

第一章 绪论

1.1 课题研究背景及意义

集成电路的出现已近半个世纪，它一直遵循着摩尔定律不断向前发展，约每隔 18 个月其规模和性能将提高一倍。如今无论是在消费市场还是在商业应用领域，日益需求体积更小、成本更低、功耗更低、功能更强的电子产品，专用集成电路在一定程度上已经不再满足用户需求。近年来，由于半导体制造技术及微电子科学不断取得进步，使得芯片设计工程师能够将数千万甚至数亿颗晶体管集成到一个芯片上，并在其中设计了愈来愈复杂的功能，从而设计出一个更高复杂度和集成度的电子系统。于是，在现代电子系统设计领域中形成了一个新的研究热点，即片上系统 SoC (System On Chip) 设计。

我们可以把 SoC 简单的理解为在单颗芯片上实现了一个以嵌入式为核心的电子系统，集成了系统的关键部件，包含了软件和硬件部分，如图 1.1 所示就是一个蓝牙 SoC 芯片内部系统组成的架构图。SoC 系统集成了多个特定的电路功能模块，这些模块是之前已经设计好，并通过了相关测试和验证，即所谓的知识产权 IP 核。常见的可重用的 IP 可能是中央处理器、数字信号处理器、ADC、DAC、RF、模拟功能模块、数字功能模块等^[1]。同样，软件上可重用的库函数、操作系统、驱动程序也可集成于其中。SoC 系统通过集成原先多个分立的芯片或器件，提高系统的集成度，使系统内芯片的数量大大减少，从而减小了系统面积，系统的信号完整性更加优化，这样不仅使得系统的设计成本降低，而且大大提高了整个系统的性能。如今，片上系统正快速发展，得到广泛研究和应用，如应用于多核手机、智能家居、智能交通、社区安防、多媒体应用、可穿戴设备等领域，为人们的生活带来极大的便利。

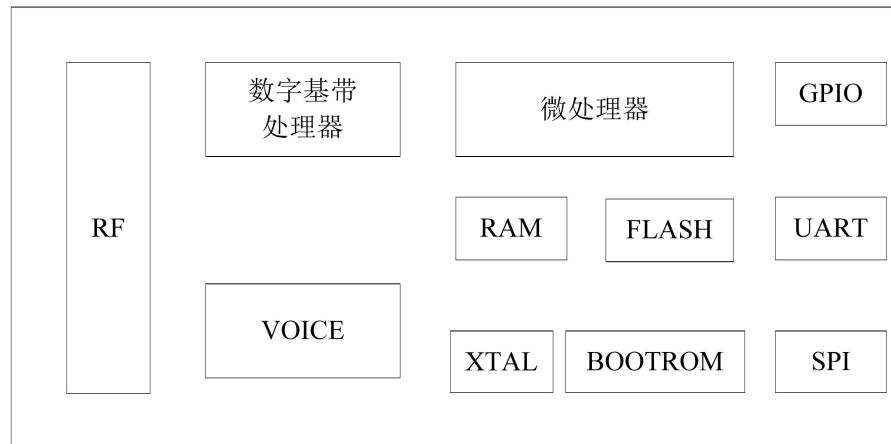


图 1.1 蓝牙 SoC 芯片系统架构图

现在的 SoC 设计除了开发可重用 IP 和前端设计，其后续的后端设计以及整个系统级验证也是其非常重要的内容。随着 SoC 系统的规模和设计复杂度的迅速提高，其功能验证的复杂度也成倍上升，使得其验证工作成为 SoC 设计过程中最耗时的工作。统计表明，IC 公司的验证工程师的数量至少是前端 RTL 设计工程师数量的两倍，并花费整个设计时间的 60%~80% 来对设计进行验证^[2]。据统计，近年来，SoC 芯片一次流片的成功率已经从 50% 下降到了 39%^[3]，如果一次流片不成功，那么就要花费数月的时间去修改设计和重新验证，这样不仅是花费了巨额投片成本，更重要是浪费了大量宝贵的时间，这对中小 IC 设计企业来说是一个较大的考验。因此，为提高流片的成功率，降低重复投片带来的风险，工程师不得不对 SoC 设计进行严格验证。考虑到 SoC 系统一般在实际环境下工作，因实际环境导致的功能性问题，不一定都能够通过相关仿真工具发现和解决。因此，为了缩短验证时间开销，有必要寻求一种在接近实际环境下对 SoC 系统进行验证的方式。现场可编程门阵列（FPGA）因为具有灵活的可编程性，并且它的运行速度可以接近实际的 SoC 系统工作速度，用它设计的验证平台可以非常真实地模拟 SoC 的功能和实际运行环境。能够帮助工程师较快地发现在仿真期间没有发现的设计缺陷和错误，尤其是时序违规问题，以便及时地修改设计。

随着半导体工艺和微电子设计技术的不断进步，FPGA 器件的密度和工作速度迅速地提高，目前已经能够在一颗芯片内部实现数千万个逻辑门，内部时钟频率甚至可以达到近 1GHz，其内部集成了很多成熟的 IP 核，如存储器控制器、数字信号处理器和嵌入式软核 Microblaze、PowerPC 等，并且支持丰富的高速 I/O 接口，如高速收发器 RocketIO、PCI_e、Ethernet 等。另外，FPGA 成本和功耗越来越低，而其性能和功能却在不断地提升，因此选择 FPGA 搭建硬件平台做 SoC 原型验证是比较理想的选择，对于流片成功率的提高具有重要实用意义。

1.2 国内外研究状况

随着 SoC 功能和复杂度的不断增加，投片费用变得非常昂贵，由设计缺陷导致的重复投片，延长了开发周期，浪费了资金，所以 SoC 验证构成了当前 SoC 设计的主要瓶颈^[4]。因此，怎样提高 SoC 设计的效率和产品的可靠性，是各个 SoC 设计公司面临的主要难题。于是，SoC 设计的功能验证也是目前国内外 IC 行业研究的重点之一，以寻求一种更加高效的、完善的验证方案。目前，关于 SoC 设计验证方法正处于研究阶段，也提出了相关技术和方案，并在实际中得以初步使用，其可靠性和有效性也在不断地完善中。

基于 FPGA 的特性，用其搭建的验证平台可以模拟 SoC 实际的运行状态，因而被许多公司所采用。有些公司专门从事通用 SoC 验证平台的设计和开发，而有

些 SoC 设计公司则根据项目需求和自身条件，自己设计合适的验证平台，并且成本开销可能比购买第三方平台更小。美国的 S2C 公司是全球著名的提供整体解决方案的企业，专注于在 FPGA 上实现从系统级到芯片级快速地整合 SoC 原型。2011 年，S2C 推出了具有 3280 万门的 SoC/ASIC 原型验证系统 Quad S4 TAI Logical Module，并后续发布了性能更加强大的 V6/K7 TAI Logical Module，在行业内得到广泛应用。业界主流的 SoC 设计和验证平台及工具都是由外国公司开发的，占据着市场主导地位，国内 SoC 设计及验证研究相对国外起步较晚，与国外同行差距明显。目前，国内的一些科技公司在 SoC 设计行业也取得了一些比较好的成就，典型的代表如在移动通信领域表现出色的华为、中兴、大唐等，以及在数字家电领域很成功的海尔、格力等。国内 SoC 设计产业正在快速发展，而相应的验证平台设计则相对滞后，主要依靠国外的产品。目前，国内也有一些公司和科研院所也在开发一些基于 FPGA 的 SoC 原型验证平台，如北京大学开发的基于 Xilinx XC2V2000 的 SoC 原型验证平台，主要面向大专院校、研究机构、企业等市场。电子科技大学也开发过类似的采用多片 FPGA 构成的验证平台，能够验证较大规模的 SoC 原型设计。

1.3 课题主要工作及章节安排

本文的选题背景来源于一些芯片设计公司对 SoC 及 IP 设计进行验证需求，要求设计出具有通用性和功能强大的验证平台。因此本文在对 SoC 验证方法进行研究的基础上，利用 FPGA 器件搭建了 SoC 原型验证平台，具有很高的实用价值。

本文的主要结构安排如下：

第一章主要阐述了课题的研究背景和意义，以及目前国内外本课题的研究和发展状况。

第二章主要研究了目前一些常见的 SoC 验证技术，对比了几种技术的特点，提出采用 FPGA 进行验证平台设计。

第三章主要介绍了 FPGA 器件的工作原理、内部结构和资源，以及基于 FPGA 的开发流程。

第四章主要对本平台的硬件设计进行了详细地说明，包括电源、时钟、存储器、通信接口等。

第五章主要介绍了本平台的原理图设计和 PCB 设计流程和要点，并对部分功能模块进行了调试和分析。

第二章 SoC 验证技术研究

2.1 验证概述

图 2.1 给出了 SoC 芯片设计的一般流程，可以看到从前端的行为级描述开始，到后端的硬件版图设计，在多个设计阶段都需要对芯片进行相应的验证。证明一个复杂的设计是否符合要求是验证的目的所在，所谓 SoC 原型验证，就是对 SoC 系统前端设计进行功能验证、时序分析、功耗分析以及功能检查等，在整个设计流程都要进行相关的设计验证^{[5][6]}。

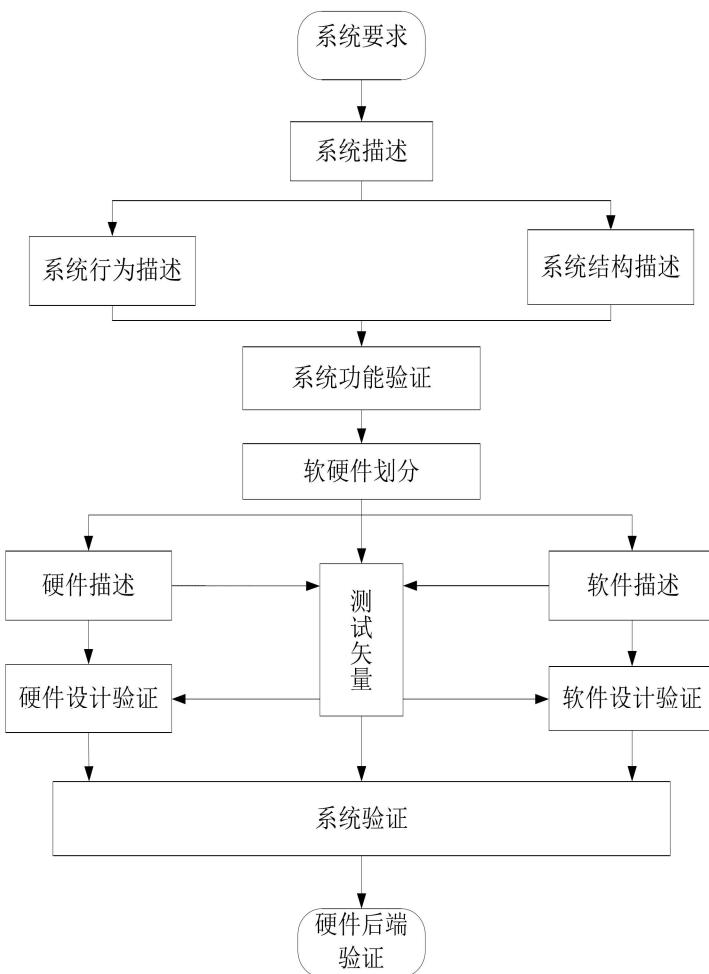


图 2.1 SoC 芯片设计流程图

验证和测试是有区别的，容易混淆，测试和验证二者之间的关系如图 2.2 所示，测试的目的是看设计是否被正确地“制造出来”，而验证是为了确保设计是否达到了设计说明所要求的功能和时序要求^[7]。在软件开发中，受技术手段的限制，我们不太可能去区分测试和验证，而 SoC 本质上是软件和硬件的结合，软件的运行需要硬件作为载体，并采用 EDA 软件和硬件平台两种环境来开发。虽然测试速度快，

但是很难达到高的覆盖率，可能会遗漏某些测试空间，因此在实际的 SoC 设计中用测试代替验证的做法是不可取的。

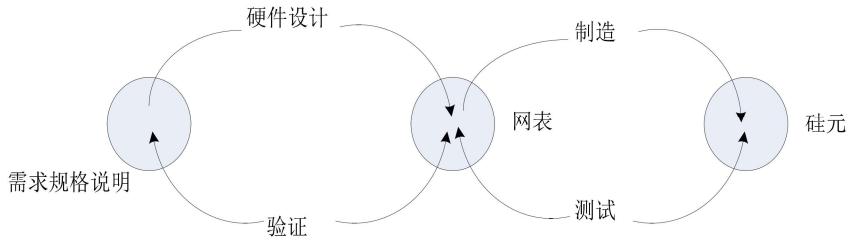


图 2.2 验证与测试关系图

可以这样认为，在 EDA 设计软件下做验证，硬件实现后做测试。EDA 设计软件不仅提供了硬件验证语言，还支持功能分析和自动检查特性。利用软件的灵活特性，我们可以达到很高的测试向量空间覆盖率和验证可控性，如果软件下所做的功能以及时序验证结果与板级硬件测试的结果一致的话，那么就可以认为这个设计就达到了要求。

2.2 常用验证技术

由于 SoC 的规模不断增大，投片风险也变得不可接受，设计的验证工作成为了 SoC 设计和开发的瓶颈，因此工程师们就不得不寻求快速、有效的验证方法，于是 SoC 设计领域出现一个重要的分支，就是 SoC 验证方法研究^[8]。目前业界没有统一的验证标准，在此对目前常用的验证技术做一总结，可以归结为几下几种：仿真技术、静态时序分析技术、软硬件协同验证技术、形式化验证技术、FPGA 验证技术。

(1) 仿真技术

仿真的方法是目前进行设计时常用的方法，一般是从电路的行为级描述构建出仿真模型，然后通过施加激励信号来模拟实际的物理输入信号，检查该模型在激励作用下的响应是否满足要求^[9]。其优点是有比较好的直观性，仿真结果输出一般为波形或者文字信息。这一技术是 ASIC 设计过程中应用较多的验证手段，现在在 SoC 芯片设计中也被广泛使用。在对 SoC 芯片中各个低层的基本功能模块或者 IP 进行验证的时候，工程师可以通过仿真结果来判断基本模块的功能是否完整和是否有逻辑设计错误，这是一个非常有效的手段。在设计所需逻辑门规模不是太大的情况下，门级仿真时间是可预知的。但是，随着 SoC 芯片的日益复杂化，输入激励信号的数量以指数的级数迅速增长，激励难以做到足够高覆盖率，也就是验证的不完备性，这样仿真的方法就不太适合做功能和时序的验证工作。另外逻辑门需求太大导致仿真时间也会变长，这可能是工程师所无法接受的。所以若要

