嵌入式系统开发技术丛书

ASIC 芯片设计从实践到提高

池雅庆　廖　峰　刘　毅　编著
杨银堂　主审

中国电力出版社
www.infopower.com.cn

内 容 简 介

本书结合具体的实例，对 ASIC 芯片设计开发的整个流程及各个阶段所使用的 EDA 工具进行了系统地介绍。全书共 8 章，分为 4 个部分：第 1 部分介绍了 ASIC 的基础知识和目前广泛应用的 EDA 工具；第 2 部分分别介绍了两种常用的集成电路设计语言 Verilog HDL 和 VHDL；第 3 部分详细地阐述了 ASIC 设计的仿真验证及综合技术；第 4 部分主要介绍了布局布线技术。

本书既阐述了 ASIC 设计的基本理论，又结合实际的工程项目，给出了一些具体的例子，适合初学 ASIC 开发的技术人员阅读，也可作为高等院校相关专业本科生和研究生学习 ASIC 系统开发的参考书。

图书在版编目（CIP）数据

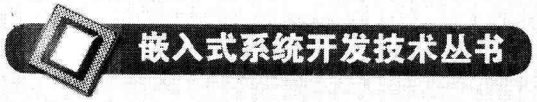
ASIC芯片设计从实践到提高 / 池雅庆，廖峰，刘毅编著.—北京：中国电力出版社，2007.6
（嵌入式系统开发技术丛书）
ISBN 978-7-5083-5378-4

Ⅰ. A… Ⅱ. ①池…②廖…③刘… Ⅲ. 集成电路 - 电路设计 Ⅳ. TN402

中国版本图书馆CIP数据核字（2007）第053922号

责任编辑：夏华香
责任校对：崔燕菊
责任印制：李文志

书　　名：ASIC 芯片设计从实践到提高
编　　著：池雅庆　廖峰　刘毅
出版发行：中国电力出版社
　　地址：北京市三里河路 6 号　邮政编码：100044
　　电话：（010）68362602　　传真：（010）68316497
印　　刷：航远印刷有限公司
开本尺寸：185 × 260　　印　张：15.5　　字　数：372 千字
书　　号：ISBN 978-7-5083-5378-4
版　　次：2007 年 6 月北京第 1 版
印　　次：2007 年 6 月第 1 次印刷
印　　数：0001—4000
定　　价：26.00 元



丛书序

这是一个令人激动的时代，嵌入式系统的应用深入到了整个社会。环顾四周，总会发现嵌入式系统正在深刻地改变着我们的生活。十年前，一个数字寻呼机就能彰显用户的身份，而如今，手机已经成为我们生活的必备品，不仅能打电话和发短信，还能拍照片、玩游戏、看电影；十年前，我们得冲洗胶卷后才能看到拍摄的相片，发现拍摄的失误时往往已经为时已晚，十年后的今天，数码相机已经深入到每个家庭，它不仅可以让我们马上看见拍摄的照片，而且不使用胶卷，数字信息也更易于传递。这一切的变化，都得益于嵌入式系统发展所起的举足轻重的作用。

关于嵌入式系统

嵌入式系统是一种专用的计算机系统，它根据应用要求，把相应的计算机直接嵌入到应用系统中。嵌入式系统涉及到了当前信息技术最新成果的方方面面，融合了计算机软/硬件技术、通信技术、半导体微电子技术等，设计与制造相当不易。嵌入式系统的重要性与日俱增，而这方面的人才却十分紧缺。

丛书的内容

本丛书包括ARM系统开发、ASIC芯片设计、嵌入式系统无线互联、FPGA应用开发、SoC系统开发和DSP应用开发6个专题，共分6本书。每本书以一个或几个案例为基础展开，并提供了大量的源代码，便于读者学习和使用。

