

RapidIO IP 核的验证方法研究

王玉欢, 田 泽, 蔡叶芳

(中国航空计算技术研究所, 陕西 西安 710068)

摘 要: 串行 RapidIO 是针对高性能嵌入式系统芯片间和板间互连而设计的, 是未来十几年中嵌入式系统互连的最佳选择之一。在以 RapidIO 为接口的 SoC 设计中, 对 RapidIO IP 核的验证是其基础。基于对 RapidIO 协议的理解, 研究了 RapidIO IP 核功能验证的方法、验证平台的搭建以及验证测试过程的实施, 提出了虚拟平台验证与 FPGA 原型验证相结合的验证方法。该验证过程搭建了可靠的验证平台, 为 RapidIO IP 核的可靠工作提供了保证。文中的研究工作, 从验证思路和方法上对于类似设计的验证具有一定的参考价值。

关键词: RapidIO; IP 核; 验证平台; 验证

中图分类号: TP39

文献标识码: A

文章编号: 1673-629X(2011)07-0183-03

Verification Method Research for RapidIO IP Core

WANG Yu-huan, TIAN Ze, CAI Ye-fang

(Aeronautical Computing Technique Research Institute, Xi'an 710068, China)

Abstract: Serial RapidIO is designed for the connection among chips of high performance embedded system and that among boards. It is one of the best choices for the interconnection of embedded system in the future. In the SoC design with the RapidIO as its interface, verification for RapidIO core is the foundation. Based on the understanding of RapidIO protocol, take a research of the method of RapidIO core functional verification, the foundation of verification terrace and the implementation of the verification test process, raised the verification method that combined the virtual terrace verification and FPGA archetype verification. This verification process found a dependable verification testbench, guaranteeing the dependable work of RapidIO IP core. For similar designed verification, research in this essay has some reference value in mentality and method field of verification.

Key words: RapidIO IP core; testbench; verification

0 引言

RapidIO 技术最初是由 Freescale 和 Mercury 共同研发的一项互连技术^[1], 研发初衷是将其作为处理器的前端总线, 用于处理器之间的互连。随着互连技术的发展, 目前, RapidIO 已由一个标准定制阶段进展到产品阶段, 成为电信、通讯以及嵌入式系统内芯片与芯片之间互连技术的生力军。

在以 RapidIO 为接口的高性能 SoC 设计中, 流片前对 RapidIO IP 核的充分验证是保证集成电路设计成功的基础和关键^[2]。如何构建能模拟芯片工作环境、高效可复用的验证平台是 SoC 设计中值得关注的重要问题之一^[3,4]。

文中结合项目实践, 基于对 RapidIO 协议的研究,

基于虚拟平台验证和 FPGA 原型验证相结合的方法, 通过软硬件协同验证手段^[5], 对 RapidIO IP 核进行了验证, 达到了比较理想的代码覆盖率和功能覆盖率, 有效地提高了验证效率。

1 RapidIO 协议概述

如图 1 所示, 为了满足灵活性和可扩展性的要求, RapidIO 协议分为三层: 逻辑层、传输层和物理层^[6]。

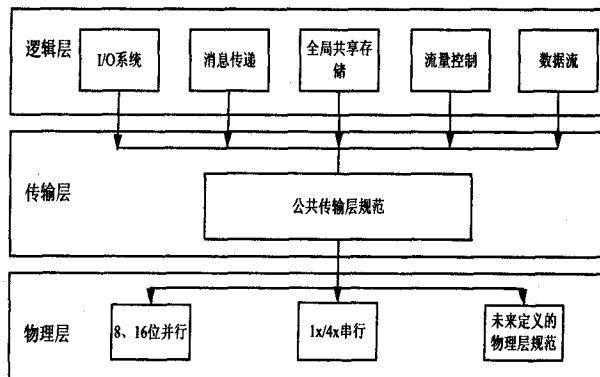


图 1 RapidIO 协议分层

收稿日期: 2011-02-28; 修回日期: 2011-05-10

基金项目: 中国航空工业集团公司技术创新基金(2009D63120)