继续使用仿真技术来验证 SoC 设计，就必须确保输入激励向量空间的完备性和提高系统的仿真速度。

(2) 静态时序分析技术^[10]

随着半导体工艺进入到深亚微米级甚至纳米级，元件的电气和物理性质会发生一些变化，导致内部的门级时延和布线时延对时序收敛的影响越来越显著。静态时序分析技术将电路中所有的时序路径提取出来，并分析出信号在所有路径上的时延情况，从而提取出不符合时序约束的路径。“静态”的意思是指这种分析方式与激励的是否输入是没有关系的，其目的就是要提取所有的信号传播路径，寻找所有输入组合下电路的最坏时延情况，主要是检查寄存器的数据建立时间和数据保持时间是否满足所设计的时序要求，而这两个时间信息的得到需要分别对最大路径时延和最小路径时延进行分析和计算。

静态时序分析曾作为一种辅助手段运用于 ASIC 设计中，同样，在 SoC 芯片的设计中，这种分析方法也扮演着重要作用。由于在进行超大规模 SoC 设计的验证时，仿真方式很难做到测试向量空间的较高覆盖率，不能发现某些路径上的时序违规问题，而静态时序分析克服了这点不足，能够比较完整地找出设计中的缺陷和错误，所以这一技术在 SoC 设计的时序分析中的重要作用被业界广泛重视。

(3) 软硬件协同验证技术

在硬件原型被生产出来之前，检查设计的软件是否可以在所设计的硬件上符合预期地运行，软硬件之间能否及时有效地进行数据交换，这一过程一般被称为软硬件协同验证。因为 SoC 芯片的内部集成了处理器核，系统的部分功能是通过软件来实现的，这一技术主要是验证硬件模块和软件模块之间接口的功能和有关时序关系，以获取整个软硬件设计中存在的缺陷。协同验证技术把一个专用处理器模型和一个嵌入式系统模型结合起来，而这个嵌入式系统模型是运行于逻辑模拟器件中的。其中，软件的设计和调试是基于系统虚拟的处理器原型进行的，而这个原型的搭建主要需要编译器、调试器和仿真器来完成。而对于硬件的设计和调试，主要是将硬件描述集成到虚拟原型中，并将经过软件调试通过的应用程序导入到硬件的测试文件中作为测试向量，从而进行硬件仿真，若仿真正确就说明整个硬件的设计是正确的，软硬件协同验证的简化框架如图 2.3 所示。

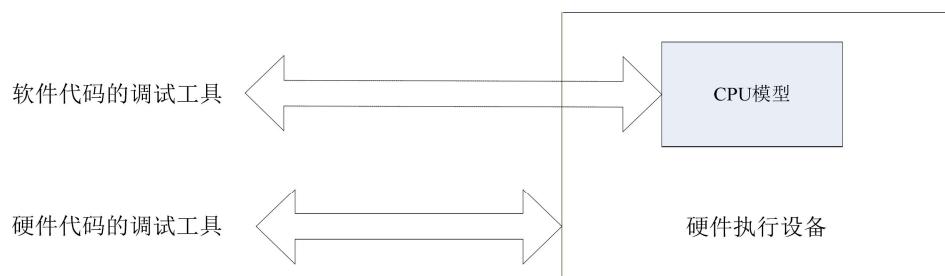


图 2.3 软硬件协同验证系统框架图

相对于传统的等到硬件原型生产出来之后再进行软件的设计，协同验证技术使得软件工程师可以提前利用虚拟原型进行软件设计和调试，因此在流片之前，如果检查出硬件方面存在问题，解决的办法就是修改硬件设计部分。因此，这个优点使得软硬件的并行开发成为可能，降低了首次将硬件和软件连接在一起出现意外的风险，从而大大缩短了产品的设计时间。

(4) 形式化验证技术

利用传统的仿真验证技术，首先是生成输入向量然后得到输出响应，但对于大型设计如 SoC 芯片设计，验证工程师不太可能考虑到所有的测试向量空间，而且如果使用的测试向量空间很大的话，要耗费的验证时间也将是很长的。而基于数学方法的形式化验证方式不需要关心输入激励，工程师先规定什么样的行为或特性是需要的，相应的验证工具可以遍历了整个状态空间，把所有有可能导致设计错误的状态条件找出来，从而实现对所做的设计进行比较全面地验证。

等效性检查和模型检查是形式化验证的主要内容^[11]。等效性检查采用静态的算法检查 RTL 设计代码和门级网表是否在功能上保持一致，通过读入两个设计，分析两者的数据结构，具有较高的准确性。不管网表如何被修改，它都以 RTL 代码作为参考模型，只要对 RTL 代码做相应的改动，仍然可以对两者进行等效性检查。模型检查是自动的基于模型的检验方法，其原理是首先将被测系统的有限状态模型建立起来，然后用相关算法检查模型中所有状态，观察它是否和待测的属性保持一致，最终如果有违背这些属性的执行路径就会被反馈出来^[12]。

(5) FPGA 验证技术

目前，FPGA 技术随着半导体技术的进步而日益提高和完善。由于 FPGA 的可编程性，其设计和验证工作变得更加具有灵活性。因此为了缩短产品设计时间，加快推入市场，越来越多的设计工程师选择利用 FPGA 来对设计进行全面验证。采用 FPGA 器件搭建一个硬件平台对 SoC 芯片设计进行验证，其费用开销与昂贵的流片费用相比要小得多。对于结构简单、规模较小的芯片前端设计，仿真通过后就可置于 FPGA 器件中运行，如果结果稳定、可靠，那么就可以进行其后端设计。对于大规模的 SoC 芯片设计，特别是涉及到功耗设计、数模混合设计、多时钟域设计，其验证方案将会变得复杂一些，一般要先对子模块或者 IP 进行单独验证，然后再系统整体验证。如图 2.4 所示是一种高效的、可靠的 FPGA 验证方式。

这是一个通用的 FPGA 硬件验证平台，能够比较真实地将 SoC 芯片的各项功能模拟出来。它把 FPGA 的综合与仿真环境，以及 IC 的设计环境统一了起来，通过比较 IC 设计的前端仿真、后端仿真以及 FPGA 在线仿真的结果，使得发现设计中存在的问题变得容易起来。同时，软件工程师可以并行地在该平台上开发和调试软件，通过加入一些监测模块，提取芯片内部运行数据，然后反馈给前端设计人员。

因为基于 FPGA 的验证系统能够与 SoC 系统实际的运行状态接近，二者工作速度也可达到接近^[13]，所以基于这个优越性，本文采用 FPGA 器件搭建了 SoC 原型验证平台，为 SoC 原型验证提供软硬件环境。

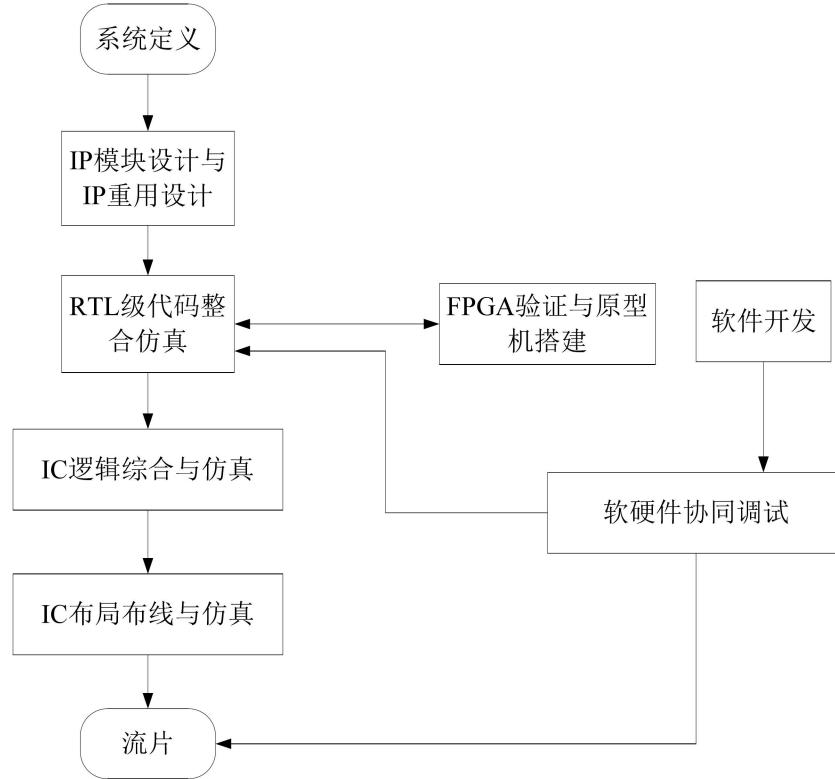


图 2.4 FPGA 验证方式流程图

2.3 验证层次

由于 SoC 芯片规模较大，一般采用自顶向下的设计方法^[14]，总体设计师首先把设计目标按照某种原则进行划分形成子模块，这些子模块再往下细分，最终从比较简单的低层模块开始设计。为缩短开发周期，SoC 验证工作也是伴随设计工作同步进行，相对于 SoC 设计的自顶向下的设计层次，对其验证的过程也分为几个层次，分别为：系统级、门级、物理级^[15]。

系统级验证主要是验证芯片的原型设计在总体功能上是否正确，这时并不做时序检查。一般通过 C 语言或者硬件描述语言来设计激励向量，然后在相关仿真工具的作用下，模拟 SoC 芯片中代码的运行情况，可及时地发现设计初期功能上的缺陷。

系统级验证若没有问题，通过 EDA 软件对设计代码进行综合处理，得出其门级网表，之后可以进行门级验证。此时使用与系统级验证相同的激励，但这时由于门时延的存在，就必须考虑时序是否满足的问题。有时会出现因寄存器的建立和保持时间得不到保障而出现门级验证无法通过的情况，我们可以通过修改门级

网表来解决时序违规的问题。

最后的验证层次是物理级验证，一般是在 SoC 设计完成布局布线之后进行的，是为了保证芯片设计的最终物理实现不会违反相应的物理规则，解决一些信号完整性问题，如信号串扰、反射、功耗等问题。

2.4 本章小结

本章首先从 SoC 芯片设计流程引出了芯片验证的概念，并与测试的概念进行了比较，并说明了验证的必要性。针对目前常见的五种验证技术：仿真技术、静态时序分析技术、软硬件协同验证技术、形式化验证技术、FPGA 验证技术，分别进行了简要地说明，并简要地介绍了验证的几个层次。

第二章 SoC的验证

2.1 什么是验证

在SoC的设计过程中，有关验证方面的工作要贯穿整个设计流程。从前端行为级HDL设计开始，一直到后端版图设计，在芯片流片之前都需要做足够多的验证流程。当前验证工作已经占整个设计工作50-80%，因此验证是非常重要也是比较繁杂的。

2.1.1 验证的定义

Janick Bergeron 给“验证”下的定义是：“验证是证明一个设计的功能是否正确的过程”。使用汇聚模型可以从概念上清晰地描述验证过程，如图2.1.1所示，图中Transformation（转换）可以是任何包含有输入输出的过程，例如根据Specification（规范）写出RTL代码、扫描链插入、把RTL级代码综合为门级网表、根据门级网表布局布线等。Verification过程是一个相反的过程，它从Transformation的结果出发回到起点。

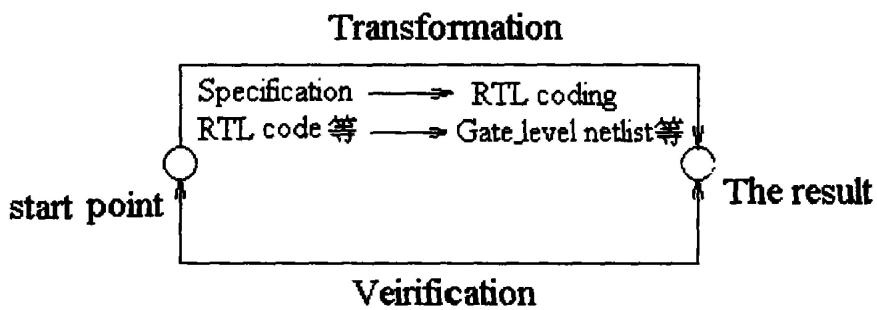


图2.1.1 汇聚模型

在实际验证工作中，一般采用由验证平台TESTBENCH和DUV(Design Under Verification)组成的验证体系，如图2.1.2所示。这是验证系统普遍适用的模型。验证平台主要包括激励生成(Stimulus Generation)和响应检验(Response Check)两个部分，激励部分用于对DUV提供激励，响应部分用于对DUV的输出进行处理。其中，激励又可分为两类：(1) 确定性激励。用于测试设计的典型功能或具体的角落实例(Corner Case)。(2) 约束驱动的随机激励(Constrained Random Test Vector)。而响应检验则主要通过以下两种方法实现：(1) 对待验证设计的黄金

模型(Golden Model)，通常是经过验证的设计的高层次行为模型和其自身施加相同的激励，并比较两者的输出。(2)通过专门的检验器监视和检查响应输出是否与设计规范相一致。验证平台与DUV一起构成一个闭合系统，对外没有输入，也没有输出。

验证工程师所面临的挑战就是如何确定DUV的输入模式，并判断DUV输出的正确性。设计团队首先需要根据设计规范开发相应的模块级(Block Level)和系统级(System Level)验证计划。验证计划通常包含验证设计功能正确性的各种角落实例和覆盖率需求。接下来，根据这个计划开发验证平台并对各个层次上的验证进行设计。这项工作将一直持续到满足测试计划的全部要求为止。