丛书特点

在编写本丛书的过程中，力求以简明扼要的语言，重点突出地描述清楚基本概念和开发流程。在本丛书的内容中，融入了作者以往的研发经验和科研工作中的实例。通过案例分析与设计，逐步完成某个完整嵌入式系统的设计。案例剖析不仅能够提高读者的阅读兴趣，克服对复杂问题的恐惧心理，而且能够使设计思想与设计过程更容易理解，帮助读者尽快上手训练。

作者优势

我们组织了在嵌入式领域具有长期开发经验的研究人员与工程师编写了本套丛书，他们都有丰富的电子产品研发编程经验，在专业期刊上发表过很多学术论文，在实际开发过程中积累了丰富的项目实践经验，相信他们提供的应用方法和技巧能有效地帮助读者提高实际操作能力。

读者对象

本丛书可作为有关科学研究与产品开发人员的工作学习参考书，也可作为高等院校相关专业本科生与研究生的教学参考书。

其他声明

尽管作者做了很大努力，但限于水平和时间，错误和不妥之处在所难免，敬请读者批评指正，我们的联系方式是 liu_chi@cepp.com.cn。同时希望各行业从事嵌入式系统及相关技术工作的专家、学者、工程技术人员借此机会积极参与图书的选题开发和编写工作，将您在工作实践中获得的丰富经验总结出来，共同推进我国嵌入式系统技术的发展！

丛书编委会

2007 年 5 月

前　言

从20世纪50年代末开始，电子技术的发展进入了集成电路发展的新时期，随着集成电路技术的发展，无论是在通信、汽车电子、自动化控制系统，还是在一般的仪器仪表、玩具电路等领域，集成电路都得到了广泛地应用，集成电路技术已成为现代高科技发展的基础。在这种情况下，专用集成电路（ASIC）技术应运而生，专用集成电路能够降低设备的价格和功耗，提高设备可靠性，减小体积，减轻重量，它为电子设备和电子线路的设计方法提出了重大的变革。近几年来，专用集成电路在国外发展很快，在国内也受到了很大程度的重视，国家也在不断地加大投入，培养专用集成电路设计人员。专用集成电路的设计已成为电子系统设计人员必须掌握的知识。

本书基于一个实际项目的开发，涵盖了ASIC的前端设计、功能验证、综合仿真以及最后的版图设计、布局布线等技术，让读者对ASIC设计的整个流程有一个整体的认识，并对广泛使用的EDA工具及其使用方法进行了比较详细的叙述，希望对进行ASIC设计的技术人员和广大学生有一定程度的启发。

本书深入浅出，通俗易懂，理论与实践紧密结合，适合作为ASIC的初学者进行ASIC设计的入门指导书，也适合作为一般的工程技术人员进行设计时的参考书。

本书得以出版要感谢西安电子科技大学副校长、博士生导师杨银堂教授的大力支持。全书由池雅庆博士主编，廖峰和刘毅教授编写，李建伟、王青松和郭晋亮参与了本书的统稿工作，杨银堂教授对本书的技术部分进行了审校，在此一并致谢。

限于作者水平和时间，书中难免存在疏漏之处，敬请广大读者批评指正。

作　者

2006年3月于西安

目　录

第5章 ▸▸

5 验证技术

验证在设计中占有很重要的地位，从设计流程中可以看到，设计工作的每一步几乎都要进行验证。对于今天上百万门的大规模 ASIC，可重用的 IP 核以及片上系统（SoC）的设计，随着工艺水平的不断提高、集成度的不断增大以及速度的不断增加，ASIC 芯片的设计变得越来越复杂，同时也对验证提出了更高的要求，验证的工作量和时间所占的比例也越来越大，在目前超大规模 ASIC 设计的整个过程中，验证占用了整个设计的 70%左右的时间。对于一个设计团队来讲，验证工程师的人数通常是设计工程师的两倍。

要做好对设计的验证，在验证之前应提出切实可行的验证计划，确定采用的验证手段和验证策略。从是否需要执行待验证设计（Design Under Verification，DUV）的角度，验证可以分为动态验证和静态验证两大类；按照验证的目的可以将验证分为意图测试、压力测试、回归测试等。

