 并称为传统硬件描述语言。基于它们的设计方法是，在系统级一般采用 C/C++等高级语言进行算法描述，然后再手工转换为 VHDL 或者 Verilog HDL。由于两个阶段使用不同的设计语言，不能够很好地结合起来，软硬件协同设计和仿真、验证很困难。随着集成电路工艺水平的提高，现代业界进入了片上系统集成(System on Chip)的时代，一大特征就是软件设计的比重越来越大。然而，使用传统设计方法只能在硬件设计接近完成时才能开始软件的设计和调试，从而导致设计成为整个集成电路产业的瓶颈。

基于以上原因，人们致力于开发一种新的设计语言，它能够提供全面的系统描述能力。由于在软件领域 C/C++的绝对优势，人们渴望能够利用它们来描述硬件。这方面的研究工作已经开展了 10 多年的时间，一直处于自发的状态，直到 1999 年 SystemC 的出现才改变了这一格局。

1999 年 9 月，一些微电子业内一流的 EDA 公司、IP 提供商、半导体制造商及系统和嵌入式软件设计公司在加利福尼亚州 Saint Jose 举行的“嵌入式系统会议”上，联合创建了开放 SystemC 推动（OSCI 或者 Open SystemC Initiative）组织，并推出了基于 C++的系统设计语言——SystemC。OSCI 是一个非盈利性组织，它负责维护和发展 SystemC。SystemC 是完全免费的，这使得 EDA 供应商能够充分自由地了解 SystemC 库的源代码以优化它们的设计工具。

SystemC 出身名门，是绝对的宠儿，它一开始出现就受到业界前所未有的欢迎。又由于 SystemC 是开放的标准，所以它很快就流行开来。目前业界知名的公司包括 Synopsys、Cadence、Frontier Design、ARM、Erission、Lucent、Sony、Synopsys、TI 等都支持 SystemC。

SystemC 是一种完成电子系统从软件到硬件的全部建模过程的语言，比起

VHDL/Verilog HDL，其在系统设计方面的优势明显。

SystemC 最近的版本是 2.01，目前许多 EDA 厂商都在开发或者扩展它们的设计工具以支持 SystemC。现在已经推出的商用 SystemC 软件有 CCSS、SPW、SystemC ART 等 50 多种。已经有很多著名公司（包括西门子、ARM、Alcatel、富士通，中国的东方通信、大唐飞利浦等）开始使用 SystemC 作为其系统级开发语言。一些归国的留学生在他们自己的小公司里也已经开始使用 SystemC 进行开发和验证。作者所在的实验室之一的清华大学深圳研究院 EDA 实验室已经将其用 SystemC 设计的第一款数字音频芯片进行了流片。

业界一般认为，SystemC 可望在 1～2 年之内成为 IEEE 标准。目前 SystemC 2.01 作为一个稳定的版本已经提交到 IEEE 进行标准化。SystemC 的标准化会加速 SystemC 的应用和发展。

目前，另外一种被业界看好的硬件描述语言是 SystemVerilog，它是 Verilog 2001 标准的扩展，增加了类似 C 语言的结构和增强的验证功能，它与 SystemC 恰好形成互补的关系。SystemC 和 SystemVerilog 在不同的设计领域中，分别有各自的特点。两种语言的结合，将从系统规范、门电路布局直至设计验证，为设计人员提供一种可供选择的综合语言环境。这种结合可以看作是工具和语言相统一的转变潮流中的重要部分，这种统一是为了满足最终用户的需求，而不仅仅是为了方便工具供应商。

需要特别澄清的是，SystemC 语言的主要目的是实现用单一语言完成设计规范、系统体系结构分析、验证平台和行为模型，而寄存器传输级和门级建模并非 SystemC 所长。现在的很多设计都从算法开始，SystemC 比 C/C++更适合对算法的描述和分析。传统硬件描述语言和 SystemVerilog 的 C 接口不支持硬件设计中的并发性、层次性和互连，所以并不适合描述复杂多样的系统行为。通常用 SystemC 的行为模型描述的模块比 RTL 模块仿真速度快 10～100 倍，这是 SystemC 的价值之一。SystemC 的更大价值在于其提供的高层次设计流程。基于渐趋成熟的行为级综合工具和 SystemC 行为模型的高层次设计流程将会大大提高设计和验证效率。SystemC 特别为行为综合设计的语法将使它成为非常适合行为级综合的设计语言。

2 本书内容安排

本书内容安排如下：第 1 章着重介绍硬件描述语言的发展史，以及 SystemC 的出现和成为 IEEE 标准的必然趋势。第 2 章介绍 SystemC 的基本语法，包括模块、端口、信号、进程、基本数据类型、定点数据类型、sc_main()函数以及波形跟踪等。该章内容是所有后续章节的基础，读完了第 2 章就可以完成全部的 SystemC 寄存器传输级建模。第 3 章着重讨论 SystemC 的寄存器传输级设计和 SystemC 的可综合语言子集，并根据作者的设计经验讲述 RTL 设计的一些准则和经验。第 4

