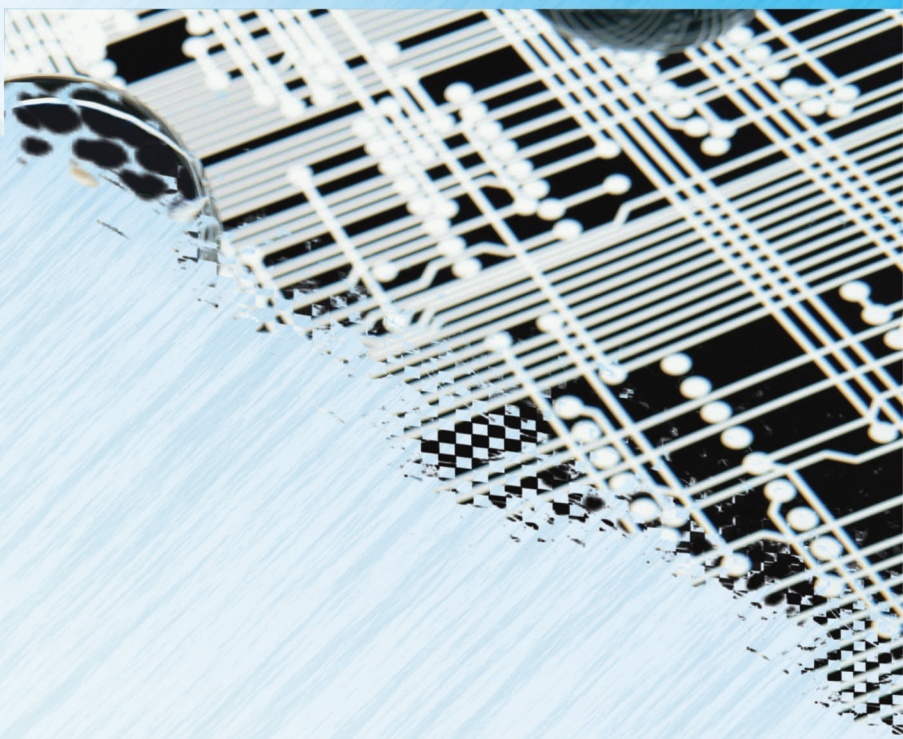
李晓维　吕 涛　李华伟　李光辉 著

数字集成电路设计验证

——量化评估、激励生成、形式化验证



科学出版社
www.sciencep.com

数字集成电路设计验证

——量化评估、激励生成、形式化验证

李晓维　吕　涛　李华伟　李光辉　著

科学出版社

北　京

内 容 简 介

本书内容涉及数字集成电路设计验证的三个主要方面:量化评估、激励生成和形式化验证。主要包括寄存器传输级(RTL)电路建模、基于可观测性的覆盖率评估方法、设计错误模型;基于故障模型的激励生成、基于 RTL 行为模型的激励生成、覆盖率驱动的激励生成;基于可满足性的等价性检验、包含黑盒电路的形式化验证,以及不可满足问题。

全书图文并茂,阐述了作者及其科研团队自主创新的研究成果和结论,对致力于数字集成电路设计验证方法研究的科研人员(尤其是在读研究生),具有较大的学术参考价值,也可用作集成电路专业的高等院校教师、研究生和高年级本科生的教学参考书。

图书在版编目(CIP)数据

数字集成电路设计验证:量化评估、激励生成、形式化验证/李晓维等著.—北京:科学出版社,2010

ISBN 978-7-03-027609-4

Ⅰ.①数… Ⅱ.①李… Ⅲ.①数字集成电路-电路设计-验证 Ⅳ.①TN431.2

中国版本图书馆 CIP 数据核字(2010)第 089305 号

责任编辑:刘宝莉/责任校对:刘小梅

责任印制:赵 博/封面设计:鑫联必升

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

源海印刷有限责任公司 印刷

科学出版社发行 各地新华书店经销

*

2010 年 5 月第 一 版 开本:B5(720×1000)

2010 年 5 月第一次印刷 印张:26 3/4

印数:1—3 000 字数:518 000

定价:58.00 元

(如有印装质量问题,我社负责调换〈科印〉)

FOREWORD

Verifying that a hardware design complies with its functional specifications continues to be a major challenge. It has been reported that up to 40% of the human resources and 70% of the computing resources in a typical high-end microprocessor design project are devoted to the verification task. While this is understandable (releasing buggy designs can be quite costly economically and otherwise), it still leaves one to wonder why such a mechanical task should consume so much of a design team's time and talent, resources that are better deployed to the more creative aspects of design. What's even more disconcerting is that the current approaches to verification (simulation, emulation, and formal methods) can hardly keep up with the exponential increase in design complexity-evident from the increasing numbers of bug-escapes through fabrication and consequently expensive re-spins for each new generation of design.

The Joint Research Laboratory for Advanced Test Technology at Institute of Computing Technology of Chinese Academy of Sciences has been very active in researching verification technologies and has produced a broad range of impactful results on coverage metrics, vector generation, equivalence checking, and model checking. While individual results were published as PhD theses as well as conference and journal papers, this book provides a self-contained, unified, and in-depth coverage of these timely topics in a coherent fashion and thus offers significant additional values beyond the collection of theses and papers. The coverage of each topic includes motivations and problem definitions, a survey of prior arts, detailed descriptions of new advances, and demonstrations of experimental results and comparisons. Therefore, this book will allow the readers to gain better understanding of both the fundamentals as well as more advanced development for these important subjects.

The total state space for complete verification is well beyond what can be practically checked. Thus some modeling and analysis techniques are necessary for estimating the degree of completeness of the verification tasks as well as for guiding the selection of additional test cases and/or design modifications for enhancing verifiability. Many design and verification teams today use code-, functional-, and/or assertion-coverage to estimate verification coverage while it is universally acknowledged that each type of models or metrics often has only limited

and very specific application and thus effectiveness of these metrics remain design- and flow-dependent. In Part I of the technical content of the book, including Chapters 2 to 5, gives a thorough treatment of this subject and reports some exciting new research results to address the limitations of existing models and metrics.

Automatic generation of high-quality vectors remain a central research focus for verification of debugging. Part II of the technical content ,including Chapters 6 to 9, describes error-model-driven and coverage-driven vector generation. Both deterministic and guided-random approaches are discussed.

The promise of formal verification has been hailed, by some, as a panacea. Alas, formal methods have failed to convince the skeptics and have somewhat fallen short of that promise. To be sure, some aspects of design verification are now routinely performed with tools that employ formal reasoning; for example, establishing the equivalence between two implementations of the same design is now regularly done using formal equivalence checking tools which have more or less replaced simulators as the preferred method. Property verification via model checking, while has not caught on to the same extent, is gaining increasingly popularity. Part III of the technical content of the book, including Chapters 10 to 14, addresses the scalability and computational efficiency of various formal methods. It starts with an introduction to formal verification approaches, followed by a thorough coverage of incremental equivalence checking, optimization techniques to the Boolean satisfiability problem, and a number of core issues for model checking.