5.1 验证的策略和手段

验证已经成为在整个设计流程中成本以及推向市场的时间的一个最大的瓶颈。能否采用合理的验证策略和手段已经成为设计是否成功的关键问题。

5.1.1 验证流程

图 5-1 显示了整个验证的流程。规格书和验证计划书把整个流程分成 3 个主要部分：决定验证策略；产生测试平台、运行测试和纠错；回归测试和各项测试目标的覆盖。

5.1.2 验证的策略

1．测试用例（Testcase）

测试用例反映了芯片在实际工作过程中可能遇到的情况，以及芯片在这些情况下应有的工作状态。设计测试用例的目的在于定义出在实际工作中芯片所能遇到的激励，从而在仿真过程中能够模拟出来，检查设计是否实现了相应的功能。设计测试用例的关键是确定需要验证哪些特性。根据设计需求，定义出所有需要验证的特性，然后设计相应的测试用例。在这个步骤，测试用例的涵盖范围直接决定了验证过程的功能覆盖率，决定了设计最

终能否很好地满足系统的需求。

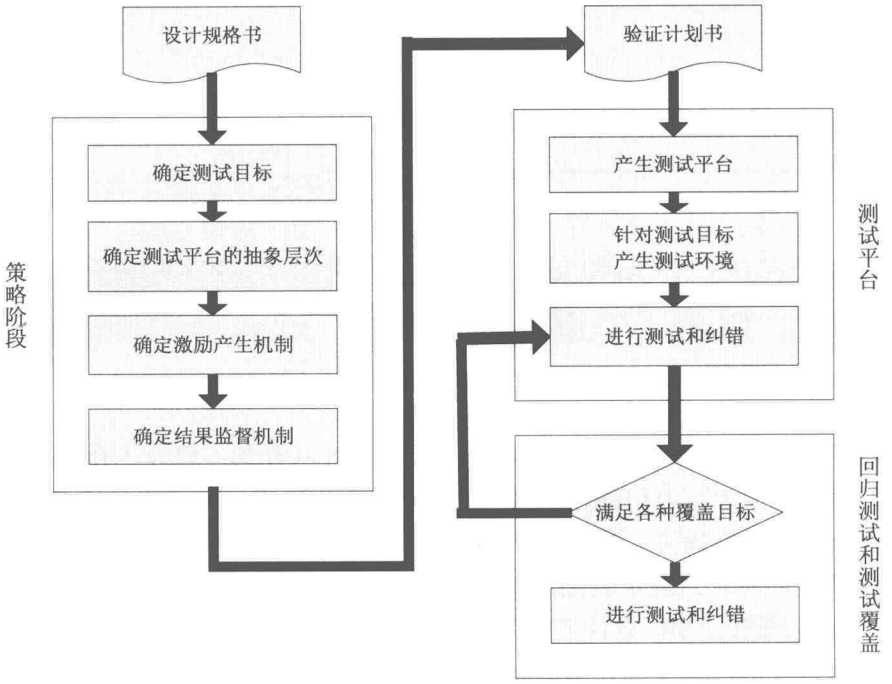


图 5-1

在实际开发过程中，测试用例通常有 4 种：接口测试用例、子模块测试用例、系统测试用例和随机测试用例。接口测试用例是针对接口的时序，检查模块的接口时序是否符合要求。子模块测试用例是针对各个子模块的功能，检查子模块是否完成其所应该实现的功能。系统测试用例是对整个系统进行测试，检查系统的功能和效率。随机测试用例并不针对某一具体功能，其主要目的是产生随机序列，针对系统的稳定性进行重点测试，提高功能验证的覆盖水平。

随机测试用例通常是在系统验证的最后阶段使用。由于它能够产生设计人员意料之外的特殊情况，并使子系统中相互独立的功能之间出现较为复杂的关系，因此常常能够找出系统存在的很多问题。但是由于随机测试实现起来比较困难，因而一般都只是将其应用到低层次的验证上，许多小模块的验证可以利用单一的随机测试环境。