章将围绕 SystemC 行为建模展开，前半部分着重介绍接口、端口和通道等 SystemC 行为建模的基础。后半部分介绍 SystemC 行为建模中的交易级建模(transaction level modeling)，内容涉及系统建模的分层模型、交易级建模的基本概念和通信细化。第 5 章讲述 SystemC 提供的 Master-Slave 通信库，这个库尤其适用于包含一个或者多个处理器核、数字信号处理器、周边设备和用户定制集成电路而且是利用一组总线相互通信的系统建模。第 6 章集中介绍一个 SystemC TLM 建模实例——片上总线系统，包括主从设备接口、快速/慢速存储器通道、通用串口通道、仲裁器接口、仲裁器模块、仲裁算法的 SystemC 实现。第 7 章写给那些熟悉传统硬件描述语言 VHDL/Verilog HDL 或者需要将传统硬件描述语言写成的 IP 向 SystemC 进行手工转换的读者。通过 SystemC 与 VHDL/Verilog HDL 的直接比较，读者可以很快地掌握和更好地理解 SystemC。第 8 章将简要介绍 SystemC 的验证标准和验证方法学，主要描述 SystemC 的验证标准的基本思路和主要功能。通过该章，读者可以对基于交易的验证 SystemC 方法有一个初步的了解，并且可以利用 SystemC 库进行简单的验证平台(Testbench)的编写。第 9 章是本书最后一章，介绍一些基于个人电脑的 SystemC 开发工具，包括 SystemC_win、WaveViewer，并详细介绍了如何使用 MATLAB 进行 SystemC 模块，尤其是 SystemC 算法模块的验证。

从电子系统建模层次的角度看，第 1、2、3 章描述 SystemC 的寄存器传输级建模；第 4、5、6 章讨论 SystemC 的行为级建模；第 7 章阐明 SystemC 与传统硬件描述语言的等效性；第 8 章介绍 SystemC 的验证；第 9 章给出 SystemC 的设计工具。其中第 5 章和第 8 章又分别围绕 SystemC 的 Master-Slave 库和验证库展开。

本书第 2.4 节描述的是 SystemC 定点数据类型，这一节是与后续章节无关的，读者如果在工作中用不到，可以略过本节。在本书作者看来，这一节放在第 2 章是否合适还有待讨论。

本书的每一章节中都给出了较多的典型设计实例以加深读者对这些概念的理解，读者可以对照着这些代码学习。

作为教材，作者建议在课堂上集中讲授第 1、2(不包括 2.4 节)、4、5、8 章，其他章节可以作为课外阅读内容。本书每一章都安排了习题，可以作为课后作业。同时可以考虑以第 6 章的片上总线模型为起点，安排一个复杂度和任务量适中的课程设计。

3 写书目的与读者对象

清华大学电子工程系已经开始在其《电子系统仿真和 VHDL》和《通信系统仿真与 ASIC 设计》两门课中讲授 SystemC，本书写作的主要目的是作为这两门课程的教材。本书作者的导师，也是这两门课的授课教授，对于本书的编写给予了细

致入微的指导。

本书出版的另外一个目的是配合 SystemC 在国内发展的需要。目前，国外已经有若干本 SystemC 方面的图书，而据作者所知，中文的相关出版物却一直没有。本书作为国内 SystemC 方面的第一本书，希望能够方便国内工程师对 SystemC 的了解、学习和使用。

作为教材，本书在注重对基本概念的诠释的同时，还引入了大量的实例以加深读者对所述内容的理解。

本书可以作为大学相关课程的高年级本科生和研究生教材，也可以作为工程技术人员的参考手册。本书读者需要有 C++的基础知识，包括 C++中的纯虚函数和模板类。如果读者已经掌握了一种硬件描述语言（如 VHDL 或者 Verilog HDL），同时又具有 C/C++的基本知识，那么阅读本书将会如鱼得水。

如果读者是一位软件工程师，阅读本书可以迅速掌握数字电路的设计和建模。如果读者是一位硬件工程师，通过阅读本书，将可以掌握新一代描述语言，从而更加有效地与软件工程师合作。作为 IC 设计公司的负责人，可以藉助本书了解业界最新的设计方法学和电子系统设计领域的动向，缩短公司产品上市时间（time to market），以在竞争中获得先发优势。

本书所有实例都有源代码，读者可以从科学出版社的网站（http://www.sciencep.com）下载或者向本书作者索取。

4 作者简介

本书以两个高年级博士生为主完成，他们的研究方向都是片上系统集成（SoC）。他们都有丰富的 Verilog HDL 和 VHDL 的开发经验，参与过骨干网路由器项目的开发，独立完成过百万门级 FPGA 的设计，并于 2002 年下半年和 2003 年上半年参与国家 863 项目期间使用了 CoCentric System Studio 和 SystemC 进行了系统建模和仿真，最终系统满足了设计要求。本书作者还多次在国内外期刊和会议上发表有关集成电路设计和 SystemC 方面的文章，他们都是中国通信学会学生会员、IEEE 学生会员。

本书第 1、2、4、6、7、9 章由陈曦完成；第 3、5、8 章由徐宁仪完成。

本书的编写得到了作者导师周祖成教授的大力支持和指导。作者导师长期从事 EDA 相关领域的研究，并在清华大学主讲《电子系统仿真和 VHDL》和《通信系统仿真与 ASIC 设计》这两门课。