The fast and continuing evolution of verification technology and the enormous growth in the complexity and sophistication of the coverage and verification tools has made it such that very few people can really master all fronts of this subject. New problems, new algorithms, and new methodologies and tools continue to emerge. As a result, it is becoming difficult even for experts to follow and comprehend the progress on a continuing basis. This book organizes a huge body of complex materials into a logical and easy-to-understand fashion and thus will help graduate students, engineers, and researchers to get a better grasp of recent advances and gains better insight into this fast evolving field.

Kwang-Ting (Tim) Cheng (郑光廷)
Univ. of California, Santa Barbara
December 4, 2009

前　言

实现所期望的功能特性是集成电路设计需要满足的最基本要求，功能验证是集成电路设计验证的基础技术，是集成电路设计的关键技术之一，是确保集成电路功能正确性的主要技术手段。

本书是全面论述数字集成电路设计验证方法的学术著作，汇集了自 2000 年以来中国科学院计算研究所（以下简称中科院计算所）在数字集成电路设计验证方法学研究中取得的自主创新的重要研究成果和结论。内容涉及数字集成电路设计验证的三个重要方面：量化评估、激励生成和形式化验证。

全书共 15 章，其中技术内容可分为三大部分。第一部分（第 2～5 章）量化评估，从可观测性信息和发现设计错误的能力两个角度，论述数字集成电路设计验证的量化评估方法。第二部分（第 6～9 章）激励生成，针对寄存器传输级的激励生成问题，从故障模型和覆盖率导向两个角度，论述确定性和非确定性的激励生成方法。第三部分（第 10～14 章）形式化验证，从提高处理速度的角度，论述形式化验证中的等价性检验方法和模型检验方法。

本书的主要技术内容汇集了李晓维研究员从 2001 年以来指导的博士生（李光辉、吕涛、鲁巍、邵明等）和硕士生（刘领一、赵阳、王天成等）的学位论文工作的部分成果；也包括李华伟和尹志刚的博士学位论文的部分成果。这些研究成果部分已经在本领域相关学术刊物和学术会议上发表。本书由李晓维研究员主持撰写，吕涛博士参与了第 3～5 章内容的整理；李华伟研究员参与了第 2、6～9 章内容的整理；李光辉教授参与了第 10～14 章内容的整理。清华大学计算机系边计年教授审阅了全部书稿，美国 UCSB 计算机系主任郑光廷教授撰写了序言。在此表示衷心的感谢。

本书汇集的部分科研成果是在国家重点基础研究计划（973）课题“高性能处理芯片的设计验证与测试”（2005CB321605）、国家自然科学基金重点项目“数字 VLSI电路测试技术研究”（60633060）和“从行为级到版图级的设计验证与测试生成”（90207002）等资助下完成的。研究过程中得到了中科院计算所李国杰院士、闵应骅研究员、胡伟武研究员、李忠诚研究员等领导和同事的关心和支持，得到了清华大学边计年教授、复旦大学唐璞山教授、浙江大学严晓浪教授、西安邮电大学韩俊刚教授等同行的支持和帮助，在此表示衷心的感谢。

由于作者水平和经验有限，书中难免存在不足之处，恳请读者批评指正。

目　　录

第1章 绪 论

验证是一个使用非常广泛的词汇。在电子设计自动化(electronic design automation,EDA)领域,所谓设计验证(design verification),指的是判定一个设计的实现(implementation)是否满足其规范(specification)。一个电子器件通常具有很多方面的特性,包括功能、性能、功耗等。相应的,设计验证也包括了很多方面的技术,如功能验证(functional verification)技术、静态定时分析技术、功耗分析技术等。因此,验证这个词在不同的上下文中有着不同的含义。本书着眼于功能验证技术。实现所期望的功能特性是一个电路设计需要满足的最基本要求。如果没有特别声明的话,后面提到的设计验证均指对功能的验证。

众所周知,在产品的设计流程中,设计错误(design error,通俗来讲即设计中的 bug)发现得越晚则带来的损失越大。著名的 Intel 公司的 Pentium 处理器的浮点除法错误,导致了约 4 亿美元的损失[1]。由此可见,在芯片(chip)设计流程中实施充分的有效的设计验证工作,其重要性非同寻常。

随着集成电路工艺的不断细化,设计验证在集成电路设计流程中所耗费的开销越来越大。2007 年度的《国际半导体技术发展报告》(*The International Technology Roadmap for Semiconductors*)指出,在当前的工程项目中,验证工程师的人数超过了设计工程师,对于复杂的设计更是达到了 2∶1 的比例[2]。

工业需求推动着验证技术的科学研究工作不断升温,受到越来越多的学者的关注。2007 年度的图灵奖就授予了提出模型检验技术的 Edmund M.Clarke 等人,这充分表现了科学界对于设计验证技术的关注。电路设计的规模和复杂性随着半导体制造工艺的细化呈上升趋势,如果没有重大突破的话,验证问题将是半导体工业发展的重大障碍。

1.1 设计验证简介

在常见的数字集成电路设计流程中,不同的设计环节对应着不同抽象层次的设计。图 1.1 是常见的不同抽象层次的设计及验证。依照抽象程度从高到低的顺序,分别是功能规范、算法级/微体系结构级设计、寄存器传输级(register-transfer level,RTL)设计、门级(gate level)设计和物理级设计。相应的,对某个设计的验证任务,常常被转化为该设计与相邻的具有较高抽象层次的设计之间是否相符的问题。在门级和物理级这样的较低抽象层次上,设计验证技术已经相

对成熟，尤其是形式化的等价性检验技术，已经被业界广泛采用。而在 RTL 以上的抽象层次上，设计验证技术虽然也取得了不断的发展和进步，但是尚没有完善的解决方案。这方面的技术包括基于模拟的验证技术，以及形式化验证中的模型检验和定理证明。