每一个测试用例通常包含两个部分：测试点和测试数据。测试点定义了待测的特性；测试数据定义了完成该测试需要使用的激励数据和标准的响应输出数据。

2．测试平台的抽象层次

测试平台的抽象层次决定了其是否能处理从字节到数据包甚至到更高级别的数据类型。高的抽象层次在继承、多态性和向下兼容低级抽象层次上有优势。

提高抽象层次不仅仅是对数据进行封装处理，更重要的是对测试对象的封装，如采用总线功能模型（Bus Function Model，BFM）等，对于包含复杂的总线结构等的设计有非常重要的意义。

在实际应用中，提高测试的抽象层次不一定是最有效、合理的。较高的抽象层次一般是用来对设计进行一个粗略的验证。在较高的抽象层次上进行验证，会降低对时序以及激

励和响应之间协调性的细节控制，如果要在具体的测试目标上进行细节控制，就要在较低的抽象层次上进行验证。

3．验证环境

对于一个系统的验证，验证环境的建立也是非常重要的。验证环境是指对设计进行验证时为了便于使用、管理和协同工作而设计的一个环境。在该环境下，能启动验证所需要的一系列工具，并且可以对设计的仿真结果进行整理、分类和组合。验证环境真实地模拟芯片在实际工作过程中所处的环境。良好的验证环境是提高验证效率的关键，要求具有较高的自动化程度和较快的错误定位机制。

4．产生激励和结果比较

这里主要是考虑测试平台如何产生激励和如何处理结果。产生激励是比较简单的，产生激励的时候可以对它的时序和内容进行完整地控制，困难的是对响应的验证。在验证之前要计划好如何确定期望响应，然后如何验证拥有期望响应的设计。如果有一系列的参考输出和验证的输出做比较，或者能尽早地发现问题，都将会提高验证的效率。

5.1.3　验证的手段

1．动态验证

动态验证是指使用软件或硬件对设计的行为进行模拟和评估，其核心是对时间序列的模拟，所以必须要有验证矢量序列来驱动 DUV。

动态验证包括模拟验证与仿真验证两大类。模拟验证通过计算机软件对硬件设计的行为进行某种程度的近似和简化，模拟出硬件应有的行为，并对行为进行评估。仿真验证则是通过硬件平台对设计进行近似。由于硬件并行运行的速度远大于软件串行的模拟速度，通常仿真会比模拟快 2 到 3 个数量级。但是软件与硬件的交互障碍等困难限制了仿真的应用。

对于模拟验证，根据对 DUV 内部数据的可观察性和可控制性的程度可以将其分为白盒测试、黑盒测试和灰盒测试技术三大类。

（1）白盒测试

白盒测试也称结构测试或逻辑驱动测试，它知道设计内部的工作过程，可通过测试来检测设计内部操作是否按照规格说明书的规定正常进行。按照设计内部的结构测试设计，检验设计中的每条通路是否都能按预定要求正确工作，而不顾它的功能。白盒测试的主要方法有逻辑驱动、基本路径测试等。

白盒测试全面了解程序内部的逻辑结构并对所有逻辑路径进行测试。白盒测试是穷举路径测试。在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。贯穿程序的独立路径数是天文数字。但即使每条路径都测试了仍然可能有错误。主要原因如下：

- 穷举路径测试决不能查出程序违反了设计规范，即程序本身是个错误的程序。
- 穷举路径测试不可能查出程序中因遗漏路径而出错。
- 穷举路径测试可能发现不了一些与数据相关的错误。

（2）黑盒测试

黑盒测试也称功能测试或数据驱动测试，它是在已知设计所应具有的功能的前提下，

通过测试来检测每个功能是否都能正常使用，在测试时，把 DUV 看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，测试者在 DUV 接口进行测试，它只检查 DUV 功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息，并且保持外部信息的完整性。黑盒测试的方法主要有等价类划分、边界值分析、因果图、错误推测等。