5 致　　谢

本书作者首先感谢恩师周祖成教授将自己带入电子设计自动化和集成电路设

计这一前景广阔、研究活跃的科学领域，感谢在同一实验室工作的罗飞同学、贺光辉博士、刘付娥博士等为本书的出版所做的努力。

本书第一作者在此感谢妻子李光在本书出版过程中所给予的理解和支持。

本书第二作者在此感谢杜爽同学对其工作的大力支持和无私的帮助。

感谢 OSCI(Open SystemC Initiative)的广大工程师在 SystemC 发展中所付出的不懈努力。

感谢科学出版社为我们出版了本书。没有出版社同仁细致认真的工作，本书也无法与读者见面。

6 再版目标

作者希望在 SystemC 进一步发展以后能够再版本书，修正本书第一版中的错误，吸收第一版读者的意见，增加新的语法内容，并给出在工作站和个人电脑上都能使用的源代码，添加更加详尽的寄存器传输级设计的经验方法和系统级设计方法学，还可能引入 SystemVerilog 的部分内容。

7 反　　馈

本书作者力求将本书写好，但一则工作中纰漏在所难免，二则作者能力有限。对于本书的任何意见和建议，欢迎读者联系本书作者。

本书作者联系方式:

清华大学东主楼 9 区 324 房间　清华大学微波与数字通信国家重点实验室 CAD 中心

陈曦 徐宁仪

邮编 100084

作者的校内Email: chenxiee@mails.tsinghua.edu.cn

xuny97@mails.tsinghua.edu.cn

作者的公众网 Email : chenxi01@tsinghua.org.cn

您的建议和意见将融入本书，成为本书第二版出版的最直接的动力。本书作者就此诚谢。

编　者

2003 年 9 月于清华园

目　录

第 1 章　电子系统设计方法学和系统级描述语言概述

第 2 章　SystemC 基本语法

第 5 章 SystemC 的 Master-Slave 通信库

第 6 章 TLM 设计实例——片上总线

第 8 章 基于 SystemC 的验证方法学

第 9 章 SystemC 的开发工具

第 1 章　电子系统设计方法学和系统级描述语言概述

本章着重介绍硬件描述语言的发展史，以及 SystemC 的出现和成为 IEEE 标准的必然趋势，简要分析 SystemC 与传统 HDL 相比的优势所在，并简要讨论 SystemC 的建模能力。最后通过一个 2 输入与非门的设计实例，说明如何使用 Visual C++ 6.x 进行 SystemC 设计输入和编译，以及如何用 ModelSim 进行仿真。

1.1 片上系统对设计描述语言的要求

一般来说，一片集成电路开发的四个步骤是设计、制造、封装和测试。随着集成电路制造技术的发展，电子系统变得越来越复杂，人们已经可以把复杂的电子系统集成到一个芯片上，这就是所谓片上系统（System on Chip，SoC）。一个典型的嵌入式系统通常包括处理器核、片上存储器、中断控制器、定时器、通用串口、通用 I/O、定制外设、实时操作系统（Real Time Operating System，RTOS）和应用软件。这样复杂的系统对设计自动化的要求越来越高，而事实上设计已经成为整个集成电路产业的瓶颈（bottleneck）。

图 1.1 显示了集成电路按照摩尔定律发展的历程，它以存储器和处理器（CPU）的发展为标记。同时，也显示了设计能力和制造能力之间的差距越来越大。

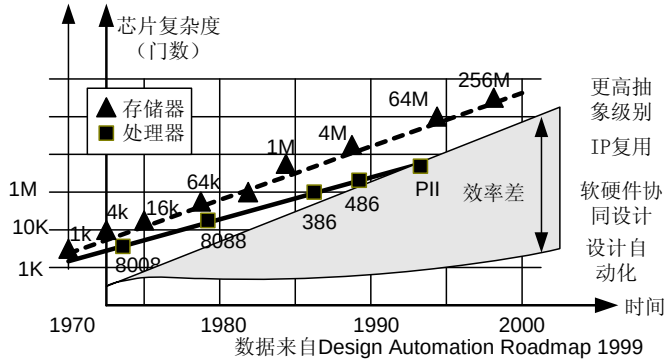

图 1.1　集成电路依照摩尔定律发展的历程

为了提高 SoC 时代集成电路的设计效率，人们提出了基于平台和基于高层次

综合的设计方法学。所谓基于平台的设计，其关键在于知识产权（Intellectual Property，IP）的设计重用（design reuse），人们希望像搭积木一样将一个复杂的集成电路构建起来；所谓基于高层次综合的设计，首先是对电路进行比寄存器传输级（Register Transfer Level，RTL）更抽象的描述，然后利用高层次综合工具完成到 RTL 电路的转换。这两种方法有各自的长处，也有各自的缺点。

无论采用什么样的设计方法学，人们都需要对 SoC 时代的复杂电子系统进行描述，以选择合适的系统架构，进行软硬件划分、算法仿真等等。描述的级别越低，细节问题就越突出，对实际系统的模仿就越精确，完成建模所消耗的时间、仿真和验证时间就越长。相反，描述的抽象级别越高，完成建模所需要的时间就越短，但对目标系统的描述也就越不精确。计算机的运行速度是有限的，设计人员必须在速度和精确性之间作出折衷。