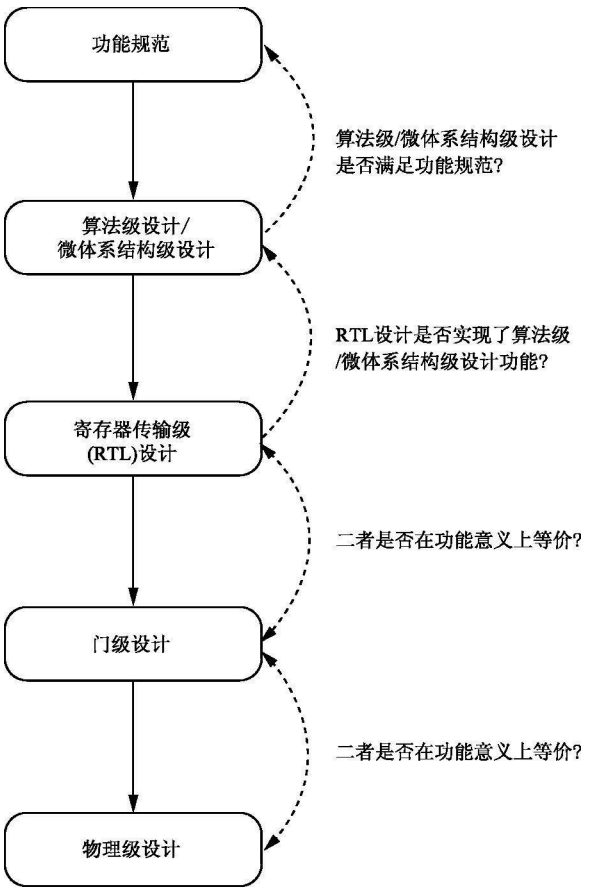


图 1.1　不同抽象层次的设计及验证

模拟是业界常用的，也是最主要的验证技术。开发模拟验证环境所用的语言经历了几次更新换代，从直接采用硬件描述语言(hardware description language，HDL)，到采用面向对象的语言(如 C＋＋)，再到专门的验证语言(如 SystemVerilog 语言和 e 语言)。专门的验证语言使得验证工程师可以方便地开发出功能强大的验证环境。模拟验证中的典型验证环境如图 1.2 中虚线内所示。图 1.2 中，矩形结点表示验证环境的组成模块，虚线外部的结点表示与模拟验证过程有关的文档，连线表示信息或数据的流向关系。

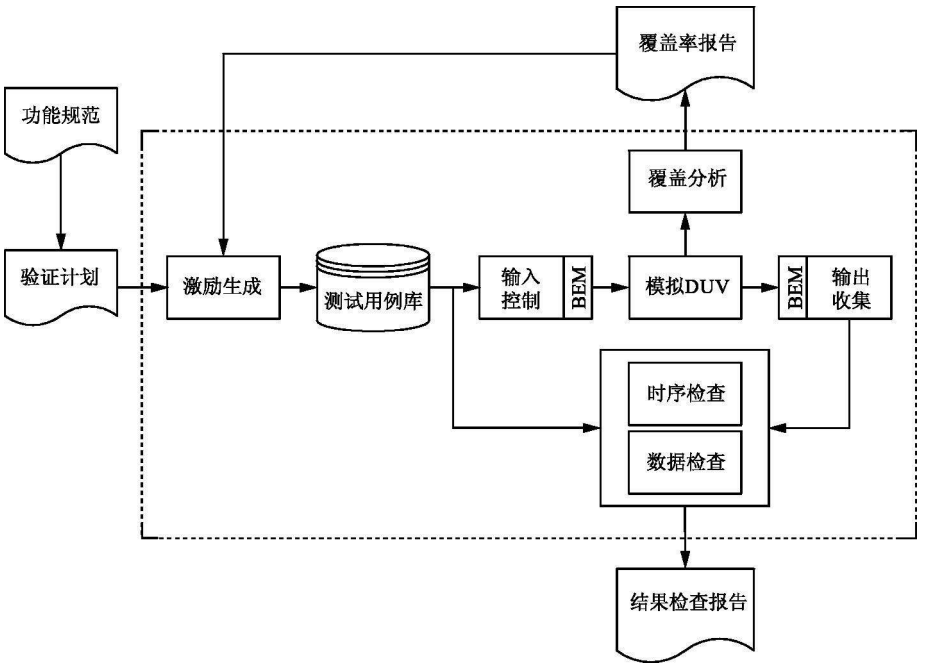


图 1.2 典型的模拟验证环境

在模拟验证过程中,验证工程师首先需要理解功能规范并制定出相应的验证计划,然后依照验证计划开发测试用例(test case)。除了定制一些测试用例之外,经常采用的技术还有基于约束的随机激励生成(constraint-based random vector generation)。为了提高验证环境的开发效率和验证环境中模块的可重用性,验证语言通常采用类似面向对象语言的方式来描述测试用例。因此在向被验证设计(design under verification,DUV),尤其是 RTL 或门级的 DUV 施加测试用例时,需要将较抽象的测试用例映射成二进制向量①(vector),必要时还需要通过总线功能模型(bus functional model,BFM)[3]按照 DUV 所需要的时序关系来施加二进制向量,这就是图 1.2 中输入控制部分需要实现的功能。输出收集部分的功能则是输入控制部分功能的逆过程。对于模拟过程中收集到的数据,一方面需要进行结果检查,包括时序检查和数据检查,形成结果检查报告以反映模拟验证过程中是否发现了设计错误;另一方面需要进行覆盖率分析,形成覆盖率报告以评估模拟验证的充分程度。对于设计验证环境中各部分的详细介绍,可见文献[4]。

模拟验证技术需要构造各种测试用例,对于大型复杂的设计,模拟会消耗大量的时间。然而,由于市场的压力,又必须尽可能地缩短设计周期来加快产品面市。

① 在后面的章节中,当不强调功能场景时,通常不讲"测试用例",而直接讲"向量"、"输入向量"或者"激励"。

面市时间(time to market)和设计复杂度这两个方面的因素导致了模拟所能覆盖的功能越来越不完全。为此,人们转而求助于其他的验证技术来补充。形式化验证(formal verification)是指利用数学方法来严格地证明一个电路或系统(实现)满足给定的需求(即规范)[5,6]。它的好处是隐式地穷举了被验证电路的输入变量空间,即它可以达到100%的功能覆盖率(functional coverage),而且不需要产生激励。如果一个电路被证明是错误的,那么形式化验证工具将会给出一个反例,即提供一个输入向量,证明这个电路与其规范关于这个激励的响应不同。模拟验证方法能够发现设计错误,但难以证明设计没有错误,然而形式化验证却可以证明设计到底有还是没有错误。从这种意义上说,形式化验证是比模拟更严格、更完全的方法[5~7]。