黑盒测试着眼于 HDL 程序外部结构，不考虑内部逻辑结构，针对功能进行测试。黑盒测试是穷举输入测试，只有把所有可能的输入都作为测试情况使用，才能以这种方法查出程序中的所有错误。实际上测试情况有无穷多个，人们不仅要测试所有合法的输入，而且还要对那些不合法但是可能的输入进行测试。黑盒测试通常有下列缺陷：

- 规格书可能不完备，特别是设计的角落部分。
- 很难检查设计的所有输入。
- 难以验证设计所有输出的行为。
- 如果输出有错误，很难确定错误的源头。
- 某一些类型的缺陷很难被检测出来。

（3）灰盒测试

灰盒测试是白盒测试与黑盒测试的混合产物。它不像黑盒测试一样对 DUV 内部一无所知，也不像白盒测试一样完全知道 DUV 的内部信息。通过对关键信号的提取，灰盒测试可以提高验证的效率。

2．静态验证

由于动态验证对资源的需求很大，因此动态验证技术无法胜任大规模集成电路验证。解决这个问题的关键是在 RTL 级采用快速的功能验证，而在门级采用静态验证。静态验证的方法是把时序验证和逻辑功能验证分开来做，不需要加任何激励。这样做可以极大地节省验证时间。静态验证包括形式验证（Formal Verification）和静态时序分析（Static Timing Analysis，STA）两种。

形式验证是用数学的方法证明设计的正确性，常用 3 种方法：理论证明技术、等价性检验（Equivalence Checking）和模型检验（Model Checking）。由于形式验证采用静态的、数学的方法，因而形式验证不需要测试向量，形式验证方法要比仿真更快、更精确。形式验证不考虑设计电路的时序问题，它只比较两个电路的功能是否一致。

理论证明技术仍然处于学术研究阶段。设计者可以通过建立设计行为的模型并通过理论论证其满足功能要求。

模型检验也称属性检验（Property Checking），该技术通过数学模型来验证设计的行为属性。模型检测工具能够自动对设计的行为与设计者定义的一套逻辑属性进行比较。属性的定义是从设计规范提取而来的。该技术非常适合验证复杂的控制结构，例如总线仲裁器、解码器、处理器与外设的桥电路等。

等价性检验对同一逻辑设计的两种不同的表现形式进行等价性比较。它用数学方法对参考设计和修改过的设计做等价性检查，常常用于寄存器传输级与寄存器传输级（RTL-RTL）、寄存器传输级与门级（RTL-Gate）、门级与门级（Gate-Gate）间的等价性检查。因为该技术是通过将目标设计与参考设计比较来验证目标设计功能是否正确，所以该技术的关键是要保证参考设计的功能正确性。

许多 EDA 公司已开始提供等价性检验工具，如 Synopsys 公司的 Formality。Formality 比较两个设计时，将它们读入存储器，然后在它们的数据结构上使用形式化的数学算法。如果它们有相同的同步功能和状态保持器件（寄存器和锁存器），就可以成功地比较。如果两个电路在各个输出引脚和每个寄存器和锁存器上的功能是相同的，就认为这两个电路是等价的。

等价性检验的最大优点是它以 RTL 作为设计的基本参照（Reference），而不管对最后的网表做过什么样的改动。即便电路的功能在设计的最后一分钟通过编辑网表改变，只要将相同的改变反映到 RTL 中，依然可以用形式化方法验证 RTL 和最终网表的等价性。

形式验证可以用来检查初始的 RTL 描述与下述设计的等价性：

- 综合后的网表。
- 插入测试逻辑的网表。最直接的是插入扫描链，如插入片上JTAG结构。
- 时钟树插入和版图设计完成后的网表。这需要比较层次化的RTL和扁平化的网表。
- 手工编辑后的网表。通常工程师会在最后做一些手工编辑来改变性能、可测性或功能。