在传统设计方法中，设计的系统级往往使用 UML、SDL、C、C++进行描述，而在寄存器传输级（Register Transfer Level，RTL）使用硬件描述语言（Hardware Description Language，HDL）进行描述。最广泛使用的两种硬件描述语言是 VHDL 和 Verilog HDL，在设计细化（refinement）阶段，原始的 C 和 C++描述必须手工转换为使用 VHDL 或者 Verilog HDL。这种设计方法的缺点是使用不同的语言进行系统描述的不一致性。随着系统越来越复杂和相较之下市场时间（Time To Market，TTM）越来越紧迫，人们迫切需要一种语言单一地完成全部设计。这种语言必须能够用于描述各种不同的抽象级别（如系统级、寄存器传输级等），能够胜任软硬件的协同设计和验证，并且仿真速度快。这就是所谓的系统级描述语言（System Level Description Language，HLDL），而传统的硬件描述语言如 VHDL 和 Verilog HDL 都不能满足这些要求。

总之，人们对系统级描述语言的要求是：高仿真速度和建模效率、时序和行为可以分开建模、支持基于接口的设计、支持软硬件混合建模、支持从系统级到门级的无缝过渡、支持系统级调试和系统性能分析等。

当前系统设计语言的发展方向有两个：一是扩展传统的硬件描述语言 Verilog HDL 或者 VHDL，使它们具有系统描述能力，如 SystemVerilog、Superlog；二是扩展传统的高级语言 C 和 C++，使它们支持硬件描述，这方面的工作最突出的是 SystemC。SystemC 在 Synopsys 等几十家业界一流的 EDA 公司、知识产权提供商、半导体系统制造商和嵌入式系统软件设计公司的支持下得到了很大的发展。由于其开放性等特点，已经得到全世界工程师的欢迎和认可。如今的 SystemC 已经成为事实上的 SLDL 标准，而 SystemVerilog、Superlog 等由于被少数公司所有，并没有得到很好的发展，但也不可否认的是它们与 SystemC 相比也有各自的优势。

1.2 传统硬件描述语言 Verilog HDL 和 VHDL

传统硬件描述语言最主要的就是 VHDL 和 Verilog HDL。

VHDL 的英文全文为 VHSIC Hardware Description Language，其中 VHSIC 指 Very High Speed Integrate Circuit，它出现于 1980 年，源自于美国国防部的高速集成电路发展计划。1987 年成为 IEEE 标准 IEEE Std.1176，1993 年进行更新，成为目前的主流版本 1176—1993。

Verilog HDL 语言最初是于 1983 年由 Gateway Design Automation 公司为其模拟器产品开发的硬件建模语言。那时它只是一种专用语言，由于该公司的模拟、仿真器产品的广泛使用，Verilog HDL 作为一种便于使用且实用的语言逐渐为众多设计者所接受。在一次努力增加语言普及性的活动中，Verilog HDL 语言于 1990 年被推向公众领域。Open Verilog International（OVI）是促进 Verilog 发展的国际性组织。1992 年，OVI 决定致力于推广 Verilog OVI 标准成为 IEEE 标准。这一努力最后获得成功，Verilog 语言于 1995 年成为 IEEE 标准，称为 IEEE Std 1364—1995。Verilog HDL 的最近一次更新是在 2000 年，即所谓 Verilog 2000。

VHDL 和 Verilog HDL 都能够完成硬件从行为级到开关级的描述，在硬件描述的 RTL 级及其以下级，SystemC 与传统硬件描述语言相比并没有优势，其中 Verilog HDL 的 RTL 描述能力最强，VHDL 则在行为级描述能力更强，而 SystemC 则具有更好的系统级描述能力。

1.3 SystemC 的历史

1999 年 9 月，微电子业内的一些一流的 EDA 公司、IP 提供商、半导体制造商及系统和内嵌式软件设计公司在加利福尼亚州 Saint Jose 举行的“内嵌式系统会议”上，联合创建了开放 SystemC 创始会（OSCI：Open SystemC Initiative）组织，并推出了基于 C++的系统级设计语言——SystemC。OSCI 是一个非盈利性组织，它负责维护和发展 SystemC。SystemC 是完全免费的，这使得 EDA 供应商能够充分自由地了解 SystemC 库的源代码以优化他们的各种解释工具；包括 Synopsys、Cadence、Frontier Design、ARM、Erission、Lucent、Sony、TI 等核心成员。目前已经有 50 多个著名的微电子公司支持该标准。

目前 SystemC 的版本是 2.01，已经有很多著名公司包括西门子、Alcatel、富士通、中国的东方通信、大唐飞利浦等开始使用 SystemC 作为其系统级开发语言。在系统开发初期，可以使用 UML 对系统需求和架构进行分析和描述，算法部分可以使用 Matlab、SPW、CCSS 等进行分析，然后利用 SystemC 进行交易级别建模（transaction level modeling），划分系统软硬件，对目标系统进行比较详细的划分，然后再利用 SystemC 完成后续流程。