形式化验证方法大体上可分为三类:等价性检验(equivalence checking)、模型检验(model checking)和定理证明(theorem proving)。等价性检验是指证明两个设计模型具有相同的功能,有时也称为逻辑验证或布尔比较(Boolean comparison)[8]。这是目前在实际的芯片设计中应用最为广泛的形式化验证技术,如Synopsys公司的Formality、Mentor Graphics公司的FormalProTM、Cadence公司的Affirma等EDA工具都能进行等价性检验,并且能处理较大规模的设计[6,9~11]。等价性检验的一个典型应用就是证明经过综合(synthesis)得到的门级网表(netlist)与综合之前的寄存器传输级模型是否功能等价。模型检验则是证明一个设计是否满足某种属性,例如,证明一个实现通信协议的设计没有死锁现象发生。目前模型检验方法受到越来越多的关注,并已被多个厂家集成到EDA工具中,如Cadence公司的Formal Checker。定理证明是指将要证明的系统性质表示成基于某种逻辑的形式化系统的公式,以该形式化系统的公理和推理规则为基础,推导出表达被证明性质的公式,从而证明设计的系统具有该性质[6]。定理证明要求用户深刻理解基本逻辑与形式证明,这种方法的代价较高,因此在学术领域之外的实际应用还很少见。

1.2 设计验证中的关键问题

1.2.1 量化评估

验证难题一直没有突破性的解决,设计规模的飞速增长是一个主要原因。功能验证技术主要有模拟验证和形式化验证两个大类。形式化验证技术[6]在某种意义上可以证明功能的正确性,但是其处理规模难以满足需要一直是一个萦绕不散的问题;模拟方法由于具有良好的可扩展性,可以很容易地应用于大规模设计,所以一直以来在业界都有广泛应用。但是模拟验证不知道验证到什么时候就可以结

束了,因此,如何量化评估模拟验证的质量,是模拟验证所关注的重要问题之一。

在设计验证的量化评估方面,已有技术主要关注覆盖率评估(coverage evaluation)研究。覆盖率主要有两个方面的作用:一是作为量化功能验证完全性的启发式尺度;二是用来发现设计中那些模拟不充分的部分,以指导产生后续的输入向量[12]。

业界常用的一些覆盖率评估方法,如代码覆盖率(code coverage)、条件覆盖率(condition coverage)、翻转覆盖率(toggle coverage)等,主要关注语句或电路结构的可控制性(controllability)。例如,语句是否被执行、电路中的寄存器(register)是否被翻转等。然而,一个好的覆盖率评估方法应该能够从可控制性和可观测性两个方面对模拟进行评估。随着设计规模的持续增长,验证工程师不可能对所有的内部信号实施结果检查——穷尽观察所有内部信号的波形显然是不现实的,如果采用模拟值与期望值之间的数据自动比较,由于实现细节上的不同,通常也很难为设计中的所有内部信号在参考模型中找到对应的信号以产生期望值,因此验证工程师通常选择输出信号和部分重要的内部信号作为观测信号,来检查功能的正确性。由此可见,仅仅考虑代码或电路结构是否被执行,而无视其执行的效果可能并没有传播到观测信号,这样的覆盖率评估方法往往存在覆盖率虚高问题(occurred but not verified problem),导致过于乐观的量化评估数据。对大规模设计而言,如何在设计验证的覆盖率评估方法中融合可观测性信息是一个必须解决的问题。

发现设计错误是模拟验证的主旨之一,因此研究模拟验证的量化评估方法除了需要考虑可观测性之外,还需要将设计错误的效果考虑在内。已有的模拟验证覆盖率评估方法,包括对功能覆盖率、代码覆盖率、有限状态机(finite state machine,FSM)覆盖率等的评估方法中,大多不考虑这方面的问题,因此对验证技术的指导意义具有局限性。例如,对于 FSM 的状态覆盖率而言,当状态覆盖率达到100%时,意味着每个状态都执行过,但是 FSM 的每个状态通常控制着一部分逻辑,而运算逻辑也会屏蔽一部分设计错误,因此一个状态被执行并不等同于其所控制的逻辑被充分验证了。如何从发现设计错误的角度来提出设计验证的量化评估方法,是一个很有意义的、需要解决的问题。

综合上述,可观测性信息和发现设计错误的能力,是设计验证量化评估方法需要考虑的因素。从这两个角度出发,如果能够提出高质量的量化评估方法,将对设计验证产生良好的促进作用。

1.2.2 激励生成

激励生成(vector generation),又称测试生成(test generation,TG),为待验证的设计产生输入向量以检测可能的设计错误,是模拟验证的基础技术之一。

模拟验证是指用对设计进行模拟的方式来发现其中的错误,其关键在于怎样得到所需要的输入向量,以及怎样知道输入向量对应的正确的结果。如图 1.3 所示,这是对模拟验证过程的简单概括。其中,把 DUV 对应的正确电路称为参考模型(reference model),对二者同时施加相同的输入向量,如果二者结果不同,则会报错,表示发现被验证设计中有错。

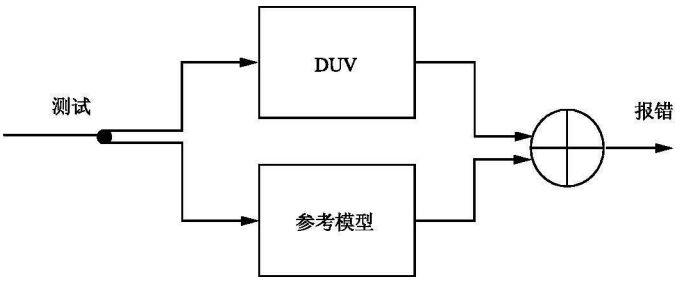


图 1.3 模拟验证的简单模型

这里,参考模型的用意在于得出正确的结果,它可以是电路形式化可模拟的规范,也可以是另一种不同形式的设计,甚至可以是一个虚拟的模型,只简单给出期望的正确结果。当然,实际中所用到的“参考模型”本身并不一定完全正确,但这样的模拟验证如果报错,说明两者肯定有一个存在错误,这也是有意义的。如果不报错,说明 DUV 通过了当前的测试,但不能说明它是正确的。

显然,测试生成对于检验设计是否存在错误非常重要。测试生成一般以参考模型或参考模型的变异体(故障电路)为对象。自动测试生成方法从不同的方面来进行分类。从其针对的电路对象的不同可以分为“组合电路测试生成”和“时序电路测试生成”等。从其采用的故障模型的不同又可以分为“固定型故障测试生成”、“开路故障测试生成”、“跳变故障测试生成”等。从故障的数目不同可以分为“单故障测试生成”和“多故障测试生成”等。从方法特征的不同可以分为“代数法测试生成”和“基于结构测试生成”等。另外,算法还有“完全性”和“非完全”之分、“随机”和“确定性”之分等。对于某个具体测试生成算法而言,它当然有各个不同方面的特征。

