可编程逻辑器件快速进阶丛书

VHDL与VerilogHDL比较学习及建模指导

随书附光盘一张

所有代码均可进行二次开发

郑亚民 董晓舟 编著

本书特色：

理论结合实践·从最核心处入手**对比讲解**，快速掌握2种语言的建模方法。

实用化的工程·本书所有工程代码都仿真验证通过，可**快速移植**于实际设计。

双语编写实例·所有综合实例均以VHDL和VerilogHDL**双语编写**，更易于对比学习。

总结经验技巧·让读者的学习过程更加高效，少走弯路，最快速成为**双语混编**高手。

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Verilog HDL比较学习及建模指导

郑亚民　董晓舟　编著

国防工业出版社
·北京·

内容简介

本书围绕 VHDL 和 Verilog HDL 两种硬件描述语言，系统介绍了相关的语法、技巧和计算机辅助设计软件，给出大量实例的综合、仿真结果和设计工程。本书的主要内容包括：VHDL 的基础知识和语法、Verilog HDL 的基础知识和语法、在 RTL 层次上利用这两种硬件描述语言进行实际建模的方法与技巧、实际设计中常见模块的实例设计，Synplify、ModelSim 和 Quartus Ⅱ等常用软件工具的使用方法。

本书内容新颖全面、叙述简明清晰、结构层次分明，利用大量的实例和图表说明问题，使读者易于接受。既可作为高年级本科生和研究生 EDA 设计方法相关课程的教材，也可以作为工程技术人员的参考资料。

本书附光盘一张，包含了书中所有设计实例的源程序和设计工程，可做二次开发。

图书在版编目(CIP)数据

VHDL 与 Verilog HDL 比较学习及建模指导/郑亚民，董晓舟编著. —北京：国防工业出版社，2008.6
(可编程逻辑器件快速进阶丛书)
ISBN 978-7-118-05779-9

Ⅰ.V... Ⅱ.①郑...②董... Ⅲ.①硬件描述语言，VHDL－程序设计②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(2008)第 079388 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)
北京诚信伟业印刷有限公司印刷
新华书店经售
*
开本 787×1092 1/16 印张 15½ 字数 356 千字
2008 年 6 月第 1 版第 1 次印刷 印数 1—4000 册 定价 33.00 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

终于完成了这一本《VHDL 和 Verilog HDL 比较学习及建模指导》的写作工作，本书第一次尝试着将 VHDL 和 Verilog HDL 这两种主流的硬件描述语言放在一起进行介绍，其中包含了 VHDL 和 Verilog HDL 的语法知识，以及如何使用这两种 HDL 进行逻辑设计。这是一本注重实用设计的书籍，作者希望通过本书向读者传递一种观点，无论是 VHDL 还是 Verilog HDL 都只是工具，最终的目的都是做出优秀的设计，因此应该灵活运用它们、发挥它们的优点并应用到实际中。

本书第 1 章概述了 EDA 和 HDL 的产生、现状以及未来发展。第 2 章介绍了业界常用的计算机辅助设计工具，如综合工具(Synplify Pro)、仿真工具(ModelSim)和 Altera 公司的综合开发平台 Quartus Ⅱ，掌握这些软件工具会对实际的设计有相当大的帮助，而且这些工具都是语言无关的。随后的 4 章用来介绍 VHDL 和 Verilog HDL 的基础知识，第 3、4 章独立介绍 VHDL、第 5、6 章用来介绍 Verilog HDL。这样做的目的是让初学者和已经掌握这两种语言其中之一的读者都可以根据自己学习的需要进行阅读，以便于可用最简便的方法实现建模。第 7 章主要介绍 RTL 建模的方法与技巧，第 8 章总结了日常设计中常见的一些综合性实例，这 2 章中所有的代码示例都以 VHDL 和 Verilog HDL 给出，配合完整的综合、仿真过程及设计工程，以方便读者学习。