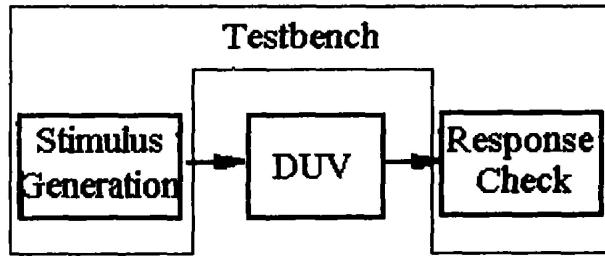


图2.1.2 验证体系

2.1.2 设计过程中的人为因素

如果Transformation过程不是自始至终全部自动完成的，那么就要由人去理解规范，并进行Transformation，RTL代码的编写就属于这种情况。设计团队根据对书面规范的理解，编写他们认为功能正确的可综合代码。通常进行验证时，是由写代码的工程师验证所写代码的功能正确。图2.1.3画出了这种情况下的人为因素。

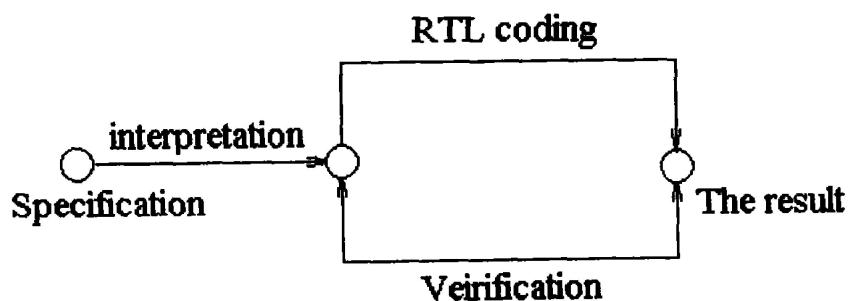


图2.1.3 同一个人完成代码设计和验证的汇聚模型

如果编写RTL代码前还需要理解规范，同时编写代码和验证工作由同一个人完成，那么验证的将是对规范理解后的实现，而不是规范本身的实现。也就是说如果理解上出现错误，将不能通过验证发现错误。

转换过程中任何人为因素，都是不确定性和不可重复性的来源。因此，排除人为因素导致的错误非常重要。在设计过程中排除人为错误的方法主要包括 Automation、Poka-Yoka 和 Redundancy 三种。

(1) Automation (自动化) 法。Automation 排除了人为的干预，但对于未能清楚定义的情况，以及象硬件设计这种需要人的智慧与创造性的工作，不可能全部自动化。

(2) Poka-Yoka 法。该法按照简化及标准化的原则，将整个工作过程分步实现，并且每一步都是极其简单不容易发生错误的，人只需根据希望的结果决定步骤的顺序。这要求对整个工作过程进行完整的、标准化的定义。根据目前的验证技术，还不能对验证过程做如此的定义，因此这种方法还不能用在验证工作中。

(3) Redundancy (冗余) 法。该法是去除人为错误的最后选择，这是一种最简单但是成本最高的方法。它要求双倍的转换资源。由一个工程师完成的设计转换要由另外一个工程师进行设计验证，或者由两个工程师同时进行转换，然后比较结果是否一致。这种方法一般用于对可靠性要求很高的系统，或者事后重新设计和替换缺陷部件的成本高于 Redundancy 本身的情况下，例如 ASIC 设计。图 2.1.4 所示为规范存在不确定性时使用 Redundancy 减小理解错误的汇聚模型。在硬件设计中，相应的转换过程为根据书面规范编写 RTL 代码，使用这种方法意味着需要一个专门的人做验证工作。

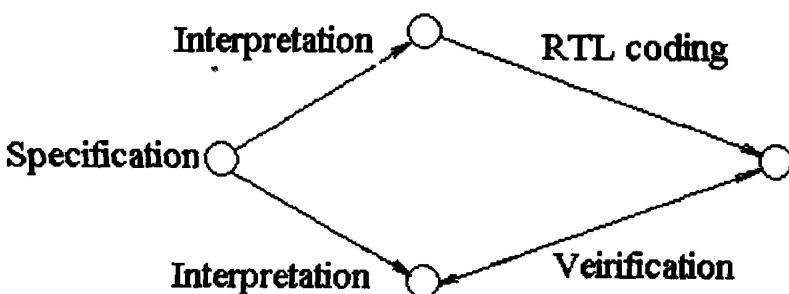


图2.1.4 使用Redundancy减小理解错误的汇聚模型。

2.1.3 功能验证

选择不同的起点和汇聚点能够决定不同的验证对象，而起点和汇聚点经常由验证工具所决定。因此，为了明确验证的对象必须先要清楚验证的起点和汇聚点。功能验证（Functional verification）由于起点和汇聚点的不同，验证的对象也各不相同。

功能验证能保证设计预定功能的实现。该验证过程显示出设计符合规范的程度，但是不能证明设计实现了规范。通过功能验证，只能证明设计存在bug，不可能证明设计不存在bug。功能验证的汇聚模型如图2.1.5所示。

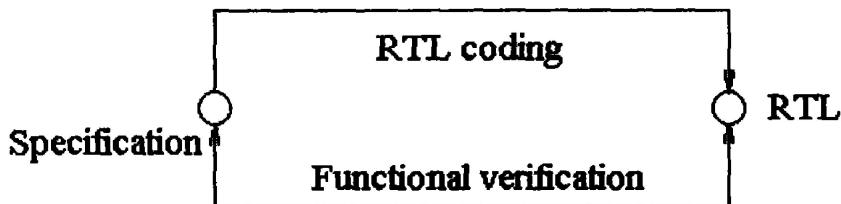


图2.1.5 功能验证的汇聚模型

功能验证主要包括：

(1) 黑盒验证 (Black-box Verification)

它通过设计顶层接口，验证那些与设计实现技术（ASIC, FPGA或软件）无关的功能。黑盒验证不直接访问设计的内部状态。功能验证可以和设计实现并行进行，但是它很难进行功能隔离（可控性差），很难发现问题的来源（可见性差），因此也就不能对设计进行全面的验证。对于大型设计，需添加一些与功能实现无关的设计，以增强验证可控性及可见性。如：增加软件可访问的寄存器及数据处理量的控制。

(2) 白盒验证 (White-box Verification)

它保证设计实现技术（ASIC, FPGA或软件）相关的功能正确实现，是黑盒验证的补充，对于设计的内部结构及实现是完全可控和可见的，但不可移植。

(3) 灰盒验证 (Grey-Box Verification)

它根据设计的内部结构写Testcase（测试实例），从设计顶层接口进行控制和观察，Testcase 的目的是验证某种设计方法是否实现了一些主要特性，而不关心其它的设计方法。

2.1.4 验证面临的新挑战

SoC和基于重用的设计方法学的出现，对传统验证方法提出了新的挑战，主要体现在以下三个方面：

(1) IP模块失去了原有的可控性和可见性

在传统的IP单独验证中，IP的输入/输出端口可以直接被验证向量所直接访问。而在SoC验证过程中，单个IP模块被嵌入到SoC中，此时的输入/输出端口已变成了IP模块的互连，不可能在对SoC的访问过程中，直接访问原来的IP模块的输入/输出端口。因为此时的IP模块输入/输出端口已在SoC内部，从SoC的外部输入/输出端口是无法直接访问的。IP模块失去了原有的可控性和可见性，从而使得SoC中的IP模块的验证变得困难。同时它还意味着要实现IP模块的验证重用，必须要为单个的IP模块提供相应的验证访问通道。

(2) 验证平台的构建和重用问题

在基于重用的设计方法学中，作为独立设计的IP需要验证，被集成到SoC中的IP也必须得到验证。然而开发多样、独特的验证平台需要花费非常多的精力，同时，SoC集成者对第三方IP缺乏足够了解或相关专业的知识。因此，与重用IP来设计SoC一样，其他验证团队也希望能够重用IP验证平台中的部分或全部元件来构建SoC验证平台。验证平台的重用是一种有效的验证策略，这种方法允许设计者对于任何层次、任何应用的设计都可以用最小的花费、最大的连贯性开发高质量的验证平台。但是在传统验证方法的许多情况下，验证平台是在信号级(Signal Level)的接口上直接与DUV相通讯，即使用激励直接驱动DUV的引脚，并通过检查接口信号的值和变化达到验证设计功能的目的。这种验证方法的抽象层次较低，平台的开发与设计的接口协议紧密相关，从而就使得验证平台的重用变得非常困难。

(3) 验证的概念和技术涉及广泛

验证的目标要确保设计在任何合理的配置和应用中都是零缺陷。为此，验证工程师必须要开发出一套高质量(高覆盖率)的测试(激励)集(Test Suite)。由于传统功能验证的抽象层次较低，随着设计规模的不断增加，其测试集的开发存在周期长、代码过于庞大等缺点，很大程度上影响了产品的质量和产品的设计周

期。因此，我们所说的验证是一个包含众多方法和工具在内的广泛意义上的概念（如，形式化验证、硬件仿真、物理原型验证等都属于该范畴）。

SoC验证这个领域所涉及的内容很广，种类也很多，如：IP核/模块级验证（Block-Level Verification）、系统级验证（System-Level Verification）、仿真验证（Simulation）、软硬件协同验证（Hardware/Software Co-verification）、等价性检查（Equivalent Checking）、静态时序分析和时序验证（Static Timing Analysis & Timing Verification）、版图验证（Physical Verification）等。随着验证技术的逐步发展，验证方法由最初的直接测试向量生成（Directed Test Vector Generation），到约束随机测试（Constrained Random Test），再到覆盖驱动验证（Coverage-driven Verification），一直到最新的基于断言的验证方法（Assertion-based Verification），各种验证方法还在不断创新和发展。

2.2 验证工具

改善验证效率和可靠性的一个方法是自动化。目前自动化工具很多，有些是功能验证必不可少的，例如仿真器；有些工具可以代替人完成最繁琐的工作，并能提高功能验证的可信度，例如linting和代码覆盖率工具。

2.2.1 Linting tool

Linting tool是根据设计的RTL描述代码结构做静态分析，推断描述代码存在的逻辑错误，但无法决定描述代码是否能够现实设计要求的功能。linting tool可用于强制代码遵从编写规范。由于Linting工具是静态验证工具，因此运行速度快，可以节省时间。其缺点是多疑问，误报率高，会产生大量不存在的错误警告，因此要对不存在的错误警告仔细过滤，注意避免滤除真实的错误警告。变量名最好符合命名惯例，这样有助于过滤。

对于Verilog描述的设计，因其语言特点，Linting tool是一种有益的验证工具。它可以检测race conditions及数据宽度不匹配，可保证Verilog正确描述的数据处理过程，避免造成数据的弃位及增位现象。通常这种错误通过仿真并不一定被发现。对VHDL使用Linting tool的作用不如对Verilog语言那么明显，但

Linting tool还是能发现一些潜在的问题。

2.2.2 仿真器

仿真器通过忽略及简化设计的物理特性，对设计的实现进行模拟。仿真器通过执行RTL级的设计描述，模拟设计的物理实现，但它无法确定设计真实的物理实现与设计描述之间的区别。仿真的结果取决于设计描述是否准确反映了设计的物理实现。仿真器不是一个静态工具，需要激励(Stimulus)和响应(Reponse)。激励由模拟设计工作环境的Testbench产生，响应为仿真的输出，由设计者确定输出的有效性。

仿真器的类型分为4种类型，事件驱动仿真器(Event-Driven Simulator)、基于周期的仿真器(Cycle-Based Simulator)、联合仿真器(Co-Simulator)、硬件模拟器(Hardware Modeler)，分别介绍如下：

(1) 事件驱动仿真器

事件驱动仿真器可将信号的变化定义为一个事件，该事件驱动仿真执行。事件驱动仿真器能准确地模拟设计的时序特征，可模拟异步设计。事件驱动仿真器的工作过程如图2.2.1所示。

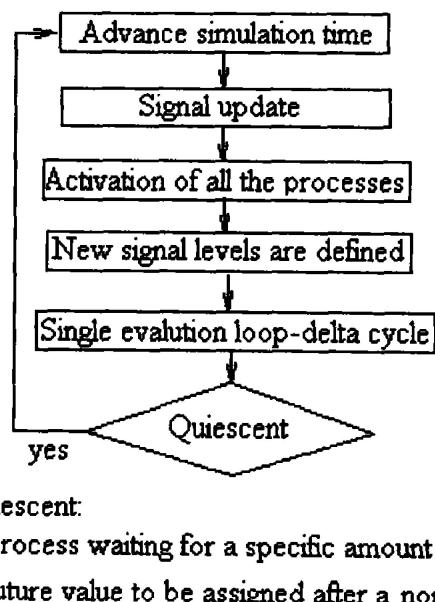


图2.2.1 事件驱动仿真器的工作过程

(2) 基于周期的仿真器

基于周期的仿真器的特点是忽略设计的时序，其仿真速度比事件驱动仿真器要高。基于周期的仿真器的工作过程步骤是，首先编译电路，将组合逻辑压缩成单独的表达式。根据该表达式可确定寄存器的输入，然后执行仿真。遇到时钟的有效沿，寄存器的值被更新。基于周期的仿真器的缺点是不能仿真异步电路，不能进行验证设计的时序。

(3) 联合仿真器

联合仿真器可以对同一设计各个部分分别用不同的仿真器进行仿真。如对既含有同步设计又含有异步设计的电路，可用事件驱动仿真器对异步设计仿真，用基于周期的仿真器对同步设计仿真。联合仿真器中各个仿真器的操作是locked-step的，类似于电路的流水线（pipeline）操作。其缺点是由于不同仿真器之间需要同步和相互通讯，联合仿真器的仿真速度受到最慢仿真器的限制，因而影响仿真器的性能，而且在各仿真器传送的信息会产生多义性。

(4) 硬件模拟器

硬件模拟器是创建一个物理芯片的逻辑模型，向仿真器提供该芯片的行为信息。芯片和仿真器的通信过程，是首先将物理芯片插入硬件仿真器，然后格式化来自仿真器的数据，作为该芯片的输入，最后将该芯片输出的数据，包含时序信息，送往仿真器。硬件模拟器可以提供很高的仿真速度，但是设备价格高昂。需要注意的是，硬件模拟器做的仍然是功能仿真，而不是时序仿真，因为芯片是降频运行的。图2.2.2为硬件模拟器系统框图。