设计中有些错误只有在特定的事件序列发生时才能表现出来，这些错误隐藏很深，如果不能在设计的早期检测到并消除该类错误，将会严重影响设计开发周期。形式验证技术能够在设计的早期有效检测这些模糊的错误。该方法不要求验证平台和测试向量，具有很快的验证速度和 100%的覆盖率，通过错误分析报告，比较容易找到出错处。但是该技术并不对设计的时序进行检查，因此必须与时序分析工具结合使用。

对于大规模的设计而言，门级和 RTL 之间的形式验证实在是太慢了，特别是如果要多次迭代。在这种情况下，最好是在 RTL 和门级网表之间使用一次形式验证，然后用门级网表作为以后迭代的基本参照。例如，可以在时钟树插入的前后使用等价性检验，等价性检验的算法在进行门与门之间的比较时的效率要远远高于门与 RTL 之间比较的效率。

静态验证的另一个技术是静态时序分析。静态时序分析是分析、诊断和确认一个设计的时序特性的彻底方法。它将整个电路分解为一组组路径，然后分别计算电路中每条路径的延时，并对照时序约束检查任何可能的时序违反情况。静态时序分析检查的重点是时序单元的建立和保持时间。静态时序分析不检查电路的逻辑功能，所以，在做完电路的时序验证后，还必须做形式验证。

由于静态时序分析不需要输入测试向量，因此它比传统的仿真技术要快好几个数量级，适用于较大规模的设计，它可以辨别出设计中所有的关键路径（Critical Path），但其所辨认出的关键路径中可能存在着伪路径（False Path）。静态时序分析只适用于同步电路设计。

相比之下，动态验证不局限于同步电路，且验证的结果比较准确，不会辨别出伪路径，但其不足在于速度较慢，且有可能遗漏某些关键路径，因为仿真技术在辨别关键路径时严重依赖于输入的测试向量。因此，在当前较大的设计中，一般要同时使用动态和静态验证技术，以尽可能保证设计的可靠性。

3．其他手段

按照验证的目的可以将验证分为意图测试、压力测试、回归测试等几类。

意图测试是指使设计运行在正常模式下完成正常功能的测试。意图测试占据验证工作的大部分工作。任何规格书中标注的设计功能都要经过意图测试。

压力测试是指使设计运行在极端负荷情况下，考验设计的处理能力的测试。比如使 FIFO

（First In First Out）运行在满的情况下，同时写入读出。使总线运行在高负荷下也是压力测试的一例。

回归测试是指运行以保证已经验证的问题不会再现的测试。在测试阶段通常会发现缺陷，然后设计工程师对发现的缺陷进行修改。然而在修改新发现的缺陷时可能导入新的缺陷，或者重新激活原来的缺陷。回归测试的把关可以保证设计的修改不会导致原来的缺陷被激活。但是在实践中回归测试可能被轻视。设计工程师对自己的自信、对修改缺陷的工作的轻视都有可能使得被修改的设计跳过回归测试，特别是当设计工程师测试自己的设计时。一个好的解决办法是通过验证工程师的严格把关，杜绝回归测试逃逸现象。

5.2 构建测试平台（Testbench）

5.2.1 Testbench 概述

在传统电路设计中，一般需要首先对电路系统进行功能划分，然后对每块电路画出真值表，用卡诺图进行逻辑简化，写出布尔表达式，画出逻辑电路图，选器件，用 PCB 板或面包板构建线路，最后进行电路测试和调试。而如果是集成电路，则需要更为复杂的工作。

今天电路规模已达到百万门的级别，使用传统方法进行电路设计调试显然已经不能满足要求。现在设计工程师一般采用构建测试平台（Testbench）的方法进行验证测试。

Testbench 主要实现两部分功能：激励产生（Stimulus Generator）和响应检测（Response Checking）。激励生成模块的主要功能是根据输入接口的信号时序，对被测设计（Design Under Test，DUT）产生信号激励，将测试信号向量输入到 DUT 中。响应检测模块根据 DUT 的输出接口的信号时序，响应 DUT 的输出请求，并检查输出结果的正确性，如图 5-2 所示。