本书适合于电子信息类高年级本科生以及研究生在学习 EDA 课程和 HDL 时学习参考，也可以作为工程技术人员的参考资料。

最后要感谢所有在本书的写作过程中支持和帮助过我们的同事与同学，没有他们就不会有今天这本书的出版。

由于作者水平有限，书中难免有许多不足之处，敬请读者批评指正。

编著者

2008 年 5 月

目　录

第1章 概 论

1.1 半导体工业

当今社会是信息化的社会，从科学的最前沿到人们的日常生活，各式各类的电子产品、信息处理设备完全融入了我们的生活。不可想象，如果没有计算机、没有互联网，世界会变成什么样子。半导体工业，这个为我们提供了所有这些电子产品的工业体系，在50年前还根本不存在，如今却拥有着万亿美元的销售额。在这个高速的增长过程中，计算机辅助设计起到了关键的作用，就像它在其他领域中带来的革命一样。

本书中简要介绍如何使用Verilog HDL和VHDL这两种标准的硬件描述语言去描述逻辑，建立实用的模型。这是整个庞大工业体系中关键性的技术之一，是非常基础和重要的。

1. 怎样的开端

在电子学发展的早期，真空电子管作为主要的元件被用于电子产品的制造。但是真空管体积大、不可靠以及耗电量大的缺点限制了电子产品的普及与应用。虽然美国宾夕法尼亚大学使用真空电子管制造了第一台电子计算器ENIAC（电子数字积分与计算器）用于计算炮弹弹道，并在后期的维纳斯计划中被用来计算原子弹的相关参数。但是其重达50t、占地3000平方英尺（1英尺=0.3048m）、需要19000只真空管，并且耗电量巨大。只有军方和大型研究机构才有条件来使用。

电子技术的的历程中，有两件事情为半导体工业的发展奠定了基础。其中之一是1847年肖克利的小组使用几片金箔片、一片半导体材料锗片和一只弯曲的纸架组成了世界上第一个晶体管（Transistor）。晶体管的名字取自“跨导”和“变阻器”两词，提供了与真空管相同的电功能，但具有固态的显著优点：尺寸小、无真空、可靠、质量小、最小的发热以及低功耗。

另一件事是1958年杰克·基尔比在美国德州仪器公司（TI）的实验室里制作了世界第一块集成电路，它包含5个电子原件，其中4个是晶体管。集成电路的发明使得工程师可以将所有的元件都集成在一块衬底上，从而能够设计更复杂的电子线路并且拥有更小的电路尺寸。

随着晶体管和集成电路的发明，电子学被注入了前所未有的生命力，电子产品的应用范围不断拓展，从航天、军事等高端领域到广播电视、便携式收音机等消费类产品。技术的创新和大规模的生产使电子产品的集成度不断提高、价格急剧下降，半导体工业进入全面的繁荣期。

2. 摩尔定律

提到半导体工业的高速发展，就不得不提到摩尔定律。1965年，仙童半导体公司研

发部经理，后成为英特尔公司首席员工的高登·摩尔在发表的文章中预言了一种发展趋势，即“单片上所用的晶体管数量每年都将翻番”。随着半导体技术的不断成熟，这种趋势又被修改为“单芯片上晶体管的数量每 18 个月翻番”，这就是著名的摩尔定律。

在摩尔定律被提出的年代，每块集成电路上的只能集成数十只晶体管，今天 Intel 公司的 Pentium 4 微处理器芯片上已拥有 4200 万只晶体管，摩尔定律确实带领着半导体工业以指数的速度增长了 50 年。但是近年来不断增加集成度带来的热噪和漏电问题日益严重，让人们开始质疑摩尔定律，质疑这种增长是否还将继续，是否会很快会达到饱和的状态。在这个问题上我们认为，没有必要过分的苛求一个定律的精确性，所有的规律都有它适用的范围，然而科学技术的进步却不会停步。不断逼近物理极限的过程或许能够让工程师们思考如何让晶体管工作得更有效率，在一定的数量下追求更高的性能，而不仅仅是数量的堆叠。即使物理极限无法逾越，量子计算机、碳纳米管、生物计算机等目前还在实验室中的新技术也必将为下一个“半导体”时代找到一个解决方案。