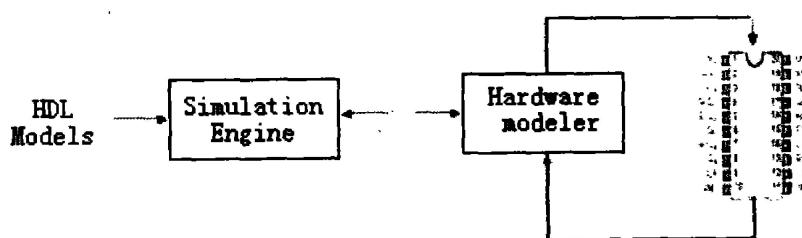


图2.2.2 硬件模拟器系统框图

2.2.3 波形观察器

在仿真调试的过程中，波形观察器是必不可少的工具。它能提供信号状态

和变化的详细信息。但是，波形观察器不能用来判断一个设计是否通过验证，因为信号波形是不可重复的，且无法用于递归仿真。

波形观察器的优点，是可以观察仿真的整个过程，有利于设计及testbench的诊断。缺点是由于要输出波形，影响了仿真的速度，因此应尽可能限制在波形图中显示的信号数量及时间长度。

波形观察器的另一个作用是波形比较。在波形比较中，不能仅看表象，需仔细分析，确认波形之间存在的差别是有意义的。例如，有时我们仅关心波形 transitions（转换）之间的相对位置，而不关心它的绝对位置。图2.2.3为波形观察器的工作流程。

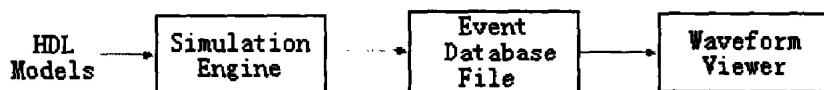


图2.2.3 波形观察器的工作流程

2.2.4 代码覆盖率

代码覆盖率（Code Coverage），可以指示Verilog代码描述的功能有多少被验证。代码覆盖率有三种计算方法，第一种是语句覆盖率（Statement Coverage），它用来指示验证过程中，设计代码被执行的语句数量。有时会添加一些没有具体设计意义的语句，仅用于监视代码执行过程中的异常或标注一些设计中的假设。这些语句可不参与语句覆盖率的统计。第二种是分支覆盖率（Path Coverage），它在设计中往往通过分支控制语句来完成对功能的控制。可以将所有分支控制语句的控制状态进行组合，产生一定数量语句执行分支路径，分支覆盖率指示所有的语句执行分支路径是否都得以执行。第三种是表达式覆盖率（Expression Coverage），它用来指示分支控制语句的控制条件是否全部有效。

需要注意的是，百分之百的代码覆盖率仅仅表示了代码都已被执行，并不能证明设计的正确性。代码覆盖率可以用于衡量验证工作是否完成。

2.3 可供选用的验证技术

验证的目的，是要确保设计对象满足在设计规范中所定义的功能要求。目前验证技术有许多种，它们大致可以分为四类：基于仿真的技术、静态技术、形式技术、物理验证与分析技术。要实现系统芯片的设计目的，必须将这些技术组合起来使用。

2.3.1 仿真技术

仿真 (Simulation) 技术是设计验证的主要形式。它可以分为两类：

(1) 从仿真的抽象层次来看

包括基于事件的仿真、基于时钟周期的仿真、基于对象转换的仿真。

① 基于事件的仿真

基于事件的仿真器把输入激励的变化认为是事件的触发，每一个仿真时间仿真器处理一个事件触发，根据事件触发的内容对整个设计重新计算，直到一个仿真稳态出现为止。如果输入的信号在一个时钟周期内到达，但是不同的输入信号达到的时间可能不尽相同，则基于事件的仿真机制决定仿真器在一个周期内要计算多次。

设计基于事件触发的仿真模型既包含了功能模型，也包含了时序模型，它为设计提供了更为精确的仿真环境。基于事件触发的仿真特点是同时覆盖了设计的功能和时序模型，仿真结果精确，它非常容易探测到设计中的毛刺电路，尤其适用于异步电路的仿真。然而，基于事件触发的仿真由于算法复杂而导致仿真速度较慢，不太适用于规模较大的电路仿真。

② 基于时钟周期的仿真

基于时钟周期的仿真器在一个时钟周期的时间内没有时间的概念，它只在时钟的上升沿或下降沿进行触发，每一个时钟周期的时间对电路计算一次。基于时钟周期的仿真技术特点是忽略设计的时序，假定所有寄存器的建立时间(setup time)和保持时间(hold time)都满足要求，在一个时钟周期，信号仅更新一次，从而信号必须与时钟同步。

基于时钟周期的仿真可以在一定程度上提高电路的仿真速度，但是它只适

用于同步设计的电路仿真，对于异步电路则有可能产生错误的仿真结果。

③ 基于对象转换的仿真

基于对象转换的仿真以数据包、图形、语音等对象作为直接的仿真激励，而不再是添加到设计引脚的激励波形。总线功能模型(BFM)在这里是必需的，总线功能模型提供了对象激励到设计引脚激励波形的转换接口，它的实现是根据设计的对外接口协议，使用HDL语言或C++进行行为级描述。

基于对象转换的仿真技术通过提高仿真激励的抽象层次，大大提高了仿真工作的效率，使得大规模的逻辑电路设计采用仿真的手段来完成自检查测试和大流量随机性测试，变得更加容易实现。

(2) 从仿真的活动内容来看

包括代码测试覆盖率检查、基于行为级模型的比较仿真、软硬件协同仿真、硬件加速器和快速原型系统仿真。

① 代码测试覆盖率检查

代码测试覆盖率检查，可以统计当一组测试用例作用于设计时各种各样的测试覆盖情况：statement, toggle, FSM, visited state, triggering, branch, expression, path and signal等。利用这些数据，设计者可以对测试用例对设计的覆盖情况有所了解，同时还可以确定设计还有哪些功能没有得到测试。测试激励的代码覆盖率至少要达到95%以上，才能基本认为代码在逻辑上是通过质量控制的，才能进入综合步骤。这一方面帮助设计者可以对测试功能覆盖情况有一个量的认识，另一方面方便设计者发现冗余逻辑和设计错误。

代码覆盖率是保证高质量代码的必要条件，但却不是充分条件。即便代码行覆盖和分支覆盖都能够达到100%，也不能肯定的说代码已经得到100%的验证，除非所有的分支覆盖都能够进行组合遍历。

② 基于行为级模型的比较仿真手段

首先建立设计的行为模型，然后对设计的RTL实现和行为模型使用同样的激励，对比两者输出的结果。设计的行为模型不考虑具体的实现结构，开发周期短。实现同样的功能，用行为级模型描述比RTL描述在代码的复杂度上能大大简化，因此也减小了行为级模型代码出错的概率，它便于产生复杂的测试环境。建立行

为模型的语言可以是HDL，也可以是SystemC、C、C++，验证语言Vera和SpecmanE等。

③ 软硬件协同仿真

主要适用于验证SoC设计。传统的嵌入式系统设计，硬件和软件的开发相对独立，只有等硬件单板完成后才能够将软硬件结合起来进行系统验证。软硬件协同仿真工具提供一个集成的软件环境，工程师可以把硬件的RTL级设计和嵌入式软件在同一个软件环境下进行同步调试。这样，软件工程师可以较早地在硬件设计上直接调试软件，硬件工程师可以较早地得到更加真实地输入激励并检验效果。目前，由于软硬件协同仿真的速度非常慢，这种仿真手段还局限于对软硬件接口的调试，一般并不用于对整个系统进行实时仿真。

④ 硬件加速器

硬件加速技术，是将软件仿真中全部或部分元件映射到特意为加快某些仿真操作的速度而设计出来的硬件平台中。普遍的做法是：测试平台仍以软件形式运行，而实际的待验证设计对象在硬件加速器中运行。商用的硬件加速器，通常采用FPGA阵列和高速的处理器阵列和相应的系统软件特殊设计加以实现。被验证的目标设计及其测试环境，通过HDL语言编程用FPGA阵列实现。仿真速度很快，甚至可以直接放到目标设计最终应用到的系统环境中进行测试。

⑤ 快速原型系统

快速原型系统（RPS，Rapid Prototype System），可用来为预期的系统芯片精确的建立快速原型模型。它是待验证的设计对象的一种硬件设计表示形式。成功建立快速原型的关键，在于要快速地将原型实现出来。可用的一些途径（如仿效系统和可重配置原型系统），都是将目标设计对相映射到现有器件之上，如控制处理器、DSP、引出焊点式（bonded-out）芯核和FPGA。这些元件安装在子板之上，而该子板插在含有用来模拟目标系统的、互连的客户自定义可编程器件的系统互连母板中。原型验证系统最常见的是用FPGA验证目标设计，首先用FPGA实现并放到应用系统环境中进行测试。对于嵌入式的SoC设计，其原型验证系统可以是嵌入式系统插板，它极有可能是包含有MCU、DSP、存储器件、实现IP宏或定制设计的一个或多个FPGA系统。

快速原型系统提供了开发和调试软件的能力，提供了给软件进行调试的真实硬件环境。这样在有了芯片原型的时候，就能够实现硬件和软件的无缝集成，其提供的仿真速度比软件仿真和协同仿真快的多。

2.3.2 静态技术

静态技术包括静态(lint)检查和静态时序分析(STA, Static Timing Analysis)。用静态技术来实现验证不需要验证平台和测试向量。

(1) 代码静态(lint)检查

对设计对象的代码作静态检查来验证其在语法上的正确性。一般是在设计流程的早期进行代码静态检查，查出设计代码中的简单错误。从而避免使用更高级的工具而耗费大量的时间。可能出现的错误类型有：未初始化的变量、不受支持(语法)结构和端口不匹配等。

2、静态时序分析

静态时序分析是指按照同步电路设计的要求，根据电路网表的拓扑结构，计算并检查电路中每一个存储器件如触发器或锁存器的建立时间和保持时间，以及其它基于路径的时延要求是否得到满足。静态时序分析技术的特点是只根据电路结构“静态”分析，判断设计在时序上是否满足，不需要测试向量。只适用于对同步设计的电路进行时序验证，而不能保证电路实现的功能是否正确。目前较通用的STA工具有Synopsys公司的Prime Time, Cadence公司的Peral, Mentor公司的SST Velocity。

2.3.3 形式验证

形式验证是一种系统验证手段。它一方面指利用边界条件从理论上推导、论证设计实现的正确性，另一方面通过分析对比两个设计，论证两个设计实现的功能是否一致，形式验证主要用来判断两个设计的等价性。形式验证的代码覆盖率好，但涉及较复杂的数学推导，推导本身的正确性难以把握。而且，系统较复杂时，形式验证会成为整个项目进展的瓶颈。较好的形式验证工具有Mentor公司的FormalPro, Synopsys公司的Formality, Cadence公司的FormalChecke。

目前，主要有三种形式验证方法，它们是定理证明方法、模型检验方法和

等价性形式检查方法。

(1) 定理证明 (Theory Proving) 方法

定理证明是半自动的基于证明的验证方法。它的基础是一个形式化的逻辑系统，该逻辑系统由一系列公理和推理规则组成。定理证明的过程是从系统的公理出发，使用推理规则逐步推导出规范成立的证明过程，可用于验证VLSI电路，主要缺点是验证方法与被验证系统相关。

目前，定理证明方法已经广泛应用于工业级的软硬件设计验证中，比较有名的定理证明系统有HOL (Higher-Order Logic)，HOL是由英国剑桥大学研制的交互式高阶逻辑定理证明系统。在HOL系统中，所有推理规则都是由8个初始推理规则派生出来，证明过程每一步都由用户选择推理规则，然后由HOL系统检查可行性并进行推理。

(2) 模型检验 (Model Checking) 方法

模型检验是完全自动的基于模型的检验方法。它用于检测设计是否违反用户定义的设计行为规则，这些行为规则用断言 (Assertions) 来定义，如状态机的状态跃迁，接口的握手过程等。模型检查的难点在于定义需证实的断言，一般对定义的断言，仅能证实它的一个子集。当前的技术还无法证实保证复杂功能实现的断言。目前，根据核心算法不同，模型检验方法主要分为四大类：显式模型检验方法，基于决策图的模型检验方法，基于SAT的模型检验方法和基于符号模拟的模型检验方法。图2.3.1为模型检验的汇聚模型。

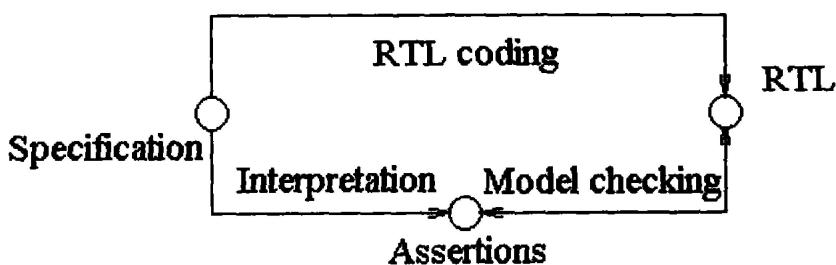


图2.3.1 模型检验的汇聚模型

(3) 等价性检查 (Equivalence Checking) 方法

等价性检查是证明同一个设计对象的两个不同视图彼此等价的一种方法。它通过一定的算法，来证明设计实现的逻辑一致性，保证设计的功能在实现过程中没有改变。其汇聚模型如图2.3.2所示。

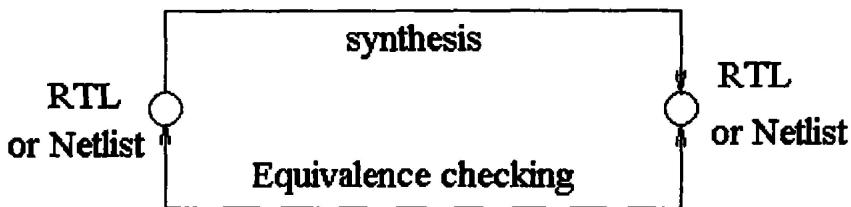


图2.3.2 等价性检查的汇聚模型

等价性检查可以用于比较两个RTL代码文件，以验证经过修改的RTL代码的功能和原来相同。也可用于比较两个网表，以保证一些网表后处理，例如扫描链插入、时钟树综合和手工修改等没有改变电路的功能。还可用于验证网表的功能是否与原来RTL代码的功能相同，如果认为综合工具是可靠的，那么这个过程可以省略。但是综合工具是个大型的软件系统，其可靠性依赖于算法和器件库的信息，综合工具发生错误的事情并不少见。等价性检查可以用于保证综合工具的可信性。例如，假设设计中含有超过48 bit的算术运算，在综合时，综合工具用 synthetic lib中的算术运算符进行映射。但在 synthetic lib的文档中指明该算术运算符不能超过48bit，对于这种bug，可利用等价性检查进行检测。