由于设计验证采用的参考模型往往是高层(寄存器传输级及以上)抽象模型,模拟验证的测试生成属于时序电路测试生成。与组合电路测试生成算法相比,因为其庞大的搜索空间,时序电路测试生成算法远没有那么高效实用。时序电路测试生成方法一般可分成时间帧扩展方法和基于模拟的方法。时间帧扩展的方法把时序电路“展开”成一个大的组合电路,需要面对庞大的搜索空间,导致效率很低,而且效果也不一定很好。基于模拟的方法首先需要的是一个好的逻辑/故障模拟器,它避免了庞大的搜索空间,在某种程度上是有效的。不过,这类方法总是存在

故障覆盖率不高的问题,特别是那些用简单的方式从随机向量中选优的方法。为此,遗传算法往往被用于这类方法的随机向量的改进与选择。

1.2.3 形式化验证

目前,等价性检验是在集成电路设计流程应用最广泛的形式验证技术之一。然而,无论是从理论还是应用领域来看,等价性检验都还有一定的局限性。例如,关于大型时序电路的验证,通常是通过构造存储元素(如寄存器)的映射,将时序电路转化为相应的组合模块,然后依次验证各组合模块的等价性,如果各组合模块是等价的,那么这两个时序电路是功能等价的。但是,当找不到这样一种映射关系时,可能不得不使用传统基于有限状态机遍历的方法来验证,这时常常因为电路状态爆炸而失败。关于组合电路的验证,目前学术界关心的是如何构造更强健的核心算法引擎,提高算法的效率和处理能力[13~16]。我们在使用商业等价性检验工具验证芯片设计的实际中,遇到的主要问题是存储元素映射时间过长,再就是处理能力不够强[11,17]。

在集成电路的设计流程中,一旦验证工具发现设计规范与实现不一致,那么设计者或验证人员都面临一个非常棘手的任务——找到实现中的错误或者尽可能压缩可疑的错误区域,以方便随后的错误定位与纠正过程,这就是所谓的设计错误诊断问题。错误诊断方法主要有两类,即基于模拟的方法和符号方法。前者准确性较差,但能应用到较大的电路;后者通常以二叉判决图(binary decision diagram,BDD)为主要数据结构,准确性较高,但由于 BDD 本身的缺陷,使得这类方法的处理能力有限[6,8]。

因此,构造强健的等价性检验核心引擎以提高算法的效率和处理能力,以及提高错误诊断算法的准确性成为解决上述相关问题的关键。

近年来,在可满足性(satisfiability,SAT)求解方面取得了一些重要进展。这使可满足性成为 EDA 相关领域的一种新方法,产生了很多富有研究价值的课题。例如,利用字位混合求解的 RTL 设计验证问题;利用 SAT 求解器的测试生成以及诊断问题等。

利用 SAT 问题进行验证的方法,一般是将电路设计和待验证的属性转化为一组布尔约束,通过求解这组布尔约束,获得关于设计正确性的信息。在等价性检验方面,SAT 问题的应用已经很广泛。而在模型检验方面,基于 SAT 问题已经产生了很多新的分支,其中具有代表性的包括基于 SAT 的无界模型检验、有界模型检验等。针对电路问题,通用的 SAT 算法有很多可以改进的地方。电路 SAT 问题有其独特的属性和结构,利用这一点可以提高求解的效率。

1.3 章节组织结构

1.2 节介绍了数字集成电路设计验证中的关键问题。相应的,本书主体内容将围绕这些关键问题而展开。本书技术内容组织如下。

第一部分(第 2～5 章)介绍量化评估方法。随着设计复杂性的持续增长,模拟验证的不完备性日益突出,这使得提出更准确、更有意义的量化评估方法成为一个亟待解决的问题。常用的量化评估方法侧重于分析设计中某些内部结构被执行的情况,这导致了覆盖率数据的虚高。针对这一问题,本书从可观测性和发现设计错误的能力两个方面入手,研究设计验证的量化评估方法。

第 2 章介绍了硬件描述语言和寄存器传输级描述的相关知识。对硬件描述语言的产生、发展及特点作简要介绍,主要探讨其中寄存器传输级行为描述的特点,并提出针对寄存器传输级行为描述的有效电路模型和数据结构,为后面的章节研究覆盖率评估和激励生成方法奠定了基础。

第 3 章介绍了基于可观测性的覆盖率评估方法。所讨论的内容包括可观测性的 DFUDO 模型、基于 DFUDO 模型的语句覆盖率评估方法和分支覆盖率评估方法,以及可观测性的 COC 模型、基于 COC 模型的语句覆盖率评估方法。同时,我们通过实验数据说明了各个覆盖率评估方法的意义和评估能力,并对所提出的几种覆盖率评估方法进行分析和比较。

第 4 章的内容与设计错误模型相关。首先,简要介绍在两个实际芯片设计项目中的功能验证工作;然后,在对验证过程中所发现的设计错误进行分析的基础上,提出了缺项错误模型及其测试方法,并针对实际芯片设计进行了实验和数据分析,以说明缺项错误模型的效果;最后,介绍了设计错误的自动注入系统,包括软件的变异测试系统 Mothra 与其在硬件验证中的扩展,以及一个我们自己设计的错误注入系统 ErrorInjector。

第 5 章给出了 RTL 设计中字操作的错误屏蔽概率计算方法,并详细讨论了典型的字操作的错误屏蔽概率,包括算术运算、关系运算、移位运算、逻辑运算,等等。接着,介绍了 HDL 设计中信号的一种静态可观测性量化评估方法。利用该方法所得到的量化值揭示了信号上的错误被观测的难度,这对于设计验证技术具有指导意义。为了说明其指导意义,还提出了一种内部观测信号选择算法,并进行了相关实验和数据分析。

第二部分(第 6～9 章)介绍激励生成方法。传统的激励生成方法是由验证者定义各种功能测试用例,但是这种方法所产生的激励只能覆盖到事先设计的测试情形,因此验证的效果完全依赖于验证人员的经验。为了应对设计规模的增长,开发自动化的激励生成技术是大势所趋。本书针对寄存器传输级描述,重点探讨了

基于故障模型的测试生成方法和覆盖率驱动的非确定性激励生成方法。

第6章介绍了寄存器传输级的激励自动生成方法的研究现状,主要包括:将遗传算法用于激励生成的非确定性方法,以及基于故障模型/错误模型的确定性激励生成方法。

第7章研究了基于故障模型的寄存器传输级激励生成方法。设计了行为倾向驱动引擎用于展现电路的行为,建立了一种新的故障模型——传输故障模型,基于行为倾向驱动引擎构造了传输故障的无回溯测试生成算法,实现了 X-Pulling 和 TiDE 两个测试生成系统,并进行了相关实验和数据分析。

第8章研究了从寄存器传输级行为描述出发,如何生成测试序列来检测有限状态机的状态转移关系。提出了有限状态机的行为阶段聚类描述,该描述可通过从寄存器传输级行为描述抽取状态信息得到。建立了一种新的行为级故障模型——行为阶段转换故障模型,进一步提出了一种基于聚类的测试生成算法。实现了基于行为阶段聚类的寄存器传输级自动测试生成系统 ATCLUB,并进行了相关实验和数据分析。