3. 设计与验证的挑战

怎样设计和验证一个包含数千万个晶体管的电路？如果当年的工程师还可以利用绘图板来手工设计的话，那么要设计有数万只晶体管的电路没有工具的辅助是不可想象的，这个工具就是计算机和与之配套的设计软件。计算机由于半导体工业的发展而变得越来越强大，反过来又促进了大规模集成电路的发展。

今天，大多数的数字 IC 设计通过硬件描述语言（HDL）来进行逻辑描述，计算机辅助工具将描述转化为网表（在 FPGA 中）或版图（在全定制 ASIC 中），并最终在芯片上完成物理实现。

1.2 电子设计自动化

电子设计自动化（EDA）是一个广泛的概念，作为所有使用计算机辅助电子设计的总称，它包含了 IC 设计、应用电路设计与仿真、印制电路板（PCB）版图设计等多个方面，甚至可以将嵌入式系统的软件开发也归入这个范畴中来。但是在目前流行的解释和各种文献、书籍中，EDA 只被用在 ASIC、FPGA 等微电子设计领域的自动化上。本节中也将沿用这种解释，介绍 EDA 的设计层次、流程以及相关的一些问题。

1.2.1 EDA 抽象(设计)层次

抽象是人们用来表示事物对象的基本方法，通过划分不同的层次进行描述允许我们以不同的细节程度来描述一个系统。例如读者面前的这本书，“书”是一个高级的表示层次，进一步细化后可以看到这是一本讲述硬件描述语言的书，再到纸张和油墨构成的几何图形（文字和图表），如果愿意甚至可以在分子级进行描述。随着描述层次的降低（加深），获取的信息量就越大，同样用以处理这些信息所要花费的工作量也就越大。

在 EDA 设计中，较完整的设计层次可以认为有 6 层：硅片级、电路级、门级、寄存器级、芯片级和系统级，见表 1-1。一个实际的设计可以使用任何层次来表示，当设计从上而下进行时，该设计就逐步接近物理实现，在表示上就更少一些抽象。所以，表示一个设计所需要的细节会随着它在层次中的下降而增加。在一个具体层次设计中，保证它

具有充分而不过多地细节是非常重要的。细节的不充分会造成不精确的结果，但是过多的细节则会使该层次的设计活动过多。

表 1-1　电子设计自动化的设计层次

层次	表示	基本部件
系统	性能规格说明(自然语言，如英语)	计算机/磁盘/部件/雷达
芯片	算法	微处理器/RAM/ROM/串行端口
寄存器	数据流	寄存器/ALU/计数器/多路器
门	布尔方程	与/或/异或/触发器
电路	微分方程	晶体管/电阻/电感/电容
版图/硅片	电子和空穴迁移方程	几何形状

1.2.2　EDA 设计流程

设计流程（Design—Flow）是一项设计从提出设想到最终测试完成所需要经历的过程，也就是指导开发人员如何一步一步构建出优秀设计的方法总和。和许多工程领域中一样，并不是所有的设计者都十分了解设计流程，并认为遍历所有流程是必要的。但对于初学者，首先了解一下业界通行的做法是有益无害的。

在 EDA 设计中，典型的设计流程主要包括以下几个方面。

1. 系统性能规格说明

系统性能规格说明是对系统模块如何划分以及各个模块特征的描述。作为一个项目管理中的词语，在大多数的公司中这个流程被称为总体设计或者方案设计。尽管名称不尽相同，但是设计者需要关注的事情是一样的。

系统规格说明需要描述的是系统模块如何划分，包括各个模块之间的端口特性、电气性能、工作电压、温度特性和频率要求等外部性能特征，但是不包含模块内部具体的实现方法。由于目前还没有一种专门的语言用来描述规格说明，设计者一般使用文字、图形混合表示的方法，或者直接使用 C 语言、伪代码（一种类似 C 或者其他标准语言的代码形式，只用来描述不能够被编译）和 VHDL 编写。