图 1.2 是 2000 年和 2001 年 OSCI 统计的 SystemC 源码下载情况图。

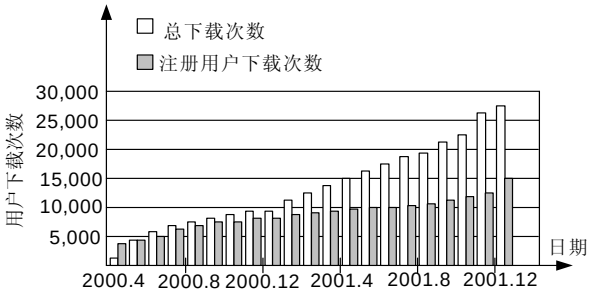

图 1.2 OSCI 统计的 SystemC 被下载的情况

SystemC 可望在 1～2 年之内成为继 VHDL、Verilog HDL 之后的又一 IEEE 标准硬件和系统描述语言。

1.4 SystemC 到底是什么

SystemC 本质上是在 C++的基础上添加的硬件扩展库和仿真核，这使得 SystemC 可以建模不同抽象级别的包括软件和硬件的复杂电子系统；它既可以描述纯功能模型和系统体系结构，也可以描述软硬件的具体实现。SystemC 源代码可以使用任何标准 C++编译环境进行编译，生成可执行文件；运行可执行文件可以生成 VCD、WIF 和 ISDB 格式的波形文件，可以使用综合工具（如 Synopsys SystemC Compiler）将 SystemC 的寄存器传输级描述综合为 Verilog HDL 的代码用于 FPGA 设计，也可以综合为门级电路用于 ASIC 设计。目前 SystemC 最高版本为 2.01，可以完成（包括）门级以上的设计描述。

SystemC 具有所有硬件描述语言所共有的基本特征，包括模块、进程、端口和信号等。不同的是，在 SystemC 中时钟被单独定义为一个特殊的信号，这大大简化了时钟信号的定义，SystemC 还支持多个时钟之间的任意相位关系。在 SystemC 中使用了 sc_set_time_resolution()和 sc_set_default_time_unit()来定义时间分辨率和时间单位，这与 Verilog HDL 的 timescale 语法在功能上是等效的。

为了支持寄存器传输级的并行描述，SystemC 采用了与传统硬件描述语言基本相同的调度模型——基于Δ（delta）延迟。一个Δ周期包括求值和更新两个阶段，在一个时间点上，这样的Δ周期会持续出现直到再求值前后的结果不再发生变化。而在宏观上，时间并没有前进。SystemC 2.01 调度模型中，在初始化阶段（相当于时间 0 点），所有进程包括方法进程和线程都将被执行一次（SystemC 1.0 中只有方法进程被执行）。不同的是，在 SystemC 中，所有的信号和变量的初始化工作在构造函数中进行，它比其他函数先执行，避免了像 Verilog HDL 中由于初始

化顺序不同引起的不同仿真器仿真结果的不一致。

为了支持进程同步和通信细化（communication refinement），SystemC 支持用户自定义的接口（interface）、端口（port）和通道（channel）。接口是方法（method）的集合，但不具体实现这些方法，在 C++语法中，它们都是纯虚函数。通道具体实现一个或者多个接口。端口定义了它能够连接的具体的接口类型，只能被用于连接实现了该类接口的通道。在有些情况下，进程可以直接读写通道而不必通过端口，而另外一些情况下则必须通过端口进程才能读写通道。

在 SystemC 中，进程只调用通道提供的接口方法。虽然接口方法是在通道中实现的，然而它是在进程上下文中被执行的。这被称作接口方法调用（Interface-Method-Call，IMC），接口方法调用和支持不同抽象级别的混合建模是通信细化的基础。

1.5 基于 SystemC 的设计流程

基于 SystemC 的设计流程与以前的设计流程本质区别在于，使用一种语言就可以完成从系统到 RTL、从软件到硬件的全部设计，整个设计的软硬件可以协同设计和仿真。SystemC 提供了在高抽象级别使用抽象数据类型对系统的硬件部分进行建模，支持软硬件早期协同设计验证，提高了建模效率和仿真速度。

图 1.3 为 SystemC 出现之前的典型设计流程。

图 1.4 为基于 SystemC 的设计流程，从 C/C++到 HDL 不再需要手工转换，一种语言就可以完成全部设计。

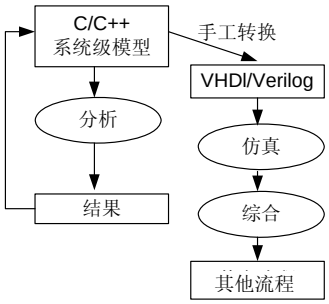

图 1.3　传统的设计流程

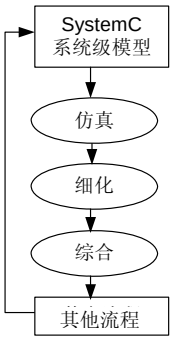

图 1.4　基于 SystemC 的设计流程

1.6 一个“Hello,SystemC!”建模实例

下面我们来看一个最简单的 SystemC 程序。其源代码如下：

```
//A "hello,SystemC!" system program
```

```
//By Chenxi,Copyright reserved 2003.3.21
//hello.h
#ifndef _HELLO_H
#define _HELLO_H
#include "systemc.h"
SC_MODULE(hello){
SC_CTOR(hello){
      cout<<"Hello,SystemC!"<<endl;
}
};
#endif

//main programme
//main.cpp
#include "hello.h"
int sc_main(int i, char* av[]){// Similar to main function in C
hello h("hello");
return 0;
}
```

使用 VC++6.x 编译后，运行结果是在屏幕上打印出“Hello,SystemC!”，这段小代码体现了 SystemC 的几个特点：

• 必须包括 systemc.h 头文件，保证程序能够正确编译。systemc.h 包括了 SystemC 的所有库文件。

• 与所有的硬件描述语言一样，SystemC 的最基本单元是模块 SC_MODULE，它是 SystemC 中定义的一个宏，一个由 SC_MODULE 定义的模块其实就是 C++中的一个类，所以它也有构造函数 SC_CTOR(hello)。