第9章研究了以有效的覆盖率为导向的非确定性的激励生成方法。介绍了两种方法:第一种,采用设计验证常用的覆盖率评估准则,附加第7章提出的传输故障模型的覆盖准则,设计了混合遗传算法来进行激励生成;第二种,针对第3章所提出的可观测性覆盖准则 COC 模型,使用第7章提出的请求-响应策略进行无回溯的激励生成。对这两种方法分别进行了相关实验和数据分析。

第三部分(第10~14章)介绍形式化验证方法。形式化验证方法使用严格的数学推理来证明设计满足规范的部分或全部属性,引起了学术界和产业界的广泛关注。然而,形式化方法仍然受限于处理规模和运行时间,难以广泛应用于实际设计。针对这一问题,本书的工作重点探讨了增量等价性检验算法、包含黑盒的电路的设计验证方法、极小布尔不可满足子式的提取算法,以及针对电路设计的模型检验方法等。

第10章介绍基于电路的布尔推理常用方法。简介布尔函数的基本运算与数字硬件行为的建模方法。介绍布尔函数的规范表示方法——有序二叉判决图的概念、运算、变量排序方法,并讨论了 BDD 引擎在形式化验证中的应用。介绍布尔可满足性问题及其判定方法,并讨论了基于 SAT 的测试产生方法、等价性检验方法和有界模型检验方法。最后讨论静态逻辑蕴涵的算法与应用。第10章的内容为后面的章节研究形式化验证方法奠定了基础。

第11章介绍了一种基于增量可满足性的组合等价性检验方法,该方法通过有选择地去掉一些候选等价结点,通过置换等价结点来简化 miter 电路,并通过合取范式(conjunctive normal form,CNF)子句的分组策略来实现增量的可满足性算法,从而提高整个验证算法的性能。实验结果表明,对于相同的电路,提出的方法

较以往算法得到的候选结点数少了 1 个数量级，而且速度要快 1 个数量级，因此更强健、有效。

第 12 章提出了一种结合逻辑模拟和布尔可满足性的含黑盒设计验证方法，将模拟方法与形式化的布尔比较相结合，使用并行逻辑模拟来检测黑盒外部可能的设计错误，通过基于布尔可满足性的布尔比较增强模拟算法。与基于二叉判决图的方法相比，该方法具有更强的处理能力，有效降低了算法的空间复杂度。该方法已成功地应用于提高设计错误诊断的效率。通过 ISCAS'85 电路的实验研究表明，在具有与同类方法相当的错误检测能力的情况下，该方法对于各实验电路平均要快 2 个数量级。

第 13 章研究了极小不可满足子式的计算问题。这类算法分为近似算法和精确算法。近似算法包括适应搜索算法、基于蕴涵图、冲突依赖图以及 AMUSE 的方法等。精确算法包括采用线性规划以及遍历子句的方法。本章利用模拟的方法，考察了基于蕴涵图的极小不可满足子式的提取算法误差与 3-SAT 密度关系。即随着密度的增加，基于蕴涵图的算法误差单调减少。本章还就遍历子句的精确算法提出对部分变量进行预先赋值的优化方案。

第 14 章简介了模型检验技术，并研究了模型检验中的若干核心问题，包括：针对符号模型检验中如何构建紧致且高效的转移关系问题，提出了对分割的转移关系进行重新分组的有效方法；针对无界模型检验的效率问题，提出一种结合测试自动生成和 SAT 的无界模型检验的前像计算方法；针对以电路为对象的模型检验问题，提出定值子句的概念，揭示了 CNF 与电路结构的关系，并在此基础上实现了一个统一基于 CNF 数据结构的无界模型检验框架，在验证层次、问题规模、效率方面有一定的优势。

最后，第 15 章对本书内容进行总结，并依据所介绍的各项工作对未来的研究做出展望。

参 考 文 献

[1] Sharagpani H P, Barton M L. Statistical analysis of floating point flaw in the Pentium processor. Intel Corporation, 1994.

[2] ITRS2007. http://public.itrs.net [2007-12-01].

[3] Bergeron J. Writing Testbenches Using SystemVerilog. New York: Springer, 2006.

[4] Palnitkar S. Design Verification with e. Upper Saddle River: Prentice Hall PTR, 2003.

[5] Kern C, Greenstreet M. Formal verification in hardware design: a survey. ACM Transactions on Design Automation of Electronic Systems, 1999, 4(8): 123－193.

[6] 韩俊刚，杜慧敏. 数字硬件的形式化验证. 北京大学出版社，2001.

[7] Jones R B, O'Leary J W, et al. Practical formal verification in microprocessor design. IEEE Design & Test of Computers, 2001, 18 (4): 16－25.

[8] Huang S Y, Cheng K T. Formal Equivalence Checking and Design Debugging. London: Kluwer Academ-

ic Publisher, 1998.

[9] Synopsys Inc. Formality User Guide V2001.08. Mountain View: Synopsys Inc., 2001.

[10] Cadence Design Systems Inc. Cadence Equivalence Checker User Guide Version 2.4, San Joes: Cadence Design Systems Inc., 2001.

[11] Hughes R B. Formal verification of equivalence in DSM designs using FormalProTM. http://www.mentor.com/formalpro [2005-12-01].

[12] Tasiran S, Keutzer K. Coverage metrics for functional validation of hardware designs. IEEE Design & Test of Computers, 2001, 18(4): 36－45.

[13] Andrade F V, Silva L M, Fernandes A O. Improving SAT-based combinational equivalence checking through circuit preprocessing // Proceedings of IEEE International Conference on Computer Design, Lake Tahoe, CA, USA, 2008: 40－45.

[14] Disch S, Schollm C. Combinational equivalence checking using incremental SAT solving, output ordering, and resets // Proceedings of Asia and South Pacific Design Automation Conference, Yokohama, Japan, 2007: 938－943.

[15] Mishchenko A, Chatterjee S, Brayton R, et al. Improvements to combinational equivalence checking // Proceeding of IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, USA, 2006: 836－843.

[16] Chatterjee S, Mishchenko A, Brayton R, et al. On resolution proofs for combinational equivalence // Proceedings of the 44th IEEE/ACM Design Automation Conference, San Diego, CA, USA, 2007: 600－605.

[17] Anastasakis D, Damiano R, Ma T, et al. A practical and efficient method for compare-point matching // Proceedings of IEEE/ACM 39th Design Automation Conference, New Orleans, USA, 2002: 305－310.