可以将系统性能规格说明看作是整个设计的蓝图、总体约束和任务分工表。说明的制定者将保证提出的规格是可实现的而且能够满足用户的需求；设计者根据规格说明设计每一个系统模块并最终组合出整个系统。随着今天 EDA 设计的复杂性不断增加，规格说明中不仅要包含电路完整的特性描述，而且还要包含如何确定各种要求的测量方法和检验方法，即系统的可测试性要求。

2. 算法设计

算法设计就是使用加号、减号、布尔运算符、过程控制语句（if-then-else 等）和存储逻辑来描述设计的结构。在这个流程中关注的是功能，不对具体的某一个运算分配硬件资源，也就是不关心各个运算的实现方法与效率。

算法设计的描述与软件程序非常相似，可以使用 C 语言或其他的标准语言编写，也可以使用 VHDL 或 Verilog HDL 等硬件描述语言。具体选择哪种语言主要取决于验证的

方法以及如何更好的和寄存器传输级描述相衔接。

使用 C 编译器可以将 C 语言编写的算法逻辑直接编译生成计算机程序，因而能够更快更方便的验证设计，而且验证速度比使用仿真器验证硬件描述语言快很多。使用 VHDL 和 Verilog HDL 来描述算法的优势在于可以方便的将描述转换到寄存器级，从而和下一设计流程结合的更紧密，目前大多数的 EDA 工具都支持这一自动转换功能。

3. 寄存器传输级描述

算法设计中只对逻辑功能进行了描述，在寄存器传输级将为算法描述中的各种运算符、条件语句和存储器分配时钟周期以及数字系统中的同步或异步工作方式。

这里以 16 位乘法器举例说明。在数字电路中，做一次乘法需要执行一串加法，使用串行加法只需要一个加法器就够了，但是要花费 16 个时钟周期（速度慢）；并行加法器可以在一个周期内完成运算，但是需要 16 个加法器同时工作，因而消耗更多的芯片面积（耗资源）。这其中的权衡与选择取决于设计对速度与面积的要求，清单 1-1 中对比了 16 位串行乘法器在算法和寄存器传输级（RTL）的描述。

清单 1-1 算法描述与寄存器传输级描述的比较

算法描述（C 语言）

```
C = A * B;
```

寄存器传输级描述（VHDL）

```
-- 串行 16 位乘法器
-- C = A * B
process
variable var_ACCU    : std_logic_vector(31 downto 0); -- 累加器
variable var_CNT     : integer range 16 downto 0;     -- 计数器
begin
 if CLK'event and CLK = '1';
     var_ACCU(31 downto 0) := var_ACCU(30 downto 0) & '0'; -- 移位
     if B(var_CNT) = '1' then
          Var_ACCU(31 downto 0) := var_ACCU(31 downto 0) + A; -- 加数
     end if;
     if RESET = '1' or var_CNT = 0 then   -- 复位或计数器为 0
          C <= var_ACCU;          -- 输出运算结果
          var_CNT := 15; -- 计数器初始化
          var_ACCU := (others => '0');    -- 累加器初始化
     else
          Var_CNT := var_CNT – 1;  -- 计数器执行减法计数
     end if;
end process;
```

注意 本书代码清单中，所有 VHDL 关键字（语法保留字）均用黑体表示。

算法设计更关心设计的功能，所以描述非常简洁，而寄存器传输级的描述则在实现乘法器的方法与时序方面提供了更多的细节。

4. 逻辑设计

如果将寄存器级描述进一步细化便可以得到布尔表达式。但是现代的设计中很少进行完全的布尔表达式转换，这是因为绝大多数的设计都是基于现成的库（Library）进行的，工艺提供商（为项目做掩膜并制造芯片的厂商）会在库中包含触发器、门电路、存储器等经过验证的逻辑单元供开发者选择使用。使用 EDA 软件可以在开发者的控制下半自动的将寄存器传输级的描述转换为这些单元之间的连接关系，称之为网表（Netlist），从而更高效地完成逻辑设计。

需要注意的是，用库的设计是和工艺提供商密不可分的，如果工艺提供商只提供逻辑门一级的库（细粒度库），则设计者就必须将设计完全转换为各种逻辑门间的连接。如果库中包含了触发器、运算电路、存储器等高级单元（粗粒度库），所需要做的转换工作就会减少。库的抽象层次越高，在逻辑设计流程所要做的细化工作就越少。