• sc_main 是所有 SystemC 设计的顶层函数（也可以说是顶层模块），等价于 C 语言的 main()函数。

当然 SystemC 中的另外一些基本构成单元和细节在这段代码中未能表达。

1.7 SystemC 的系统描述能力

SystemC 目前能够描述包括门级在内的所有更高抽象级别的数字逻辑电路和软件。由于后面将主要讨论基于 SystemC 的寄存器传输级以上的硬件描述，这里给出一个门级电路示例。

```
//Designed By Chenxi,2003.3.22
//a systemc description of 2-input nand gate
```

```
//nand2.h
#ifndef _NAND2_H
#define _NAND2_H
#include <systemc.h>
SC_MODULE(nand2){
sc_in<bool> A;
sc_in<bool> B;
sc_out<bool> F;
void do_nand(){
F=!( A & B);
};
SC_CTOR(nand2){
SC_METHOD(do_nand);
sensitive<<A<<B;
}
};
#endif
```

这是一个 2 输入与非门的 SystemC 描述。输入 A 和 B 为 sc_bit（二值，0 和 1）类型。输出也是 sc_bit 类型。进程 do_nand()对信号 A 和 B 敏感，它是一个 SC_METHOD 进程，完成对 F 的求值。

为了验证该段代码，必须再写一个验证程序（Testbench），如下面的代码：

```
//Testbench of nand2,By chenxi ,all rights reserved
//tb.h
#ifndef _TB_H
#define _TB_H
SC_MODULE(tb){
    sc_out<bool> a,b;
    sc_in<bool> f;
    sc_in_clk clk;
    void gen_input(){
        wait(); a=0; b=0;
        wait(); a=0; b=1;
        wait(); a=1; b=0;
        wait(); a=1; b=1;
        wait(100);
    }
    void display_variable(){
        cout<<"a="<<a<<",b="<<b<<",f="<<f<<endl;
    }
```

```
    SC_CTOR(tb){
        SC_CTHREAD(gen_input,clk.pos());
        SC_METHOD(display_variable);
        sensitive<<f<<a<<b;
        dont_initialize();
    }
};
#endif;
```

有了验证程序，我们就可以在 sc_main()函数中将 nand2 和 tb 例化，以验证设计的正确性。sc_main 函数的代码如下：

```
//By chenxi,all rights reserved
//main.h
#include <systemc.h>
#include "nand2.h"
#include "tb.h"
int sc_main(int, char**){
    sc_signal<bool> a,b,f;
    sc_clock clk("Clk",20,SC_NS);
    nand2 N2("Nand2");
    N2.A(a);
    N2.B(b);
    N2.F(f);
    tb tb1("tb");
    tb1.clk(clk);
    tb1.a(a);
    tb1.b(b);
    tb1.f(f);
    sc_start(200);
    return 0;
}
```

下面是仿真结果：

```
a=0,b=0,f=1
a=0,b=1,f=1
a=1,b=0,f=1
a=1,b=1,f=1
a=1,b=1,f=0
```

读者可能会注意到，在验证程序中有一行 dont_initialize()，它的作用是告诉 SystemC 的调度器不要在仿真零时刻对 SC_METHOD 类进程 display_variable 初始

化。如果没有这一行，运行的结果将会是

```
a=0,b=0,f=0
a=0,b=0,f=1
a=0,b=1,f=1
a=1,b=0,f=1
a=1,b=1,f=1
a=1,b=1,f=0
```

这个结果的第一行显然是错误的，原因是在初始化 display_variable 的时候，nand2 模块还没有来得及给 f 赋值。读者进一步学习后就会明白其中的原因。

1.8 SystemC 的开发工具

目前 SystemC 的开发工具已经有很多，据 OSCI 统计（2003 年 4 月 7 日）有 30 多种。由于 SystemC 从本质上来讲就是 C++中增加的一个类库，所以任何一个 ANSI 标准的 C++编译工具都可以对 SystemC 进行编译链接产生可执行文件。由该可执行文件生成的波形文件可以是 VCD 格式、ISDB 格式或者 WIF 格式的标准波形文件，任何支持这几种格式的波形查看工具都可以查看所产生的波形。目前 Synopsys 和 Cadence 都有商用的综合工具。

本文作者所使用的工具为：在 PC 上使用 SystemC_win 和 Visual C++ 6.0 编译，使用 ModelSim 和 WaveViewer 观察波形。在工作站（Solaris 8）上使用 G++、GCC 编译，使用 CCSS 集成环境编译仿真，使用 Virsim 和 Signalscan 看波形，使用 SystemC Compiler 综合。

考虑到 SystemC 的初学者在 PC 进行设计比较方便，本书全部代码以 Visual C++ 6.0 作为编译环境，以 ModelSim 作为仿真工具。

1.9 使用 Visual C++编辑和编译 SystemC 设计

Visual C++是广泛使用的 Windows 操作系统下的 C++编译工具，目前最新的版本是 Visual C++.net。这里讲述如何使用 Visual C++ 6.0 编辑和编译 SystemC 设计。这里我们仍然以 1.7 节的 2 输入与非门为例。

为了能够使用 SystemC 的核心库，我们首先对它进行编译。假设从 http://www.systemc.org 下载了 SystemC 的库代码并解压缩到 D:\systemc\目录下，这时可以看到 src 子目录下有一个 systemc.h 文件，这是所有 SystemC 设计必须包含的头文件。src 目录包含了所有的 SystemC 的核心源代码，与 src 同级的目录还有 msvc60、docs、examples 和 config。我们切换到 D:\systemc\msvc 60\systemc 目