作者简介: 王玉欢(1984-), 女, 陕西西安人, 硕士, 助理工程师, 主要从事数字集成电路设计和验证等方面的研究; 田 泽, 博士, 研究员, 研究方向为 SoC 设计、嵌入式系统设计、VLSI 设计等。

1.1 逻辑层协议

逻辑层协议位于 RapidIO 协议分层的最高层,定义了全部协议和包的格式,它们为端点器件和完成任务提供了必要的信息。RapidIO 支持的逻辑层事务主要是 IO 逻辑操作和消息传递操作两种^[7]。

1.2 传输层协议

传输层协议位于 RapidIO 协议分层的中间层,定义了 RapidIO 的地址空间以及在端点器件间传输包所需的路由信息。

1.3 物理层协议

物理层协议位于 RapidIO 协议分层的最底层,包括器件接口的细节,如包传输机制、流量控制机制、电器特性和错误管理机制等^[8]。

2 RapidIO IP 核介绍

如图 2 所示,RapidIO IP 核是文中的验证对象。它可以划分为 6 大部分:实现逻辑及传输层协议包括负责事务组包、拆包等功能的逻辑及传输层模块(Logical/Transport Layer Module)、实现物理层协议包括包的控制符号传送、流量控制、错误管理等功能的物理层模块(Physical Layer Module)、负责对寄存器进行读写操作的寄存器管理模块(Register Manager Module)、集中实现各层寄存器的寄存器组模块(Register Group Module)、时钟及复位模块(Clock and Reset Module)、用户定义模块(User Design Module)。

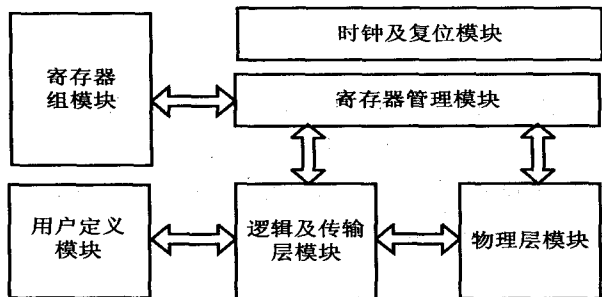


图 2 RapidIO 核功能框图

3 RapidIO IP 核验证方法

虚拟原型验证平台优点是方便纠错,但速度较慢,对于大数据量的验证具有局限性。而 FPGA 的优点是速度快,但可观测性较差。因此,采用两种验证手段相结合的方法可以有效地平衡验证效果和验证效率。

从验证流程角度可将其验证分为以下步骤:

(1) 根据 RapidIO IP 核的功能规范,针对其模块功能,制定出完备的虚拟平台验证规范以及 FPGA 平台验证规范;

(2) 依据验证规范,开发完备的虚拟验证平台、FPGA 平台进行 IP 核的验证。

(3) 最终通过代码覆盖率和功能覆盖率两方面保证验证工作的充分性及完备性。

4 RapidIO IP 验证规范

验证规范是开展验证工作的基础,因此,在验证规范制订阶段,必须保证其完备性。根据 RapidIO 协议的特点,RapidIO IP 的验证规范的制订可分为以下两个阶段。

第一阶段,为黑盒验证阶段。即把 RapidIO IP 核当作黑盒,不关心其内部实现方法。如表 1 所示^[9],从应用的角度出发,验证 RapidIO IP 核是否能满足协议规定以及应用需求,并指定相应的验证规范。使得验证过程能够覆盖到所有的应用情况。

表 1 黑盒验证阶段验证功能划分

表格编号	验证对象	验证功能
1	维护操作	本地维护写操作
		本地维护读操作
		远程维护写操作
		远程维护读操作
2	NWRITE 操作	本地访问远端存储器空间的 NWRITE 操作
		本地访问远端配置空间的 NWRITE 操作
		远端访问本地存储器空间的 NWRITE 操作
		远端访问本地配置空间的 NWRITE 操作
...