5. 晶体管设计与互连

晶体管设计的目的是将逻辑描述转换为芯片上晶体管的尺寸与互连。在 CMOS 工艺中晶体管的尺寸主要包括沟道长度、沟道宽度和氧化层厚度，它们决定了晶体管的特性曲线，即晶体管的性能。互连指的是连接晶体管之间的数层金属，它将晶体管彼此连接在一起构成电路。在实际设计中，只有全定制 ASIC 的设计者才需要进行这一级别的设计。使用标准单元和门阵列设计 ASIC 都用不着优化晶体管，而是使用预开发（库）的元器件直接构建互连就可以了。

6. 版图设计

在确定晶体管尺寸和相互连接之后，就可以转换为几何图形（晶体管都是在硅片上一层层通过掩膜制备的，几何图形用于进行掩膜），目的是找出晶体管及其互连在硅片上最合适的几何排列。所谓合适，一般指的是占用面积最小的排列，因为芯片的成本是和面积直接相关的，但有时也需要综合考虑性能和物理上的因素。

全定制 ASIC 的版图只用于本身的设计，多用于模拟单元的加工，包含完整的晶体管尺寸信息。使用库设计的半定制 ASIC 版图是建立在工艺提供商提供的外形上的，轮廓只包含标准单元的轮廓和连接点，内部几何尺寸因为版权的原因而被保护。现在的版图设计中一般采用的方法是在用户预定义的各种约束下由布局布线工具半自动地生成芯片版图。

7. 验证与测试

现在的设计正变得越来越复杂。不用说 CPU 或者 DSP，仅仅一个 32 位宽度的二进制乘法器，如果要仿真出每一种输入可能性的话，需要 2^{64}= 18446744073709551616 种仿真激励。即使仿真器每秒可以处理 1 亿次乘法计算，也需要 5849 年，生成的相关数据更可能要以 TB 计算，这显然是不可想象的。

因此，人们在仿真激励的选择上付出了巨大的努力，试图使用尽量少的仿真激励覆盖尽量多的出错可能性。这种激励的选择与仿真结果的分析都会导致高昂的费用，以至于一个典型的 ASIC 项目中验证的费用远比设计的费用高。

比较理想化的验证方法是在每一步设计流程后进行验证，这种验证是穿插在上面叙述的设计流程之间的。设计人员力图在每一个设计层次中通过仿真与必要的回溯将出错的可能减少到最低，而不是最终面对一个多层的复杂系统。

1.2.3 自顶向下还是自底向上

自顶向下的设计流程是首先把设计分解成模块，然后描述这些模块之间的接口和连接。这时的设计者并不清楚这些模块的内容，只把模块当作黑盒子对待。在接下来的步骤中才对其内容进行描述，同时也可能插入新的黑盒子。到了可以使用已知的元件，如逻辑门或晶体管来描述所有黑盒子的时候，描述就趋于完整了。

传统的自底向上设计方法是首先在最低层描述门和晶体管的类型元件，由此逐步构建越来越复杂的框图，直至最终完成芯片设计。如果系统的特性和描述比较简单且容易识别，采用自底向下的设计方法可以避免面对未知的黑盒子工作，使工程师更容易把握细节，这在经验至关重要的模拟电路设计中是非常关键的，所以在模拟芯片的设计中仍然采用这种设计方式。

现代的设计十分强调可重用性，让编写的模块可以被多个设计使用。芯片设计都是在丰富的设计资源和各种各样的库的支持下进行的，工程师并不需要从算法一直做到晶体管，借助 EDA 软件的强大功能，人工设计工作越来越多的被限制在寄存器传输级。因而自顶向下的设计方法的越来越广泛应用。

我们认为，具体采用什么样的设计方法应该取决于设计本身。自顶向下的设计优势是要在各个层次，尤其是低层次上拥有丰富的设计资源的情况下才能体现的。而这些设计资源的产生往往都源自早期自低向上的设计成果。无论是自底向上还是自顶向下，又或者两者的结合，都是为做出优秀的设计而服务的。