在应用形式验证时，它不需要开发测试平台和测试向量，但必须有一个功能完全正确的参考设计。它利用数学技术来验证参考设计与目标设计是否等价。该方法可以用来验证RTL-RTL、 RTL-Gate、 Gate-Gate之间是否等价，它比穷举式仿真更快，能保证实现100%的验证覆盖率。但是它不验证设计对象的时序，因此必须和时序分析工具联合在一起使用。

2.3.4 物理验证与分析技术

在深亚微米设计中，由于互连线延迟所产生的问题相对于门延迟所产生的问题而言，已经成为设计中主要需考虑的问题，所以必须将所有的电学问题和工艺都纳入统筹考虑，并要重点解决互连线寄生效应。

需要考虑和必须分析解决的问题有：时序 (timing)、信号完整性 (signal integrity)、串扰 (crosstalk)、IR电压降 (IR drop)、电迁移 (electro migration)、功耗分析 (power analysis)、工艺天线效应 (process antenna effects)、相移掩模 (phase shift mask) 和光学临近效应修正 (optical proximity correction)。

物理验证与分析技术主要是进行设计规则检测(DRC)和版图与原理图对照(LVS)。根据设计的不同有两种物理验证工具流程，即交互式(单元/模块)和批处理(大型模块/全芯片)验证工具，每个设计流程根据设计元件的类型和设计人员使用工具的方式而选择不同的工具。SoC设计要求在交互式和批处理两个阶段都要进行DRC和LVS。工业界现有的做法是，在版图之前的设计阶段估算各种物理效应，在版图之后的设计阶段再提取和分析这些物理效应。

2.4 常用的验证策略

常用的验证策略包括：自顶向下的验证、自底向上的验证、基于平台的验证和基于系统接口的验证。

2.4.1 自顶向下的验证流程

自顶向下的验证流程如图2.4.1所示。

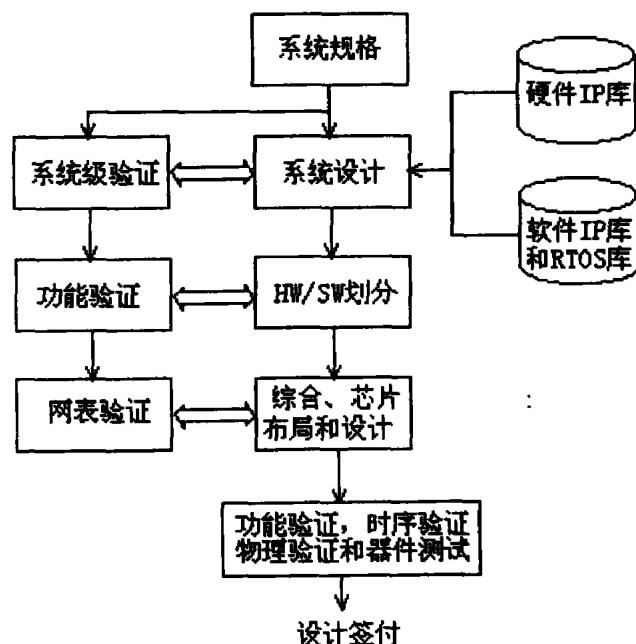


图2.4.1 自顶向下的验证流程

该验证流程的起点是设计规范书，从设计的规范书可以提取设计需要验证的所有特性。这些特性的表现形式可以用计算机语言进行描述并形成可执行的规范，但更多的时候这些需要验证特性是用自然语言列表给出的。接下来是开发验证计划，有关验证计划的内容在后面讲述。

系统级验证的内容主要包括建立系统级行为模型，并把它放到系统级的测

试环境中进行验证。建立系统级行为模型是指用系统级语言C或C++建立设计规范书定义的功能特性模型。系统级的测试环境应该是模块化的，可以很方便地移植到对设计的功能验证和网表验证，对于不同的抽象层次可根据该层次的具体要求增强相应的测试功能。一般推荐利用基于对象转换的仿真技术建立系统级测试环境，一方面它可以很方便地进行系统级的验证，另一方面可以直接用于对设计顶层模块间连接关系的验证，从系统级验证移植到功能验证。

网表验证既可以采用形式验证的手段，也可以采用门级仿真的手段，来验证设计的综合实现和物理实现是否正确。

自顶向下的验证最后的内容是时序分析和物理验证。值得注意的是，采用自顶向下验证策略的验证过程，验证工程师会周而复始地发现问题。自顶向下定位出现的问题，与相应的设计工程师讨论问题产生的原因，然后再由设计工程师修改错误，这样通常会导致较低的验证效率。

2.4.2 自底向上的验证流程

自底向上的验证流程如图2.4.2所示。

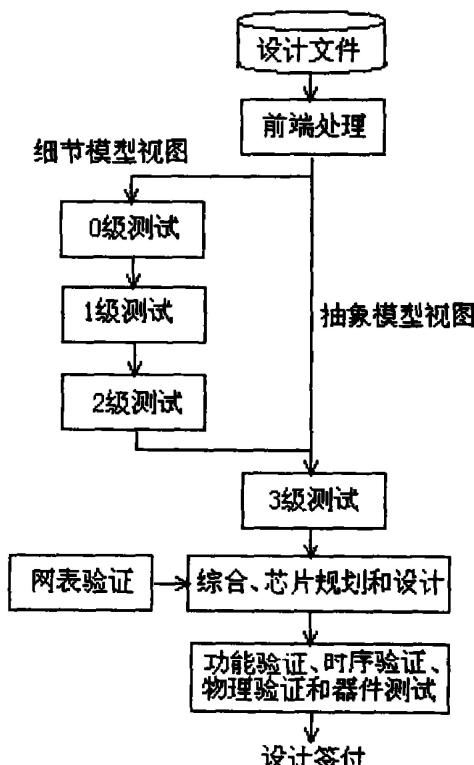


图2.4.2 自底向上的验证流程

图中0级测试主要是孤立的验证各个元件、功能块或单元。验证的思路是对元件作穷举测试，而不考虑其集成环境。0级测试及相关的测试平台可由IP供应商提供，在这种情况下，0级测试可包括代码覆盖状况分析，以确保设计对象的品质。

1级测试由设计工程师负责，进行比较仔细的模块验证，测试的内容包括直接验证和随机验证。设计工程师提交给验证工程师的设计，一方面要通过代码质量检查，其内容包括有没有未初始化的变量、综合器不支持的数据类型或行为级描述、模块例化时不匹配的端口等；另一方面要通过测试覆盖率检查，保证行测试覆盖率达到或接近100%。对于IP宏模块的1级测试，测试的环境和测试用例由IP提供者给出，测试的结果可以作为IP选择的依据。

2级测试用于验证系统存储器的映射和模块间互连的正确性。测试内容包括通过片上处理器或片外处理器模型对设计中各个模块的寄存器进行读写操作，对于设计中的每个数据通路，通过外部接口写入激励数据，经过设计中各个模块间的接口传送，在设计的输出端口采集结果。

3级测试用于验证整个芯片的系统功能。测试内容包括长时间、大流量、随机性测试，错误条件测试、边界条件测试等等。自底向上的验证最后同样是网表验证、时序分析和物理验证。

自底向上的验证方法是当前业界最为流行的方法，其验证过程的收敛性好，验证效率高。但是，这一方法的难点在于需要在模块层次开发相应的测试环境和测试激励。这一问题的解决，将有赖于使用高级的验证语言如Vera和SpecmanE等。

2.4.3 基于平台的验证流程

基于平台的验证流程如图2.4.3所示。它是一种可以达到最大程度系统重用的面向集成的设计方法。不过它一般只适用于验证在已有的平台上开发出来的衍生类设计。其前提是已有的测试平台是已经验证过的，开发过程中继承的硬件IP和软件IP也是验证过的。验证的内容主要是基本的开发平台和新增加的IP模块之间的连接关系。设计平台至少包括一个处理器核以及保证处理器有效工作的必要外设模块。通过重用方法使用验证过的IP核、IP模块和设计平台，可以显著地减

小设计开发风险、提高产品性价比和快速上市的可能。

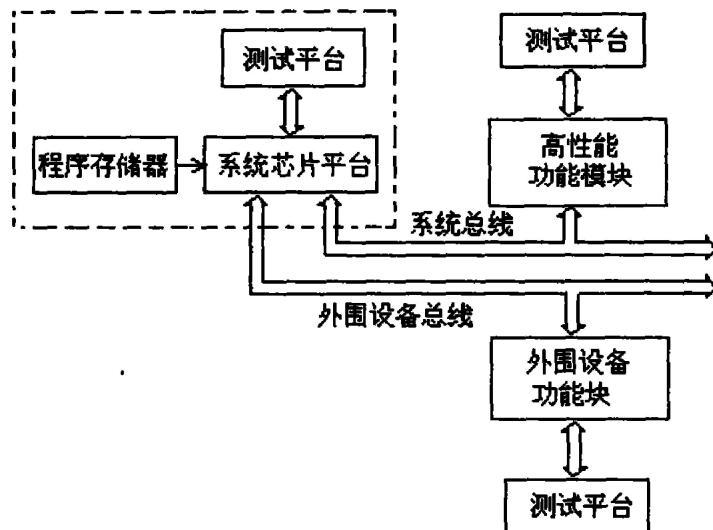


图2.4.3 基于平台的验证流程

2.4.4 基于系统接口驱动的验证流程

基于系统接口驱动的验证流程，是指在系统设计阶段为每个设计模块建立了相应的接口模型，这种接口模型可以帮助设计工程师对模块进行充分验证，验证工程师也可以很方便地进行模块间互连关系的验证和系统功能仿真。对于包含片上总线的设计，例如包含AMBA或PCI总线的设计，这种方法尤其适用。

如图2.4.4所示的示例中，系统包含了五个功能模块，在验证某一模块时，其它模块的接口模型可以被使用和替代，例如设计工程师在模块E验证时，可以利用模块A、B、C、D四个模块的接口模型，验证工程师在验证模块间连接关系时，可以采用接口模型和真实的设计混合建立测试环境以提高仿真速度。

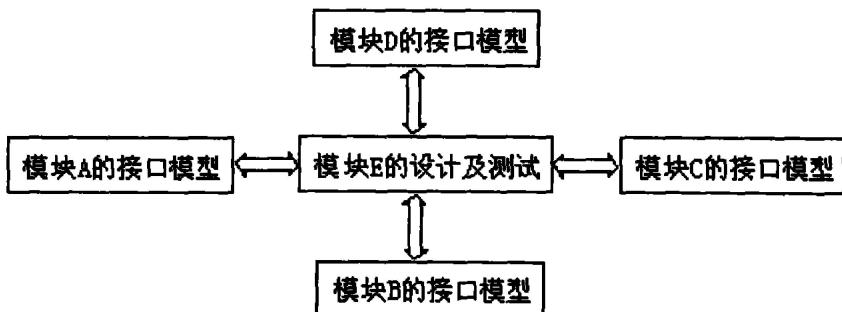


图2.4.4 基于系统接口驱动的验证流程示例

2.5 验证计划的建立

在对系统芯片的设计过程中，编制验证计划是非常重要的。它是一个可操作的文档，是接下来进行验证工作的指导书。在所有的验证要素中，验证计划决定了最终验证的好坏。所以，验证计划是非常关键的，而不是可有可无的。为了一次成功，设计者必须识别在什么条件下必须检测什么特征，以及期望的反馈应该是什么。整个设计团队都要参与计划制订，以保证它的完整性和正确性。

2.5.1 验证计划的制定

验证计划在制定时必须注意是“验证什么”而不是“怎样验证”，因为只有确定什么是需要验证的，验证团队才能保证计划的完整性和平衡性。这时，验证计划就不仅仅是怎样完成验证的工程规范了。验证计划主要包含以下内容：

- (1) 制订验证策略、验证项目、验证环境、验证手段、验证工具、验证进度、测试对象、测试方案、衡量标准、人力配备、完成期限、准备工作及所需资源。
- (2) 确定在各种条件下必须验证的项目、期望结果，确定项目的优先级及可做选择验证的项目。

2.5.2 验证的层次

当前的设计均采用层次化设计，相应地验证工作也要采用分层验证的方法。对每一层次的验证，必须保证设计的划分及接口的稳定性，否则验证工作将随着功能划分或接口变化而导致很多重复劳动，影响工程进度。验证层次一般划分如下：

- 功能单元的验证 (Unit-Level Verification)
- 可重用单元的验证 (Reusable Components Verification)
- ASIC和FPGA验证 (ASIC and FPGA Verification)
- 系统级验证 (System-level Verification)
- 板级验证 (Board-Level Verification)

各验证层次的特点和验证方法分别叙述如下：

(1) 功能单元的验证

设计单元（Design unit）的划分是一种逻辑划分，随着设计的深入，设计单元的功能和接口将发生较大的变化，因此设计单元的验证一般由设计者自行验证，验证的目的是保证设计单元的RTL代码无语法错误且能实现基本的功能，不用考虑代码的覆盖率及递归测试。

对于大型设计，每个设计单元都需要一个专门的验证环境，产生激励和检查响应要花去大量的时间，而且每个设计单元都写testbench，工作量是非常大的，所以进行正式的验证过程是不可能的，因此设计单元的验证一般采用一些特别的形式。但设计单元的集成性需在ASIC或FPGA-level进行验证。对于复杂的IC设计，可能存在复杂的设计单元，该设计单元的验证需具有较强的可见性与可控性，而且和该设计单元相关的功能尽可能都得到验证。