图 5-2

一般来说，Testbench 中包括如下部分：

- 在Verilog HDL中声明模块（Module）或在VHDL中声明实体（Entity）和结构体（Architecture）。
- 声明信号。
- 实例化顶层设计。
- 激励信号描述。

Testbench 是为行为级或 RTL 级模型仿真而编写的代码，它构造了芯片工作的虚拟环境，向行为级或 RTL 级模型施加激励，并检测其输出。在构建 Testbench 的过程中，对逻辑功能的描述应以行为级描述为主，着重于接口的时序和处理的过程，而不考虑内部逻辑时序。Testbench 通常利用 VHDL 和 Verilog 语言来实现，当然也可以利用一些其他的数据文件或者是语言，例如 OpenVera、E、Systemverilog 或者 C 语言等。

编写 Testbench 是验证工程师的一个重点工作，好的 Testbench 易于阅读、升级、重用。要写就写好的 Testbench，就要对设计进行深入地理解，并满足设计规范的要求。

5.2.2　结构化的 Testbench

单纯地使用激励产生和响应检测模块构建 Testbench 的方法在实际的工程开发中是不可行的，必须在其基础上做更加优化的设计。原因有以下几点：

（1）从结构上看，激励产生和响应检测模块同处于信号级层面，这种单一的层次结构会使模块不具有复用性。激励生成模块和响应模块会随着 DUT 的功能和信号接口的不同而改变，因此即便是在同类产品的开发中，各种设计所需的验证环境也由于功能和所使用的总线协议的一些差别而无法做到项目之间的复用。

（2）这种结构的验证环境所需的开发代码量很大。因为在系统开发过程中，需要完成模块级、芯片级和系统级的测试，并且每个环节 DUT 的功能也日益复杂，所以在各个阶段需要完成的测试用例数量是相当大的。因为根据每一个测试用例都要单独开发所需模块，并单独构造仿真环境，所以开发任务相当繁重。

（3）测试模块本身需要完成的功能较多，构造起来比较复杂。激励生成模块不仅要实现同 DUT 的接口信号协议，能够驱动总线信号，还需要能够产生符合测试用例要求的信号激励数据。响应检测模块不仅要求能正常地响应 DUT 的输出请求信号，还要完成 DUT 输出结果分析、总线信号时序检测和代码覆盖率分析等工作。

所以，为了提高 Testbench 的复用性和开发效率，必须采用结构化的方法构造 Testbench，将数据流同信号流分离开。封装是使 Testbench 层次化和结构化的重要手段。C++语言采用类的形式将内部操作的细节封装起来，从而实现了类的复用。Verilog 语言中也提供了类似的数据封装方式，这就是模块（module）。借用模块定义，对 Testbench 的信号时序和模块连接两个方面进行封装，就可以实现 Testbench 的结构化。

对信号流的封装是通过总线功能模型（Bus Function Model，BFM）和嵌入式检验器（Verification Element，VE）进行的。总线功能模型对底层总线的时序和操作进行封装，嵌入式检验器对总线信号数据的抓取操作进行封装。BFM 和 VE 都向上层提供任务接口，从而使上层可以直接通过任务调用就完成相应的总线操作，而不关心实现的细节。

BFM 和 VE 的引入将测试环境分成物理层和测试层，如图 5-3 所示。

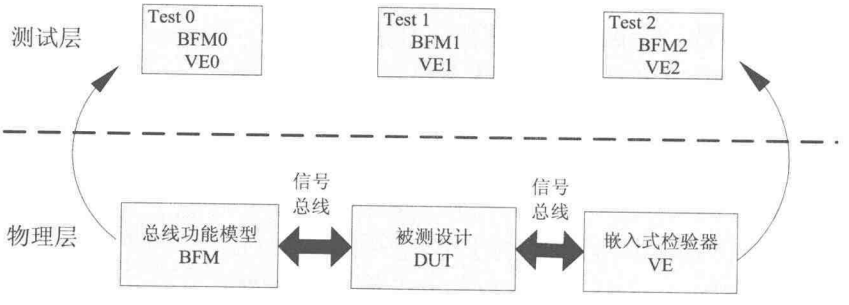


图 5-3

对模块连接的封装是通过模块 Harness 完成的。Harness 是将一个测试用例所需要的 BFM 和 DUT 都以实例化的形式相连，使在测试层看到的是物理层的一个整体。Harness 封装了所有直接对 DUT 进行操作和通信的组件，从而将信号彻底屏蔽，在测试层能够使用调度任务来控制数据流。