第 2 章　寄存器传输级行为描述抽象方法

硬件描述语言[1,2]是目前描述集成电路设计的主要形式，其诞生是集成电路设计的需要，并极大地促进了集成电路的发展。RTL（寄存器传输级）描述作为行为综合的输出描述、逻辑综合的输入描述，在设计的抽象层次中有着重要的地位。RTL 描述与算法级描述相比，更接近电路设计的结构；与逻辑级描述相比，问题规模得到简化；而 RTL 描述与逻辑级描述的等价性验证比较成熟；因此，RTL 描述是设计验证的重要对象，本书中介绍的大部分验证方法都是在 RTL 描述上展开的。

本章在对硬件描述语言的产生、发展及特点作简要介绍的基础上，主要探讨其中 RTL 行为描述（behavioral description）的特点，并提出针对 RTL 行为描述的有效电路模型和数据结构[3,4]。本章阐述的 RTL 电路抽象方法[4]为后面的章节研究覆盖率评估和激励生成方法奠定了基础。

2.1　硬件描述语言概述

2.1.1　硬件描述语言的产生与发展

技术的发展使集成电路的设计规模日益增大，复杂程度日益提高。早期的集成电路设计方法，如逻辑图和布尔方程，特别是自底而上的设计方式，已满足不了发展的需要。客观上要求用一种自顶向下的方式来设计日益庞大的电路。这种方式不仅要能提供电路不同抽象层次简练而精确的描述，而且要能很方便地进行修改和维护。更重要的是，其描述的电路要能方便地进行模拟和验证，以及在不同抽象级别之间相互转化。因此，硬件描述语言应运而生。

硬件描述语言，顾名思义，就是用来描述硬件的高级语言。它最初由 Iverson 于 1962 年提出，旨在用简单灵活的方法来描述电路的结构、行为或特征。此后，各种 HDL（硬件描述语言）如雨后春笋般蓬勃发展起来，其中很多是以当时成熟的软件程序设计语言（如 C 和 PASCAL）为基础的。不过其中绝大多数是专有的产品，如 Silvar-lisco 公司的 HHDL（Hierarchical HDL）、Zycad 公司的 ISP、Gateway Design Automation 公司的 Verilog，以及 Mentor Graphics 公司的 BLM。甚至有些大型计算机制造商也开发自己公司内部使用的 HDL。例如，TIHDL 就是 TI（得克萨斯仪器）公司开发的在其公司内部或客户中使用的 HDL。另外，高等院校

和科研单位也开发了上百种 HDL[1,2]，如 AHPL、MIMOLA 以及 SCHOLAR。

一个 HDL 的发展除了其技术上的优势以外，更离不开同行，特别是标准化组织的支持与推广。目前，工业界和学术界用得最广的是 VHDL 和 Verilog HDL。VHDL 语言是美国国防部为了解决因语言过多而造成的信息交换和设计维护困难，而为他们的超高速集成电路计划（Very High Speed Integrated Circuit，VHSIC）提出的硬件描述语言（VHSIC Hardware Description Language，VHDL）。它主要由 TI 公司、IBM 公司和 Intermetrics 公司设计完成，并于 1987 年 12 月被 IEEE 接受为标准 IEEE Std. 1076[5]。此后又经过多次修改，先后推出 1993、2000、2002、2008 等不同版本[6]。

Verilog HDL 语言最初是由 Gateway Design Automation 公司为其模拟器产品开发的硬件建模语言，那时只是一种专用语言。由于在模拟仿真器中广泛使用，它作为一种便于使用且实用的语言被众多设计者所接受，并于 1990 年被推向公众领域而成为事实的工业标准。在 OVI（Open Verilog International）的努力推广下，Verilog HDL 于 1995 年被 IEEE 接受为国际标准 IEEE Std. 1364[7]。

现在流行的 EDA（电子设计自动化）工具一般都支持 VHDL 和 Verilog。例如，Synopsys、Cadence、Mentor Graphics 等著名的 EDA 厂商提供的工具都有对这两种 HDL 的支持。很多关于 HDL 的文献也主要针对 VHDL 和 Verilog。下面将以这两种语言为基础，讨论一下 HDL 的基本特征。

2.1.2　硬件描述语言的描述特点

硬件描述语言是一种高级语言，也有其复杂的语法和丰富的语义，以及类似一般高级语言的编程特点。不过，它与一般的高级语言（如 C++）有很大的不同，其不同主要在于，硬件描述语言所要描述的是硬件的结构、行为或特性，而不是针对某种处理器，编写能够在其上面执行的程序。因此，硬件描述语言除了有描述电路行为的顺序语句外，还有描述电路并行特征的并行语句，而且它还可以对电路从不同的描述层次，用不同的描述风格来进行抽象。

1.电路的抽象层次与描述风格

用 HDL 对电路系统的描述可以从不同的抽象层次，用不同的建模风格来进行。如表 2.1 所示，这些抽象层次有系统级（architectural or system level）、算法级（algorithmic or sub-system level）、RTL（寄存器传输级）、逻辑级（logic or gate level）和电路级（circuit or switch level）。对应于每一个抽象层次，可以用行为的（behavioral）、功能的（functional）或结构的（structural）风格来进行建模，形成对应级别电路的行为模型或结构模型。

表 2.1 HDL 对电路系统的描述层次与风格

抽象层次	功能或行为建模	结构建模
系统级	系统的功能、性能描述	子系统如 CPU、内存、驱动器、总线等之间的连接方式
算法级	子系统的 I/O 应答算法	功能模块的数据结构和相互操作
RTL	功能模块的并行操作、寄存器传输、状态转移等	控制器、运算器、多路转换器、寄存器等宏单元之间的操作
逻辑级	真值表、状态表、布尔函数、二叉判决图等	门、触发器、锁存器等元件之间的连接
电路级	微分方程	晶体管、电阻、电容等之间的连接

下面分别讨论电路的这些抽象层次。

(1) 系统级:它描述的是整个系统(system,不一定只有一个芯片)或一个系统芯片(system-on-a-chip,SoC)的系统结构(architecture)。可以用行为的方式描述其功能或性能,如指令系统、存储体系等;也可以用结构的方式描述其主要子系统的连接和关系,如 CPU、内存、总线的连接等。系统级的描述可以作为系统的规范。

(2) 算法级:它描述的是一个子系统(sub-system)或一个芯片的算法或结构。例如,用行为方式描述其 I/O 应答算法,或者用结构的方式描述其功能模块的数据结构和相互操作。

(3) RTL:它描述的是一个功能模块(module)的时序和行为。其行为的描述主要采用状态机和寄存器间的数据传输等方式。结构化方法主要描述其控制器和由运算器、寄存器、多路器等宏单元(macro-cell)组成的数据通路,以及其之间的连接。