（2）可重用单元的验证

可重用单元是一种独立的设计部件，它和具体应用无关。可重用单元具有标准的外部接口，其testbench具有重用性。修改过的可重用单元应进行递归验证，以保证设计的后向兼容性，如果对设计的功能进行了修改，形式验证将不会起作用。设计可重用单元时，应将验证过程以文档的形式加以记录，获取用户对可重用单元的信任。

（3）ASIC和FPGA验证

ASIC和FPGA属于物理上的划分，它们的接口和功能在初步设计时就定义好了，不会有太大改动，这时可进行黑盒验证。对于复杂的ASIC芯片，可将ASIC验证做为系统验证。

（4）系统级验证

系统级是一种逻辑上的划分，由独立验证过的部件组成，系统级验证主要验证设计单元之间的相互关系，对于设计单元本身的功能在功能单元级或ASIC级已进行过验证。为了减少仿真的迭代，对定义的 testcase，尽量将 testcase 不需要的设计单元排除，使系统规模尽可能小。

（5）板级验证

利用板级设计工具生成的板级模型，设计的物理实现和板级仿真之间是一致的，与系统级逻辑模型不同。板级模型所包含的组件模型可来自第三方也可用

硬件模拟器代替，验证时应对板级物理参数进行模拟，保证功能的正确性。板级连通性的验证采用形式验证，将对组件管脚连接的描述和板级设计工具生成的网表进行比较。

验证时首先要定义验证的等级，是系统级的还是功能单元级的验证，同时还需确定各级的testcase，根据对设计实现的把握，来决定采用黑盒还是白盒的 testcase，即确定 testcase 的抽象级别，然后确定激励的输出结果及输出结果的检查办法。

引　　言

随着集成电路的发展，SoC（片上系统），设计已经成为主流，并且复杂度不断增加，如何提高设计成功率，缩短产品上市时间变成产品成败的关键因素。SOC的最大的特点是它具有一定的系统特性，除了大量硬件模块之外，还需要固件和软件的支持。所以传统的基于软件的验证方式已经无法胜任。由于软硬协同验证时，软件验证的耗时令人难以忍受。为了缩短SOC验证时间，原型验证（Rapid System Prototype）已经成为SOC设计流程中主要手段。

原型验证的本质是通过FPGA快速实现SOC设计中的硬件模块，让软件能在真正的硬件上全速运行，实现SOC设计中软硬件协同验证的目的。由于原型验证具有复杂的系统性，所以在验证过程中会遇到各方面的问题。

论文首先简单介绍了采用基于FPGA的原型验证的作用和需求，提供了一个如何构建一个成功的基于FPGA的原型验证的方案。

第一章介绍原型验证的目的。

第二章介绍了FPGA的器件结构，从中总结了FPGA与SOC在结构和设计的不同之处，为下几章内容提供技术铺垫。

第三章和第四章介绍了原型验证的流程和验证策略，

第五章介绍了如何设计一个可靠的原型验证硬件系统，其中还简单介绍了信号完整性相关内容。

第六章和第七章主要介绍了如何将SOC的电路通过FPGA来实现，其中需要做代码风格的转换，时序收敛和优化等工作。

第八章介绍了在原型验证中IP模块的选型和整合。

第九章规范总结了前端设计要求，以方便原型验证的顺利实现。

第十章以一个实际的NFC原型验证项目范例来证明本文所提出方法的可行性。

第一章 原型验证的目的

第一节 原型验证能做些什么

原型验证的目的	原型验证的原因
◆ 使软件开发启动时间提前 ◆ 使软硬件系统集成的启动时间提前 ◆ 实时的大数据流测试	只有 FPGA 能够提供与 ASIC 类似性能和一致性。可以完全模拟真实 SOC 应用
◆ 真实环境测试 ◆ 真实人机接口测试	某些电路的功能与外部环境和系统上的多种因素有关，使之无法得到精确的仿真模型，故无法通过软件仿真的方法全面验证
◆ 可行性测试 ◆ 算法验证	实验阶段
◆ 初期市场运作 ◆ 拉投资	出样片和样机

第二节 原型验证不能做些什么

原型验证的激励源来自物理上的真实激励源，所以原型验证无法像软件仿真那样方便的实现各种激励和测试结果自判定。

其次虽然 FPGA 跑测试向量确实比仿真器快几个数量级，但 FPGA 电路的实现需要更长的时间。一般从 code 更新，逻辑综合，物理 P&R，时序收敛，需要数小时。如果 RTL 有改动，需要重新执行一遍实现流程，非常耗时。

最后 FPGA 不能提供良好的可观性，如果要看内部信号，要使用 identify，能看的波形数量和深度取决于 FPGA 的容量。需要注意的是用 identify 看的波形其采样精度受限于 sample clk 采样频率，同时在 RTL 中插入 identify 的测试逻辑可能还会影响 RTL 的时序。

因此原型验证无法完全替代软件验证。

一款多核处理器 FPGA 验证平台的设计与实现

朱英 陈诚 许晓红 李彦哲

(上海高性能集成电路设计中心 上海 201204)

(zhuying_1116@sina.com)

Design and Implementation of FPGA Verification Platform for Multi-core Processor

Zhu Ying, Chen Cheng, Xu Xiaohong, and Li Yanzhe

(Shanghai High Performance IC Design Center, Shanghai 201204)

Abstract As the design of high performance microprocessor becomes more and more complicated, the hardware-software co-verification, which is based on the FPGA (field programmable gate-array) prototyping verification platform, is usually used before tape-out. The use of FPGA platform is aimed to shorten the period of verification and reduce the risk of manufacture. With the developing of multi-core microprocessor, the implement of FPGA prototyping verification platform becomes more and more complex. Firstly, the paper introduces how to design and implement the FPGA prototyping verification platform for a high-performance multi-core microprocessor. Secondly, the paper describes the details about the construction of the FPGA platform, including strategy of FPGA partitioning, time division multiplexing communication, and implement of I/O interfaces. The FPGA platform is constructed by several mother boards and daughter boards, so it is adaptable for different periods of chip verification by changing the scale. This FPGA verification platform is scalable and flexible, and contributes a lot to the function correctness verification and performance evaluation of the target design. As a result, the success of the FPGA verification efforts underscored by first prototype chips which can boot operating system and test lots of application programs successfully. In the end, the paper also analyses the application prospect of this FPGA prototyping verification platform.

Key words FPGA prototyping verification; FPGA partitioning; time division multiplexing transport; delay adjust; performance evaluation

摘要 高性能处理器设计日趋复杂,为了缩短验证周期,降低研制风险通常需要在流片之前进行基于现场可编程门阵列(field programmable gate-array, FPGA)原型验证平台的软硬件协同验证。随着处理器多核化的发展,FPGA 原型验证平台的实现变得越来越具有挑战性。介绍了一款高性能多核微处理器 FPGA 验证平台的设计与实现方法,详细阐述了该 FPGA 验证平台采用的母板/子板总体架构、分片策略、时分复用实现技术及 I/O 接口实现方法。该平台具有良好的可扩展性,能够方便灵活地实现目标芯片在各种规模和配置下的 FPGA 验证,用于在流片前对目标芯片进行功能正确性验证和性能评估。经过该 FPGA 平台验证的目标芯片,首次流片返回的芯片能成功运行操作系统和各种应用程序,实现了一次流片成功的目。最后对该 FPGA 验证平台的应用前景进行了分析总结。

关键词 FPGA 原型验证;FPGA 分片;时分复用传输;延迟调节;性能评测

中图法分类号 TP302

收稿日期:2013-02-04;修回日期:2013-07-01

基金项目:“核高基”国家科技重大专项基金项目(2009ZX01028-002-001)

通信作者:陈诚(hmioycc@gmail.com)

随着集成电路工艺的不断发展和处理器体系结构的不断创新,高性能处理器的复杂度和规模不断提升,正确性验证已经成为高性能处理器芯片研制过程中的重要瓶颈之一。验证过程通常占整个芯片开发周期的 70%以上^[1],验证方法的选择成为芯片研制成功与否的关键因素之一^[2]。

通常在高性能处理器芯片的验证过程中,软件模拟验证、硬件加速器仿真验证和 FPGA (field programmable gate-array) 原型验证是最常见的 3 种方法^[3-5]。

软件模拟验证是当今芯片正确性验证的一种最常用的验证手段,具有模拟精准、使用灵活方便的优点。但随着芯片规模的提升,全片级的模拟验证速度急剧下降,全片级的验证速度往往只能达到每秒几百拍,无法进行大规模的操作系统和应用程序的功能验证和性能评估,验证的效率受到了极大制约。

基于硬件加速器的仿真验证,采用了特殊的验证加速技术,运行速度可以达到兆级,同时具有良好可观性和可控性,在一定程度上缓解了软件模拟验证运行速度慢的缺点。但是硬件加速器价格昂贵,而且运行速度仍然不够快,仍不能满足大规模系统及应用程序验证需要。如龙芯 2 号微处理器的验证过程中就使用了 Mentor 公司的 VStation Pro 产品,仿真速度达到 800 KHz,实际运行速度只是软件模拟速度的 10~50 倍^[6-7]。

FPGA 原型验证通过使用可编程器件实现设计逻辑,进行物理原型仿真,最大的优点就是运行速度快,适合流片前的系统级验证和性能评估。目前业界已推出通用 FPGA 原型验证平台,如 Synopsys 公司的高性能 ASIC 原型机系统 (high-performance ASIC system) 系列产品和 S2C 公司的 FPGA 原型验证平台。这种商业通用 FPGA 原型验证平台通常采用母板加子板的结构,母板上使用 1 片或多片大容量 FPGA 器件,子板上提供常用 I/O 接口,通过标准接口可以将多块母板和子板互连使用^[8-9]。采用这种通用灵活结构方式的 FPGA 原型验证平台可以在多个不同项目中重复使用,但这种商用平台往往价格昂贵,实现一款高性能多核处理器芯片的 FPGA 原型验证需要很高的代价。

FPGA 原型验证平台还可以根据研制项目需求自行设计。这种 FPGA 原型验证平台通常采用 1~2 块电路板级联使用,将所有器件都放在这 1~2 块电路板上,从而可以减少由于分片、级联使用带来的性能损耗。比如 Intel Atom 处理器在投片前开发的

FPGA 原型验证平台,在 1 块电路板上使用 1 块 Virtex5 LX330 FPGA 来实现整个 Atom 处理器核心逻辑,运行速度为 20~50 MHz^[10]。在 Intel Nehalem 处理器的研制过程中由于芯片规模增大,原来的 FPGA 原型验证平台不能适用,在投片前重新开发了新的 FPGA 原型验证平台,在 1 个电路板上使用 5 块 FPGA 来实现核心逻辑,运行速度为 520 KHz^[11]。这种固定结构方式的 FPGA 原型验证平台通常对当前项目适用,但是很难用于其他项目。

本文要验证的目标芯片是一款高性能多核处理器。为了缩短验证周期,降低研制风险,需要在流片前在 FPGA 原型验证平台上使用操作系统、编译器及应用程序来验证目标芯片的功能正确性,并进行性能评测。为了降低成本,同时考虑系列产品研发的需要,在借鉴国内外相关领域研究成果的基础上,本文采用了自行研制的具有一定通用性的 FPGA 原型验证平台策略,并采用母板加子板的总体结构、合理的分片策略、可靠的时分复用实现技术及完整的 I/O 接口实现,使得设计实现的 FPGA 平台具有可扩展性好、综合成本低、可靠性高、可重用性好和验证覆盖率高等优点,而且性能评估精度较高,流片返回芯片的 Linpack 测试效率与 FPGA 平台上的测试结果相比,误差只有 3.33%。经过本文创建的 FPGA 平台验证的目标芯片,首次流片返回的芯片能成功运行操作系统和各种应用程序,并达到了预期的性能目标,实现了一次流片成功的目标。

1 目标处理器简介

1.1 目标处理器结构

本文设计实现的 FPGA 原型验证平台所验证的目标芯片是一款高性能多核处理器。该处理器为 64 位 Load/Store 型 RISC 结构,采用可伸缩多核结构和片上系统技术,芯片顶层由多个对称核组(最多可集成 4 个核组)、片上网络和系统接口组成,其中系统接口包含传输速率为 8×5 Gbps 的 PCI-E2.0 接口、千兆以太网接口及低速的维护调试接口。每个核组包含多个对称 64 位通用核心(最多可集成 4 个核心)、核组互连、片上网络接口和 1 路存储控制器。每个核心由指令部件、整数/浮点运算部件、Cache 管理部件、Cache 阵列及核组互连接口几部分组成。目标芯片核心运行频率为 1.1 GHz,单核双精度浮点峰值运算速度为 8.8Gflops@1.1 GHz,全芯片 16 个核心双精度浮点峰值运算速度为 140.8Gflops@1.1 GHz。

1.2 FPGA 实现面临的挑战

目标芯片结构复杂,规模庞大,高速 I/O 接口种类多,基于 FPGA 实现目标芯片的原型验证平台将面临如下挑战:

- 1) 目标芯片结构复杂,如何确定合理的验证平台总体架构,以适应目标芯片的对称可伸缩多核结构,满足芯片在单/多核、单/多核组及全片等不同规模下的验证需要,具有一定难度;
- 2) 目标芯片规模庞大,验证平台只能采用多片 FPGA 来实现. 如何设计一个能够综合考虑验证平台可伸缩性、成本、系统规模、工程实现难度等多方面因素的分片方案,是一个挑战;
- 3) 验证平台需要采用多片 FPGA 实现,如何利用有限的 FPGA 芯片引脚资源满足数量巨大的片间互连需求,是一个亟待解决的关键问题;
- 4) 目标芯片集成了 DDR3,PCI-E2.0 等多种基于 IP 实现的高速接口,为了验证这些 I/O 接口,需要解决这些 IP 所包含硬核的 FPGA 实现问题,同时为了保证平台进行性能评测的精度,还要解决 FPGA 实现导致的核心与 I/O 接口频率比与实际芯片设计不匹配问题.

本文后续将分别介绍如何解决上述挑战,从而建立起目标芯片的 FPGA 原型验证平台.