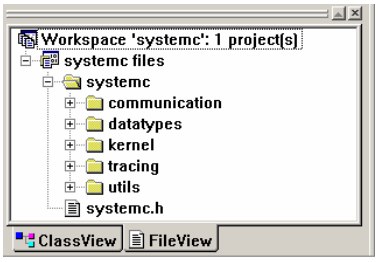

图 1.5 编译 SystemC 库时 Visual C++ 6.0 的工作间

录，用 Visual C++ 6.0 打开 systemc.dsw。这时 Visual C++ 6.0 的工作间（Workspace）将出现所有的 SystemC 的核心文件，展开 systemc 文件夹，则工作间如图 1.5 所示。

编译这个项目。正常编译的结果是产生一个 systemc.lib 文件，被放在 Debug 目录下。这个库文件是我们在所有 SystemC 项目中必须包括的库文件。

关闭 SystemC 项目，我们来新建一个自己的项目。新建项目时选择 C++ Console Application。我们假设项目的目录为 D:\SC_LIB\BOOK，项目的名字为 Nand2。如图 1.6 所示。

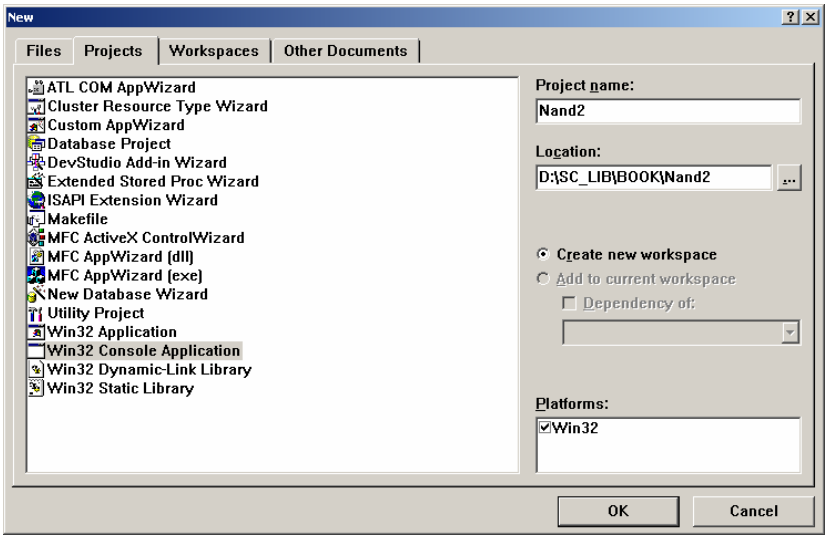

图 1.6 新建一个名为 Nand2 的项目

选择 OK，在接下来出现的对话框中选择 An Empty Project（默认），然后选择 Finish，再选择 OK。

这个时候一个空的 Nand2 项目就建立了。

为了能够让 Visual C++ 6.0 正常工作，我们还需要设置一些选项。选择 Project->settings 菜单，切换到 C/C++页面（Tab），在 Category 中选择 C++ Language，选定 Enable exception handling（默认）和 Enable Run-Time Type Information 项，如图 1.7 所示。

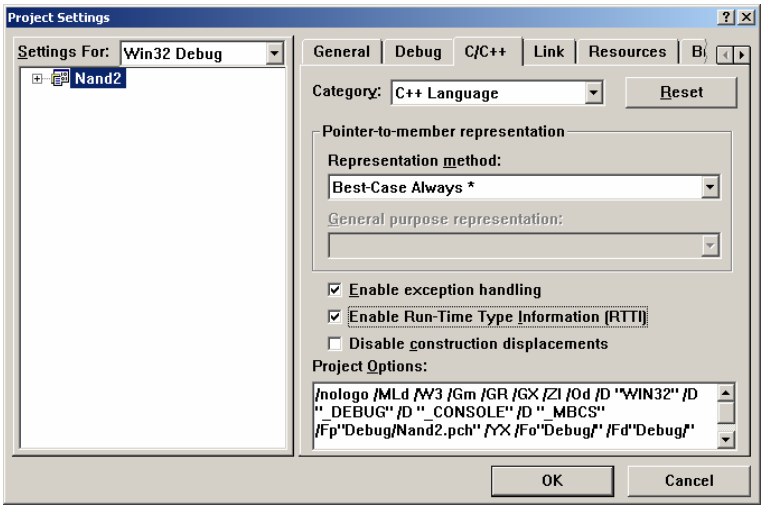

图 1.7　选定 Enable Run-Time Type Information 项

然后再在 Category 中选择 Preprocessor，在 Additional include directories 中指定一个源文件目录，这个目录就是 systemc.h 所在的那个目录。在这里我们已经假设 systemc.h 在 D:\systemc\src 目录下，所以这里我们应该填入 D:\systemc\src。如图 1.8 所示。