通过对信号和连接两方面的封装，Testbench 的结构如图 5-4 所示。

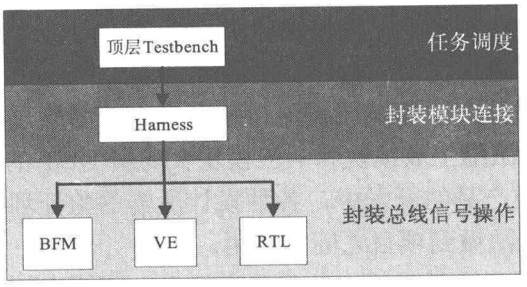


图 5-4

结构化的 Testbench 体现出了很强的复用性。在同一个项目中，BFM 和 VE 既可以在模块级的仿真环境中使用，还可以嵌入到系统级的仿真环境中使用：针对同一个模块的测试只需要使用 Harness 模块封装一次，就可以胜任针对该模块的每一个测试用例的测试。在不同项目中，由于 BFM 和 VE 只同信号接口有关而与测试功能无关，于是可以将其使用在引脚接口信号通信协议相近的项目开发中，例如一个 PCI 总线的 BFM 可以使用在任何一个应用了 PCI 总线接口的芯片的验证过程中；测试用例则可以使用在系统功能相近的项目中，例如一个可以产生网络数据包的工具包则可以使用在任何一个路由器或以太网交换芯片的开发验证过程中。良好的复用性可以通过积累而提高后续项目的验证效率。

5.3 仿真技术

5.3.1 前仿真

仿真，是对电路设计的一种间接的监测方法，用软件的方法对电路设计的逻辑行为和运行功能进行模拟测试，可以获得电路设计的更多信息，以进一步对原设计进行排错和修改。尤其对于使用硬件描述语言（HDL）设计的大型系统，进行可靠、快速、全面的仿真测试更加重要。

仿真可以分为前仿真和后仿真。图 5-5 显示了仿真过程的流程示意图。

所谓前仿真，就是将设计的文件在未经综合、布局布线之前调入 HDL 仿真软件进行仿真，检查逻辑功能是否正确，也被称为功能仿真。前仿真针对的对象是手工设计出的硬件描述语言（HDL）代码，一般为行为级描述或 RTL 级描述，一般认为 RTL 级描述是可综合的，而行为级描述的可综合性非常差，所以在电路设计中要求工程师最后交付的一般都是 RTL 级的代码，所以在针对 RTL 级的代码进行前仿真时，也有人称其为 RTL 级仿真。在前仿真中还有一个值得注意的问题是前仿真中不包含任何延时信息。

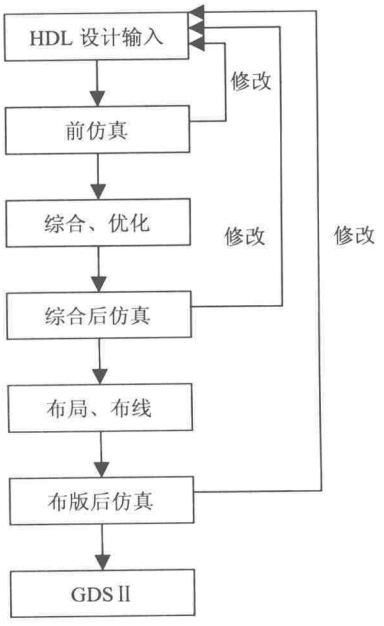


图 5-5