第二阶段,为协议一致性验证阶段。在该验证阶段,主要关注 RapidIO IP 核的设计实现是否符合协议规定。如表 2 所示,验证 1x/Nx 模式初始化状态机跳转情况是否与协议一致。该部分验证的对象在第一阶段已经覆盖了大部分内容。只是在第二阶段验证的角度与第一阶段不同。同时,这部分验证是在第一阶段验证的基础上的查漏补缺,确保协议相关部分都能够在测试项量中覆盖到。

表 2 协议一致性验证阶段验证功能划分

表格编号	验证对象	验证功能
1	Lane 对齐状态机	NOT_ALIGNED 状态
		NOT_ALIGNED1 状态
		NOT_ALIGNED2 状态
		ALIGNED 状态
2	1x/4x 模式初始化状态机	...
...

5 RapidIO IP 的验证

5.1 虚拟平台验证

如图 3 所示,RapidIO 端点的仿真平台包括以下部分^[10]:

- (1) 被验证对象(图 3 所示的 ep_top.v);
- (2) 验证激励(通过主机模型发出);
- (3) 仿真主机模型(图 3 所示的 Simulation Host);
- (4) 串行链路协议监视器(图 3 所示的 srio_snooper_top.v);

(5) 发起方以及目标方的用户参考设计(图 3 所示的 User Design)。

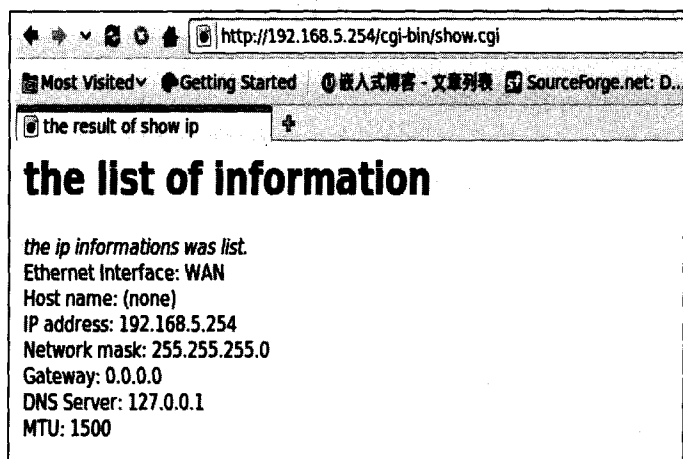


图 3 仿真验证平台

被验证对象,即 RapidIO IP 核;testbench 设计的核心是仿真主机模型及验证激励的开发;验证激励的开发,即以第二部分中提到的验证规范为依据,使用 HDL 语言开发测试用例。

如图 4 所示,仿真主机模型给出了如何发起设计的 testbench 的端点操作。仿真主机模型发起向目标用户设计的存储器的写数据请求。使用 SWRITE, NWRITE 或 NWRITE_R 命令写入到存储器中。使用 NREAD 命令把数据从目标用户设计的存储器中读回,并且在仿真主机模型中与最初发送的数据进行比对。通过仿真主机模型中的打印信息可以打印出状态和错误信息,使得用户能方便地监控 testbench 的运行情况。