2 FPGA 原型验证平台的总体架构

2.1 总体原则

本文开发的验证平台采用母板加子板架构. 总体结构上采用可扩展插件结构,将系列产品验证中具有通用性的插件作为母板,将特定芯片验证需要的插件作为子板. 验证特定芯片时,通过标准接口将多个母板和子板连接起来构成一个完整的原型验证平台. 基于上述的总体原则以及对验证目标芯片的 FPGA 实现进行评估的结果,本文的目标平台设计了 2 种母板、3 种子板,下面作具体介绍.

2.2 母板结构

目标平台设计了 2 种母板,分别是单芯片板和双芯片板,具体结构如下:

- 1) 单芯片板. 逻辑示意图如图 1 所示,主要包含 1 块 Xilinx 公司的 XC5VLX330 芯片,用于实现设计逻辑. 芯片的 I/O 引脚数为 1200 根^[12-13],引出到 14 个标准插座上,每个插座的有效信号数为 80 个,插座上的 I/O 电源大小可以在一定范围内调整^[14]. 适合用于实现逻辑规模大、分片困难、引脚数

较多的功能模块,在目标平台中用于实现系统接口、核组内除核心之外的逻辑.

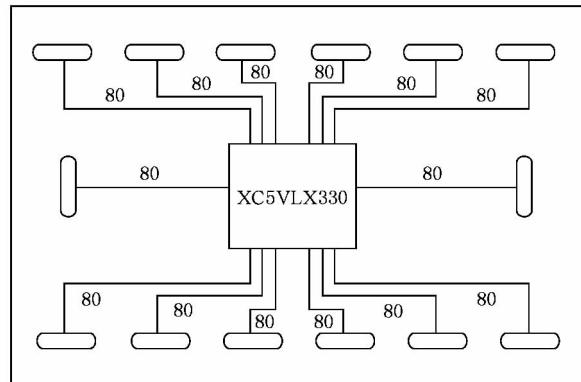


Fig. 1 Single chip board logic.

图 1 单芯片板逻辑示意图

- 2) 双芯片板. 逻辑示意图如图 2 所示,主要包含 1 块 Xilinx 公司的 XC5VLX330 芯片和 1 块 Xilinx 公司的 XC5VLX220 芯片,用于实现设计逻辑. 2 个 FPGA 芯片之间板上直接连线数为 260 根,每个 FPGA 芯片的引脚各自引出到 6 个插座,2 个 FPGA 芯片共 12 个插座,每个插座有效信号数为 80 根,插座上的 I/O 电源大小也可在一定范围内调整^[14]. 适合用于实现逻辑规模大、容易分片、引脚数较多的功能模块,在目标平台中用于实现每个核心的逻辑.

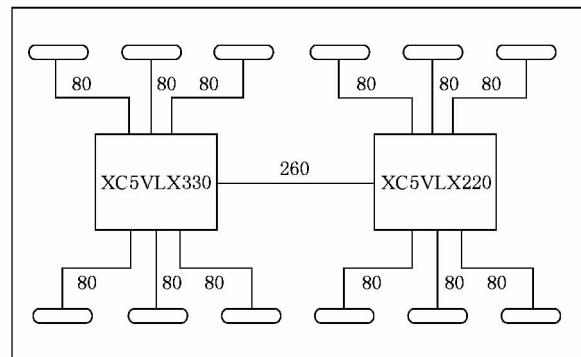


Fig. 2 Dual chip board logic.

图 2 双芯片板逻辑示意图

2.3 子板结构

目标平台设计了 3 种子板,分别是存储板、PCI-E 接口板和时钟板,具体功能如下:

- 1) 存储板. 主要包含 2 个 DDR3 存储器条的插槽,用于 DDR3 存储器接口验证.
- 2) PCI-E 接口板. 主要包含 1 块 Xilinx 公司的 XC5VLX220T 芯片和 PCI-E 插槽,用于 PCI-E 接口验证.

3) 时钟板. 主要用于提供整个 FPGA 原型验证系统的时钟.

2.4 总体架构

基于母板、子板的目标平台总体架构如图 3 所示, 其中 1 个核组的结构如图 4 所示. 这种架构能方

便地实现以核心、核组为单位的验证规模伸缩, 并且构成平台的母板和子板在后续 3 个不同规模、不同架构的系列产品验证中得到继续应用, 避免出现 Intel Atom 处理器和 Nehalem 处理器研制过程中 2 个项目的 FPGA 原型验证平台不能重用的问题^[10-11].

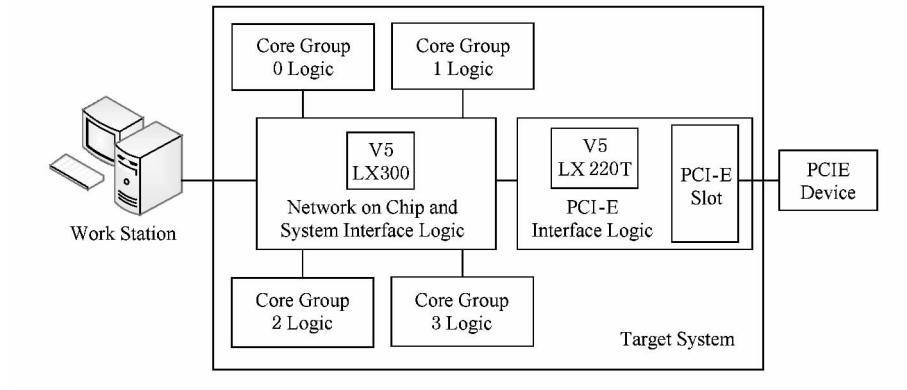


Fig. 3 The architecture of FPGA target platform.

图 3 FPGA 目标平台总体架构图

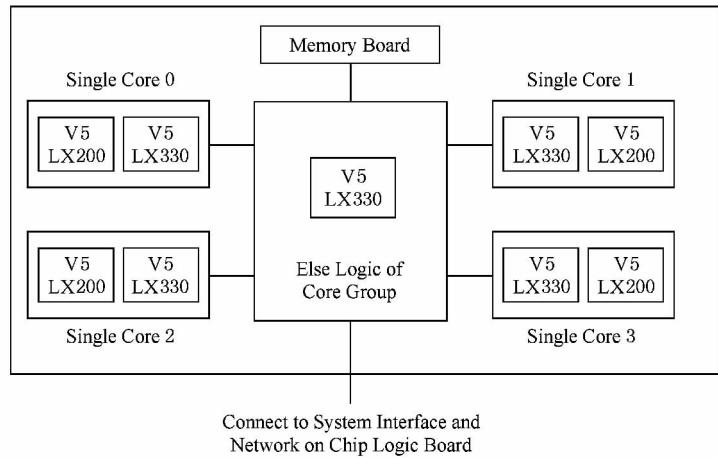


Fig. 4 The architecture of single core group FPGA target platform.

图 4 FPGA 目标平台单核组架构图

3 分片原则及方案

3.1 分片原则

面对结构功能复杂、规模庞大的目标芯片的 FPGA 实现, 如果使用传统的综合工具自动分片的方法需要使用 40 多片 FPGA 实现, 每个 FPGA 中映射的逻辑都是独立的, 需要完成 40 多片 FPGA 的综合才能实现所有设计逻辑, 代价非常大.

为了解决上述问题, 本文根据验证目标芯片的结构特征, FPGA 实现的划分遵循了如下分片原则:

1) 充分利用芯片以核心和核组为单位的对称结构特点, 以核心和核组为功能模块来进行逻辑划

分, 以尽量减少综合芯片数量, 降低综合成本;

2) 采用自底向上、层次化的分片原则, 按照核心、核组、全片的顺序进行分片;

3) 尽量将 FPGA 分片的边界放在片间互连信号较少的区域, 以缓解片间互连引脚紧张的问题.

3.2 分片方案

为了对目标芯片进行合理的 FPGA 实现划分, 我们使用综合工具 Synplify_pro 对芯片各功能模块的资源占用量进行了综合评估, 同时综合考虑相关功能模块之间的连接信号数量. 核心、核组、全片 3 个层次的评估结果和分片方案具体说明如下:

1) 核心分片方案

单个核心主要由取指令部件、整数/浮点运算部

件、数据 Cache 管理部件、二级 Cache 管理部件及 Cache 阵列组成。其中浮点运算部件逻辑规模最大,功能上也比较独立,和其他功能模块之间的连线也相对较少,因此考虑将浮点运算部件和其他功能部件分开,单独使用 1 片 FPGA 实现,综合评估结果如表 1 所示:

Table 1 Single Core Logic Synthesize Evaluation Result**表 1 单核心逻辑综合评估结果 %**

Device	Implement Logic	Resource Type	Utilization
XC5VLX330	Single Core	Registers	62
		LUTs	85
		Slice	99
		BlockRAM	74
		IOB	41
XC5VLX220	Floating Point Unit	Registers	27
		LUTs	78
		Slice	91
		BlockRAM	0
		IOB	44

从评估结果可见,2 个 FPGA 芯片 LUTs 资源的利用率都在 80% 左右,分片后的逻辑分布基本平衡,因此单核心逻辑使用双芯片母板来实现,其中浮点运算部件使用 1 块 XC5VLX220 芯片实现,其余功能模块使用 1 块 XC5VLX330 芯片实现。

2) 核组分片方案

单核组由最多 4 个核心、核组互连、片上网络接口、存储控制器几部分组成。4 个单核心按照核心分片方法实现,核组内其他逻辑的综合评估结果如表 2 所示:

Table 2 Other Logic of Core Group Synthesize Evaluation Results**表 2 核组内其他逻辑综合评估结果 %**

Device	Implement Logic	Resource Type	Utilization
XC5VLX330	Single Core Group	Registers	76
		LUTs	67
		Slice	98
		BlockRAM	17
		IOB	90

从评估结果可见,核组内除核心外的其他逻辑适合用 1 块单芯片母板来实现,由于存储控制器需要和 DDR3 存储器相连,因此该单芯片板还需要和 1 块存储器子板相连。

3) 全片分片方案

全芯片顶层由最多 4 个核组、片上网络、系统接口几部分组成。4 个核组的分片方法按照核组分片方案实现。系统接口中包含的 PCI-E 接口处理部件需要使用高速接口单元来实现,只有特定型号的 FPGA 芯片才支持^[15]。芯片顶层的片上网络和系统接口的综合评估结果如表 3 所示:

Table 3 Network on Chip and System Interface Synthesize Evaluation Results**表 3 片上网络和系统接口综合评估结果 %**

Device	Implement Logic	Resource Type	Utilization
XC5VLX330	Network on Chip and System Interface	Registers	38
		LUTs	37
		Slice	65
		BlockRAM	0
		IOB	80
XC5VLX220T	PCI-E Interface Unit	Registers	19
		LUTs	25
		Slice	42
		BlockRAM	11
		IOB	33

从评估结果可见,芯片顶层的片上网络和系统接口的其他逻辑可以采用 1 块单芯片母板来实现,采用单芯片板实现这部分逻辑,FPGA 资源利用率较低,但这能为后续系列芯片扩展系统接口的功能留下一定资源余量。该顶层母板同时也实现了目标芯片的维护/控制逻辑,并通过标准接口和平台的前端控制台连接。PCI-E 接口处理部件使用 PCI-E 接口子板上的 XC5VLX220T 芯片实现。

基于上述层次化的分片方案,最终用 16 块双芯片母板、5 块单芯片母板、4 块存储器子板、1 块 PCI-E 接口子板实现了整个目标芯片的逻辑。全片实现包含 21 片 XC5VLX330、16 片 XC5VLX220 和 1 片 XC5VLX220T,合计 38 个 FPGA 芯片。但最终只需要综合 5 片 FPGA 逻辑,分别是用于实现核心逻辑的 2 片 FPGA、核组内除核心外逻辑的 1 片 FPGA、片上网络和系统接口的 1 片 FPGA,以及 PCI-E 接口的 1 片 FPGA。上述这种层次化的分片方案显著降低了综合成本和系统实现难度。

4 片间时分复用传输技术的实现

目标芯片采用 38 个 FPGA 芯片来实现整片逻辑,由于 FPGA 芯片引脚数量的限制,实际 FPGA

芯片之间的连线数量远远少于映射逻辑之间的互连信号数量,必须要在1根连线引脚上传输多个信号,本文采用时分复用传输技术解决这个问题.

4.1 时分复用传输实现方法的选择

通常时分复用传输有2种实现方法:1)使用简单的多路复用器实现;2)使用基于低电压差分信号(low voltage differential signal, LVDS)的源同步传输方式^[16].

使用多路复用器的实现方法,实现电路简单,对FPGA电路板没有特殊需求,但是受到传输延迟的限制,通常使用这种实现方法接口传输时钟只能达到100 MHz^[16].

基于LVDS的源同步传输方式,通过将数据和时钟一起发送来提高传输速度,通常接口传输时钟可以达到800 MHz.这种实现方法的实现电路复杂,要求FPGA电路板支持高速串口信号传输,每个信号需要使用1对差分信号传输.通常时分复用比例达到或超过16:1,即1根物理连线上时分复用传输超过16根信号,使用这种实现方法比较合适.

根据目标芯片各个功能模块之间的互连情况,本文的目标平台FPGA片间时分复用比例通常在6:1以下,FPGA内部逻辑工作在5 MHz左右,因此不需要采用过高频率的接口传输时钟.如果采用基于LVDS的源同步传输方式,不仅实现电路复杂,而且功耗较大,由于平台散热条件的限制,不利于整个平台的长时间稳定运行.综合考虑到实现复杂度

和功耗等因素,本文的目标平台FPGA片间时分复用采用多路复用器实现.

4.2 时分复用发送及接收窗口实时调节功能

本文的目标平台FPGA片间均采用基于多路复用器的时分复用传输技术,部分片间接口使用4倍速传输,部分引脚资源更加紧张的接口使用了6倍速传输.为了增加传输可靠性和调试灵活性,本文目标平台上实现的时分复用传输技术在常用方法的基础上增加了接收窗口内多个接收边沿实时可调节的功能.