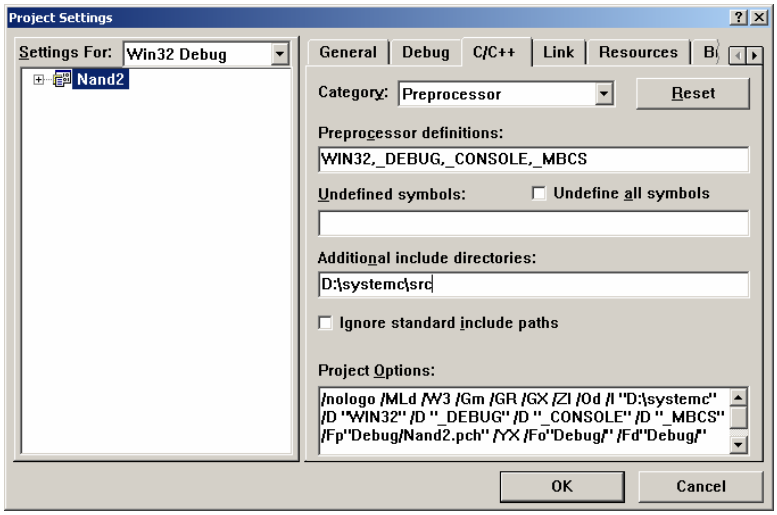

图 1.8　指定 systemc.h 的路径

再将编译好的 systemc.lib 包括到设计中。这时所有的准备工作都做好了，只需再加入 SystemC 设计文件，就可以编译整个设计了。如果还通不过，那就是设计本身有问题了，需要认真修改设计直到编译通过。

我们将 1.7 节的设计文件 nand2.h、tb.h 和 main.cpp 加入到 Nand2 的项目中。这时的工作间如图 1.9 所示。

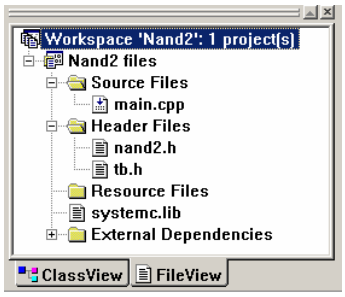

图 1.9 Nand2 项目的工作间显示

我们编译并执行这个设计，就会得到 1.7 节所显示的结果。

1.10 利用 ModelSim 查看 SystemC 产生的波形文件

到此，读者可能还不满意，因为还想看看这个设计的波形，不用着急，这一节我们就讲如何使用 ModelSim 查看波形。当然，如果没有 ModelSim，就要使用其他波形查看器了。

我们首先修改一下 main.cpp 的设计。修改后的代码如下：

```
//By chenxi,all rights reserved
//main.h
#include <systemc.h>
#include "nand2.h"
#include "tb.h"
int sc_main(int, char**){
        sc_signal<bool> a,b,f;
        sc_clock clk("Clk",20,SC_NS);
        nand2 N2("Nand2");
      N2.A(a);
      N2.B(b);
      N2.F(f);
      tb tb1("tb");
       tb1.clk(clk);
       tb1.a(a);
      tb1.b(b);
       tb1.f(f);
    // trace file creation
```

```
    sc_trace_file *tf = sc_create_vcd_trace_file("Nand2");
    sc_trace(tf,N2.A, "A");
    sc_trace(tf,N2.B, "B");
    sc_trace(tf,N2.F, "F");
    sc_start(200);
    sc_close_vcd_trace_file(tf);
      return 0;
}
```

斜体部分是增加的代码，它们用于产生波形文件 Nand2.vcd，代码的含义我们将在后面的章节讲解。目前 SystemC 支持生成 VCD 或者 WIF 或者 ISDB 波形文件。

编译并运行新的代码，将会在 Debug 目录下产生 Nand2.vcd 文件，这就是我们需要的波形文件。

请读者注意 WIF 和 WLF 的区别。WIF 的中文意思是波形中间格式，是一种标准波形格式；而 WLF 是 ModeSim 所专有的波形日志文件。由于在 ModelSim 下只能打开 WLF 文件，所以推荐在 ModelSim 下看波形的办法是

- 新建一个 ModelSim 项目，将 Nand2.vcd 复制到该项目的目录下。
- 使用 ModelSim 行命令 vcd2wlf 将 VCD 文件转化为 WLF 文件。命令格式为

vcd2wlf <source.vcd> <target wlf>

这里我们输入：vcd2wlf Nand2.vcd Nand2.wlf。请读者注意 vcd2wlf 与 Nand2.vcd、Nand2.vc 与 Nand2.wlf 之间都有空格。如果没有任何提示，则命令被正确执行，这时将产生一个 Nand2.wlf 文件。

- 打开 signal、structure 和 wave 窗口。对应的 ModelSim 行命令为

view signals
view wave
view structure

- 在 wave 窗口下选择 Open dataset 菜单，将生成的 Nand2.wlf 文件导入进来。这时将在 structure 窗口中看到设计 SystemC:，在 signal 窗口中选择所要看的信号放到 wave 窗口中就可以看到波形了。所显示的波形如图 1.10 所示。

图 1.10　Nand2 设计的仿真波形

这里，我们假设读者对 ModelSim 有一定的了解。当然，还具备其他波形工具可以查看波形，如 Signalscan 和 Virsim，它们都可以直接查看 vcd 格式记录的

波形而不必进行格式转换。

另外，有一些免费的工具如 SystemC_win 可以编译和查看波形。关于工具的使用，第 10 章还将给予介绍。

习 题

1．简述硬件描述语言的发展史，总结复杂电子系统设计对系统级描述语言的要求。

2．比较 VHDL、Verilog HDL 与 SystemC 的优缺点。

3．编译并仿真 1.7 节～1.10 节中给出的 2 输入与非门的例子。

4．使用 SystemC 描述和仿真 4 输入与门。