(4) 逻辑级:它用来描述一个模块或宏单元的输入输出关系。其行为的方式主要有真值表、状态表、布尔表达式等。其结构的方式主要采用基本元件如逻辑门、触发器等逻辑器件来搭建,因此逻辑级也称为门级。

(5) 电路级:它用来表述电路的电学特型,一般针对基本单元来进行。它可以采用行为的微分方程来表征,也可以用结构的晶体管、电容、电阻等电路元件来建模。因此,数字电路的电路级也称为晶体管级(transistor level)。

上述这些级别是按照从高到低的顺序进行排列的。它们之间并没有绝对的界限,不过这种层次的划分实际上反映了一种自顶向下(top-down)的设计思路,如图 2.1 所示。其中的版图级(layout level)是电路更低一级的抽象级别,它主要考虑物理制造上具体元器件的布局(placement)及其相互之间的布线(routing)连接。它一般不方便用 HDL 来描述,而采用目前流行的版图描述形式 GDSII 来表示。

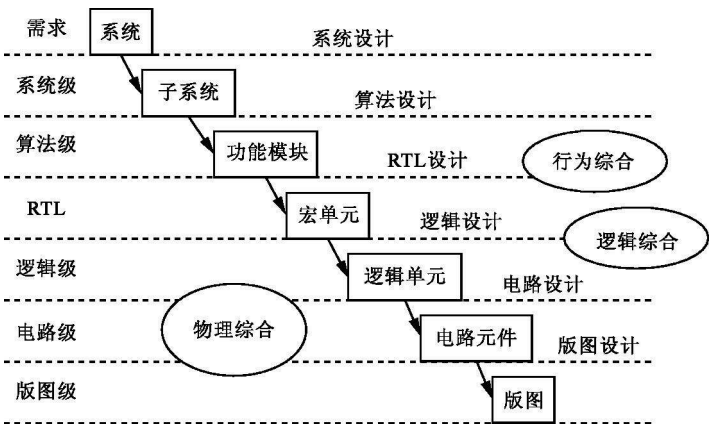


图 2.1　电路抽象级别与自顶向下设计思路

从系统级到版图级，要经过系统设计、算法设计、RTL 设计、逻辑设计、电路设计和版图设计。每一次设计，完成从高一级别到低一级别的转换，得到低一级别的描述。这种转换可以用自动化的方式来完成，称为综合(synthesis)。例如，把算法级的设计转化为 RTL 的设计称为行为综合(behavioral synthesis)。目前行为综合技术还不够完善，相应工具的能力非常有限。把 RTL 的描述转化为逻辑级的描述称为逻辑综合(logic synthesis)，目前有相当成熟的工具可以利用。经过逻辑综合得到的电路描述称为电路的网表。把网表转化为对应的版图级描述称为物理综合(physical synthesis)，这与电路的生产工艺有很大关系，一般需要有强大的计算机辅助设计(computer aided design，CAD)工具来支持。目前对某些成熟的工艺而言，其相应 CAD 工具的发展也相当成熟。

通过这种自顶向下的方式对电路进行设计，得到层次化描述(hierarchical description)。其描述的层次是通过结构建模来实现的，即这一层的结构以下一层的描述为元件(element)，通过实例化(instantiate)来实现。而下一层的描述既可以是结构化的，又可以是行为化的。因此，从这个意义上讲，行为建模和结构建模并非对立的两个方面，而是设计的两个阶段。行为建模只关心其功能而不顾其具体实现；而结构建模限定了其实现的方式和结构，且同样完成相应的功能。实际上，对一个子系统或功能模块的设计往往是先给定其功能模型，当确认其功能满足需要以后，再设计为对应的结构模型，以实现相应的功能。因此，在设计过程中的某一时刻，往往是多种描述级别和不同描述风格并存。这些不同的设计形式可以在统一的 HDL 环境甚至仿真环境下方便地进行功能模拟或仿真。

2.*并行语句与顺序语句*

与一般的程序设计高级语言不同，HDL 是用来描述硬件的行为或结构的。硬

件最基本的特征是其各部分之间是并行的,因此硬件的行为是并行的。所以,HDL 必须能够描述这种特性,这也是其区别于一般高级语言的特征。HDL 采用并行语句(concurrent statement)来表征电路的这种并行特征。并行语句能很好地体现电路的结构,却不能方便地描述电路的行为。因此,与一般高级语言一样,HDL 也大量采用顺序语句(或过程性语句,procedural statement)。利用顺序语句可以很方便地用程序设计的形式来描述电路的行为。所以,硬件描述语言有并行和顺序两种语句。

图 2.2 表示了 HDL 描述与其并行语句和顺序语句之间的关系。描述由一系列的并行语句组成,主要包括并行赋值语句、进程语句(VHDL 中的 process 语句或 Verilog 中的 always 语句)、例化语句和生成语句。它们对应了描述的"结构"。而其中的进程语句则是由一系列的顺序语句组成,主要包括赋值语句、条件分支语句和循环语句。它们反映了进程的"行为"。

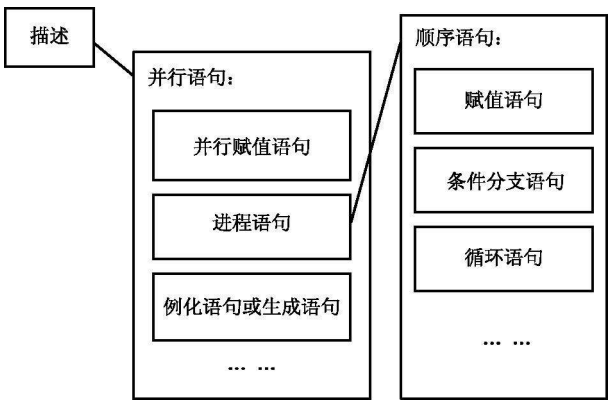


图 2.2 描述的并行语句与顺序语句及其关系

考虑到并行赋值语句相当于一个简单的进程,而例化语句和生成语句只不过体现了描述的层次,可以认为进程是最基本的并行语句。进程的行为是通过其内部的顺序语句的串行执行来体现的。因为每个进程之间是并行的,所以若要模拟整个电路的行为,则必须要求所有进程的内部语句"同时"开始串行执行。在实际的模拟中,这可以通过为每个模拟时间(simulation time)点设置多个模拟周期(simulation cycle)来实现。

如图 2.3 所示,对应一个模拟时间点可以有多个模拟周期(注意,有的资料把一个模拟时间点的模拟称为一个模拟周期,本书的处理不同),而一个模拟周期代表所有被触发进程的一次执行。在一个模拟周期内,各个进程可以按照任意的顺序执行。每个进程的执行是通过串行执行其内部的语句来实现的,其控制与一般高级语言的执行方式类似,如也有条件分支或循环,但其中最基本的操作就是计算