下面以本文实现的6倍速传输为例,具体说明接收窗口内多个接收边沿实时可调节功能的实现方法.如图5所示,芯片工作时钟周期和传输时钟周期的比例为16:1,发送方设置8个发送窗口,考虑到数据传输的稳定性,舍弃最不稳定的第1个和最后一个窗口,选择中间6个窗口发送.接收方可选取2个传输时钟上升沿与2个下降沿作为接收信号的触发条件,即在运行过程中每个接收窗口有4个信号接收时刻可实时选择.目标FPGA平台实现中,可以在平台每次运行之前先通过训练,为各个FPGA片间互连接口寻找到最稳定可靠的接收边沿,正常运行时通过配置内部寄存器选择最稳定可靠的接收边沿.这种1个窗口具有多个接收边沿的时分复用传输方式可以适应FPGA芯片间传输信号的各种延时情况.基于这种时分复用实现方法,目标FPGA平台片间信号传输时钟频率达到80 MHz,内部核心逻辑稳定工作频率达到5 MHz.

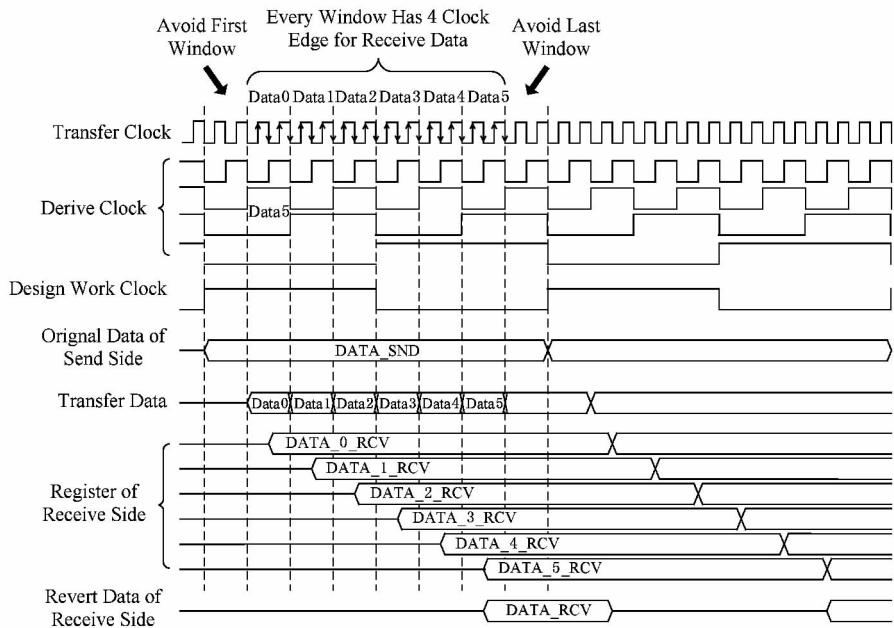


Fig. 5 Time-division multiplexing transmission.

图5 时分复用传输

为了确保时分复用传输信号的稳定可靠性,同时尽可能提高平台的工作频率,在具体实现时还遵循了如下原则:

- 1) 必须保证输入给多个 FPGA 芯片的时钟和复位信号是同步的;
- 2) 发送方通常选择中间窗口发送传输信号,舍弃第 1 个和最后 1 个窗口,避免发生时序问题;
- 3) 发送方延迟较长的接口信号放在靠后的窗口发送,以避免出现建立时间不够的时序问题,延迟较小的接口信号放在靠前的窗口发送,避免由于不同芯片的时钟偏斜导致出现保持时间不够的时序问题;
- 4) 尽量选择低频时钟域的信号进行多路复用传输,避免对高频时钟域的信号进行多路复用传输,以提高系统工作频率;
- 5) 在引脚资源允许的情况下,通常对数据信号采用时分复用的方法进行传输,对控制信号由于时序的复杂性尽量采用直接传输的方法.

5 I/O 接口的 FPGA 实现

本文验证的目标芯片内集成了 4 路 3 代内存(double data rate 3)存储控制器、PCI-E2.0 与千兆以太网 I/O 接口。这些接口的设计实现集成了第三方开发的 IP,PCI-E2.0 与 DDR3 存储控制器接口,还集成了硬核物理层(physical layer, PHY)。为了验证这些 I/O 接口的功能正确性,需要在 FPGA 上实现这些 I/O 接口。

I/O 接口集成的第 3 方硬核 PHY 不能直接在 FPGA 上实现。为了在 FPGA 上进行这些 I/O 接口的验证,需要对 I/O 接口集成的硬核 PHY 进行设计修改。DDR3 存储控制器接口的 PHY 在 FPGA 上实现时,自行设计了专用于 FPGA 实现的 DDR3 PHY,简化了 DDR3 PHY 的部分功能。PCI-E2.0 接口在 FPGA 上的实现也进行了一定设计修改。为了降低 FPGA 上的实现难度,基于 PCI-E 接口板上的 XC5VLX220T 芯片支持的 ROCKET I/O, FPGA 上只验证了 PCI-E Gen1.0 模式,但这种实现模式已能满足对自行设计逻辑的功能验证需要。

在 FPGA 实现时,由于 FPGA 分片、片间时分复用传输及资源限制等影响,芯片内部逻辑的工作频率往往不会很高,而 DDR3 和 PCI-E 这些标准接口又有最低工作频率限制,这些因素造成芯片的 FPGA 实现版本的各种时钟频率关系和实际目标芯

片不同,特别是 I/O 接口与内部逻辑之间的频率关系相差较大,这是 FPGA 原型验证中难以避免的。本文实现的 FPGA 平台也不可避免地出现内部逻辑频率与 I/O 接口频率之比远小于实际芯片的情况。为了降低这种频率关系的变化导致性能评估的误差,我们让 DDR3 存储器接口尽量工作在低频模式下,同时在访存通路上增加固定级数的触发器来达到延迟调节的功能,用于校准访存延时误差,通过这些方法来保证 FPGA 平台进行性能评测的精度。

6 FPGA 原型验证平台的应用

本文设计实现的 FPGA 原型验证平台具有良好的可扩展性,灵活方便地实现了单/多核心、单/多核组及全片等各种规模的验证平台,以适应验证目标芯片不同研制阶段的各种验证需求,并且各种规模的验证平台均被复制多套,加快了芯片的验证周期。平台的运行速度达到 5 MHzps,相当于模拟速度的数万倍,相当于硬件仿真加速器上仿真速度的 10 倍。设计实现的 FPGA 原型验证平台主要用于验证目标芯片的功能正确性和进行性能评估。

在实际目标芯片验证中,本文设计实现的 FPGA 平台上建立了完善的软件环境,在 FPGA 原型验证平台上可以进行操作系统、编译器及应用程序的验证。芯片流片之前在设计实现的各种规模平台上完成了几千道测试课题的验证,包括各类串行课题和多核并行课题。验证过程中发现了多个软硬件接口协议及芯片逻辑设计问题,为保证芯片的功能正确性发挥了重要作用。

基于设计实现的 FPGA 原型验证平台可以对目标芯片进行各种性能测试,包括芯片的 Linpack 效率、访存性能等。芯片在不同微结构实现方案下的性能测试结果可以为设计人员确定一个性价比较高的微结构实现方案提供很好的支持。芯片流片之前在该平台上完成了各种性能测试,以确保流片返回芯片达到预期的性能目标,其中 Linpack 程序测试结果如表 4 所示。

从测试结果可见,流片返回芯片的全片 Linpack 效率与 FPGA 平台上的测试结果相比,误差只有 3.33%。

除了 Linpack 程序之外,在使用其他典型性能评测程序测试的过程中,发现有少数程序的性能测试结果与实际系统平台上的测试结果误差超过 15%,出现这些较大误差的主要原因是 DDR3 接口

与内部逻辑之间的频率关系和实际设计相差较大, 虽然为了校准访存延时误差, 在访存通路上增加了延迟调节功能, 但是这种固定延迟调节的方法不能满足所有应用程序的访存延时误差的校准。为了解

决以上性能测试误差问题, 需要对误差较大的程序进行深入分析, 掌握这些程序的特点, 有针对性地在原有固定延迟调节的基础上增加动态延迟调节功能, 进一步提高性能评测程序的测试结果精度。

Table 4 Linpack Test Results

表 4 Linpack 测试结果

Test Platform	Core Frequency/MHz	Floating Point Peak/Mflops	Efficiency/%	Equivalent Error/%
FPGA	5	476.727	74.49	3.33
Real System	1100	101500	72.09	3.33

经过本文设计实现的 FPGA 平台验证的目标芯片, 首次流片返回的芯片能成功运行操作系统和各种应用程序, 并达到了预期的性能目标, 实现了一次流片成功的目标。

7 总 结

在高性能微处理器的研制过程中, 为了缩短研制周期、降低研制风险, 通常需要在流片之前进行基于操作系统、编译器、应用程序等大规模测试程序的验证, 为了实现这个目标, 自主研制具有一定通用性的 FPGA 原型验证平台是一个比较经济实用的选择。

本文开发的一款高性能多核微处理器 FPGA 验证平台, 和其他同类型的平台相比有如下优点:

1) 充分利用目标芯片的对称结构特点, 采用合理的层次化分片策略, 使得由 38 个 FPGA 芯片构成的整个 FPGA 验证平台, 只需要综合 5 片 FPGA 就能实现所有设计逻辑, 大大降低了综合成本和系统实现难度;

2) 本文设计实现的 FPGA 验证平台, 采用基于窗口实时调节的时分复用传输技术, 在基于常用的多路复用器实现方法基础上, 实现了接收窗口内多个接收边沿实时可调节功能, 这种方法不但实现电路简单、功耗小, 而且信号传输的可靠性和平台调试的灵活性得到了有效提高;

3) 本文设计实现的 FPGA 验证平台实现了完整的 I/O 接口, 并采用延迟调节等校准方法, 保证了性能评测的高精度。基于流片返回芯片测试的全片 Linpack 效率与 FPGA 平台上的测试结果相比, 误差只有 3.33%。

本文设计实现的 FPGA 平台可以继续用于下一代同系列处理器的验证, 平台设计实现过程中采用的一些策略和关键技术, 也可以被推广应用到其

他系列处理器芯片, 甚至其他领域芯片的 FPGA 原型验证平台设计实现。随着处理器结构日趋复杂, 其 FPGA 原型验证平台的设计实现也将面临新的挑战, 我们将进一步研究如何根据 FPGA 芯片技术的发展, 更加合理地创建结构更加复杂的高性能处理器 FPGA 原型验证平台, 并使得创建的平台具有更高的性能评测精度和功能验证覆盖率。

致谢 在此, 我们向对本文的工作给予支持和建议的同事表示感谢!

参 考 文 献

- [1] David D, Michael S. Verification Methodology Manual Techniques for Verifying HDL Designs [M]. Winchester: Teamwork International, 2002: 1-5
- [2] Doug A. Understanding the real cost of prototyping hardware [R]. Mountain View: Synopsys Inc, 2010: 1-13
- [3] Taylor S, Quinn M, Brown D, et al. Functional verification of a multiple-issue, out-of-order, superscalar Alpha processor—the DEC Alpha 21264 microprocessor [C] //Proc of the 35th Design Automation Conf. New York: ACM, 1998: 638-643
- [4] Lutte J, Roesner W, Heiling G, et al. Functional verification of the POWER4 microprocessor and POWER4 multiprocessor systems [J]. IBM Journal of Research and Development, 2002, 46(1): 53-73
- [5] Wazlowski M, Adiga N, Beece D, et al. Verification strategy for the Blue Gene/L chip [J]. IBM Journal of Research and Development, 2005, 49(2/3): 303-318
- [6] Zhang Heng, Shen Haihua. Function verification of Godson2 processor [J]. Journal of Computer Research and Development, 2006, 43(6): 974-979 (in Chinese)
(张珩, 沈海华. 龙芯 2 号微处理器的功能验证[J]. 计算机研究与发展, 2006, 43(6): 974-979)
- [7] Zhang Heng. The application of Mentor verification solution proposal on the Godson processor design [C] //Proc of Mentor User Conf. Wilsonville: Mentor Inc, 2010: 3-5 (in Chinese)

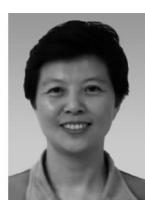
- (张珩. Mentor 验证解决方案在龙芯处理器设计中的应用 [C] //Mentor 用户大会论文集. 威尔逊维尔: Mentor 公司, 2010: 3-5)
- [8] Synopsys Inc. HAPS-52 Virtex-5 Motherboard Datasheet [M]. Mountain View: Synopsys Inc, 2010: 1-3
- [9] S2C Inc. Design SoC Using FPGA-based IP [M]. San Jose: S2C Inc, 2005: 3-10
- [10] Perry H, Jamison D, Chris T, et al. Intel Atom processor core made FPGA-synthesizable [C] //Proc of the 17th Int Symp on Field Programmable Gabe Array. New York: ACM, 2009: 209-218
- [11] Graham S, Jamison C, Ethan S, et al. Intel Nehalem processor core made FPGA synthesizable [C] //Proc of the 18th Int Symp on Field Programmable Gabe Array. New York: ACM, 2010: 3-12
- [12] Xilinx Inc. Virtex-5 Packaging and Pinout Specification UG190 V3. 1 [M]. San Jose: Xilinx Inc, 2007
- [13] Xilinx Inc. Virtex-5 User Guide UG190 V3. 1 [M]. San Jose: Xilinx Inc, 2007
- [14] Xilinx Inc. Virtex-5 Data Sheet DC and Switching Characteristics DS202 V3. 6 [M]. San Jose: Xilinx Inc, 2007
- [15] Xilinx Inc. Virtex-5 RocketI/O GTP Transceiver User Guide UG196 V1. 4 [M]. San Jose: Xilinx Inc, 2007
- [16] Doug A, Austin L, Rene R. FPGA-Based Prototyping Methodology Manual [M]. Mountain View: Synopsys Inc, 2011: 153-156



Zhu Ying, born in 1964. Professor in Shanghai High Performance IC Design Center. Her main research interests include high performance computer Architecture, processor design and verification.



Chen Cheng, born in 1980. Assistant professor in Shanghai High Performance IC Design Center. His main research interests include processor verification and performance evaluation.



Xu Xiaohong, born in 1967. Professor in Shanghai High Performance IC Design Center. Her main research interests include processor design and verification.



Li Yanzhe, born in 1983. Assistant professor in Shanghai High Performance IC Design Center. Her main research interests include processor verification and simulation.