1.2.4 EDA 技术的发展

1. 第一代 EDA

普遍认为 EDA 的发展开始于 20 世纪的 70 年代。在杰克·基尔比发明集成电路的后几年，市场上出现了 20 个～200 个晶体管的集成电路。这时的逻辑和版图设计工作完全由手工完成，多边形由透明膜按照（1:1）感光的方式投射到硅片上，经验和估算在整个设计过程中起了至关重要的作用。电路性能由于晶体管固有的非线性而变得很难把握，往往需要通过制造许多样品多次重复设计来改善。

当著名的 SPICE 在加州大学伯克利分校的研制成功，工程师可以无需实际制造便可以对电路进行优化，而且成本很低，大大的提高了 EDA 的设计效率。直至今天 SPICE 的后续版本仍用于仿真如晶体管这类非线性元件的电气网络并分析时间和频率响应。

在机械制造和建筑工程中被首先应用的计算机辅助设计（CAD）系统，很快就被用于代替繁琐的光学薄膜制作掩膜。这时可以在有鼠标的计算机上绘制几何图形，满意后将计算机存储的结构图形转换成物理版图，进一步促进了 EDA 的发展

第一代 EDA 中实际还没有一种工具可以用于自动化设计过程，因此 EDA 的名称尚不准确，只有用于重复结构自动复制的版图编辑器在今天的模拟电路设计中仍有使用。

2. 第二代 EDA

到了 20 世纪 80 年代，第二代 EDA 已经允许工程师舍弃逻辑门的设计工作，单元库从晶体管级提高到了门电路级。在单元库大量应用的基础上，自动化的布局布线工具使得版图制作的自动化设计程度大大提高，高性能的计算机可以根据网表格式的电路数据

输入确定布线方法。

随着集成度的提高，在验证领域使用 SPICE 完全分析所有电路特性需要花费很长的时间和大量的计算资源。所以这一时期推出的仿真工具开始简化仿真分析功能，但是强化了仿真时间响应和简单逻辑状态验证这两种关键应用，从而使得设计验证工作更加经济有效。

3. 第三代 EDA

第三代 EDA 的时间跨度从 20 世纪 80 年代直至几天，并以 HDL 的产生和逻辑综合的逐渐成熟为标志。

这里尤其要提到的是 IEEE 在 1987 年通过的 VHDL 标准，以及后来对 Verilog HDL 的标准化，这对 EDA 的发展具有重大的意义。这些标准化工作最初的目的是把这些语言用于单元模块的统一，但是很快便被广泛应用在了整个复杂数字系统的描述上。这些语言的标准化在一定程度上推动了高级语言结构的自动翻译，并促成了逻辑综合工具的产生。

逻辑综合的成熟是第三代 EDA 的另一个标志性事件。而提到逻辑综合就不得不提到在 20 世纪 80 年代末将第一个完整的逻辑综合工具 Design Compiler 投入市场的著名的 EDA 厂商——Synopsys 公司。虽然在初期并没有得到认可，但是随着性能的不断改进，“硬件描述语言+逻辑综合工具”的搭配很快就成为了 IC 设计中极广泛的设计模式并一直沿用至今。

可以这样认为，第三代 EDA 给设计人员提供了一套在寄存器传输级设计并自动转换到门电路级的完整工具。而在今天的工程实践中，没有这些工具依然不行。

4. 未来的 EDA

40 年来电子工业的飞速发展使 EDA 从概念走到了现实，随着半导体的结构越来越小、复杂度不断提高，EDA 必将不断发展并起到至关重要的作用。

在未来的 EDA 中，设计重用会进一步发展成为复杂设计的基础之一。设计重用就是重复利用设计。在进行一项复杂产品设计时，可以在授权下采用一些来自其他公司的标准化的、经过验证的模块来搭建系统，从而大大的加快新产品的上市时间。IP Core（知识产权核心）技术就是这种趋势的体现并已经得到了一定程度上的应用。

行为级综合是逻辑综合的发展，寄存器传输级的综合势必要向更高的抽象级别延伸。硬件描述语言也将更加贴近高级语言以适应更高的描述级别。

微控制器和微处理器技术的发展使很多电气系统可以由软件进行控制。软件具有更大的灵活性而且更便宜，硬件则拥有更高的速度。在一个系统中软硬件的分配就影响了系统的费用与性能。随着 SOC 概念的实用化，将整个系统集成在一枚芯片上就要求软件、硬件的互补、协同设计。

1.3　HDL 基础

1.3.1　HDL 的产生与发展

在数字电路设计初期，工程师习惯于首先使用卡诺图简化设计，然后在面板上进行

验证。随着设计越来越复杂，这种方法开始令工程师们濒临崩溃（想象一下项目组长让你在面包板上搭出一个 CPU 时你的感受）。于是借助计算机进行电子设计（EDA）的方法得到了广泛的应用：首先使用原理图来描述系统，然后在 EDA 工具的协助下通过预先设计好的元件库中的元件来完成设计。但是问题很快又出现了，各个公司之间，甚至同一家公司不同的项目组之间使用的元件库相都互不兼容，原理图的移植和重用变得十分困难，维护起来也费时费力。工程师们开始寻找一种与平台无关、灵活方便并可以在多个设计层次进行描述的设计工具，HDL 应运而生。

HDL 允许工程师将数字电路系统从抽象到具体逐层逐次的进行描述，然后借助综合工具将较高的描述层次自动转换到门级网表，并最终在 ASIC 或其他平台上实现。HDL 的出现大大推动了电子设计自动化的发展，并为今天的 IC 设计方法奠定了基础。据统计，硅谷 90%以上的 ASIC 和 FPGA 采用 HDL 进行设计。

目前，业界主要的 HDL 有：VHDL、Verilog HDL、Superlog、SystemVerilog、SystemC、Cynlib 等数种。其中 VHDL 和 Verilog HDL 作为 IEEE 标准得到了最广泛的支持与应用，Superlog、SystemC 等新生代 HDL 也在很多新兴领域中占有一席之地。为了更加直观的了解这些语言之间的区别，图 1-1 中将这些语言所支持的描述层次作了归纳。

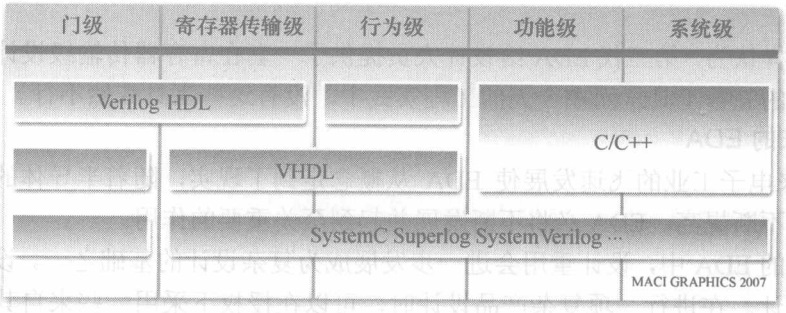

图 1-1 不同 HDL 的描述层次关系

从图 1-1 中可以看出，VHDL 和 Verilog HDL 在底层描述尤其是寄存器传输级描述中占据了主导地位，而在功能级和系统级的设计与验证中则以 C/C++作为补充。随着设计抽象层次的提高，设计复杂度增加带来的仿真验证上的挑战，这促成了 SystemC、Superlog 等新兴 HDL 的语法结构更加丰富，在系统级、功能级等高层次的设计描述和仿真中更有优势。

1.3.2 基于 HDL 的设计流程

HDL 的设计流程规范的是如何使用 HDL 进行系统描述并最后在硬件平台上实现的步骤，它与类似于 EDA 设计流程但是并不相同，可以认为是 EDA 设计流程的一种实例化。图 1-2 中从设计和验证两个方面分析了典型的 HDL 设计流程，其中设计区域中的所有步骤是必须遍历的，而验证区域中的步骤可能因设计复杂性的不同而省略。