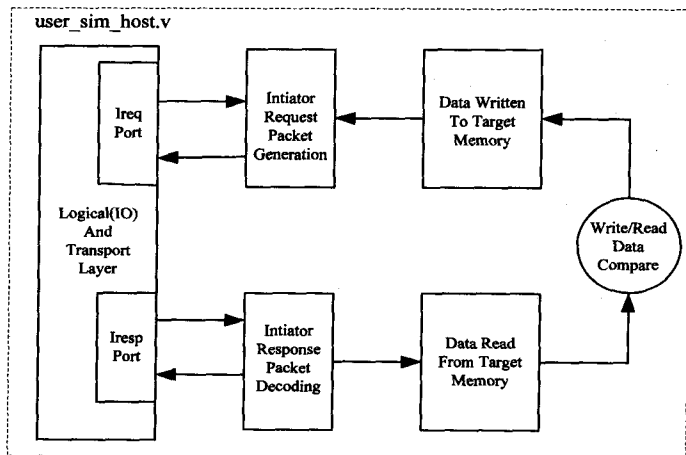


图 4 主机模型结构图

串行链路协议监视器用于解码 RapidIO 串行物理层传输的数据。协议监视器用于发送方和接收方之间的串行物理层传输。随着数据在链路中的传输,会同时向发送和接收记录文件中写入调试信息。这些记录信息文件能用于调试。

发起方以及目标方的用户参考设计示意出如何使

用 RapidIO 逻辑层核的接口。

根据 RapidIO IP 核验证规范中定义的验证内容和具体实施办法,开发相应的测试用例(HDL 激励)对被测模块进行验证。配合相应的仿真脚本,运行仿真,施加测试激励后,在 Modelsim 仿真工具中通过查看波形、输出测试报告等多种方式来对验证结果进行判断其是否与设计预期一致。通过观察对应的信息报告、代码覆盖率和仿真波形确定最终仿真的完成。

5.2 FPGA 原型验证

FPGA 虚拟原型验证平台如图 5 所示, FPGA 芯片使用 Xilinx 的 LX30T 芯片,将 RapidIO IP 逻辑综合后下载到 FPGA 芯片中。FPGA 验证板上除了 FPGA 芯片,还有与虚拟原型验证平台对应的验证组件、存储器模型。除此之外还有电源、电容等器件单元。首先将 C 语言编程的测试程序烧入 Flash 单元。然后加电,上电后, FPGA 验证板上的处理器从 Flash 中开始取指令,执行测试程序。这样,就达到了与 RapidIO IP 核实际工作相同的效果^[11,12]。

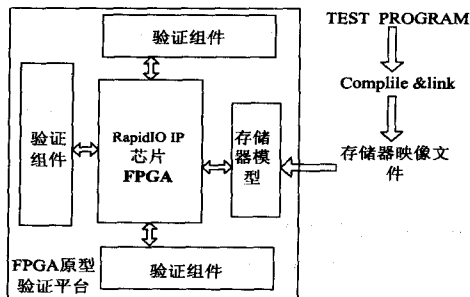


图 5 虚拟原型验证平台

由于 FPGA 原型验证效率较高,因此通过 FPGA 原型验证,可以进行大数据量的测试验证工作,可以与虚拟平台验证起到很好的互补效果。

6 结束语

文中介绍了 RapidIO IP 核功能验证的方法、验证平台的搭建以及验证测试的实施过程,提出了虚拟平台验证与 FPGA 原型验证相结合的验证方法,对于类似设计的验证均有参考价值。存在问题是一些协议符合性验证在现有验证平台上难以实现,还需要借助其他一些测试设备才能完成,这部分工作后续应当进一步补充完善。

参考文献:

[1] 孙灯亮. RapidIO 测试思路和方法[J]. 电子质量, 2009

(下转第 189 页)

洗、数据替换、数据计算、数据验证、数据加解密、数据合并、数据拆分等。这些组件可插拔,可任意组装,各组件之间通过数据总线共享数据。

数据清洗主要是针对源数据库中,对出现二义性、重复、不完整、违反业务或逻辑规则等问题的数据进行相应的清洗操作。在清洗之前需要进行数据质量分析,以找出存在问题的数据。数据清洗和纠错处理主要实现如下:对源数据的数据质量进行分析;建立清洗规则;根据清洗处理的结果进行纠错处理;差异比对和修复;比较任意两个数据库的两张表,可以指定比对规则;自动生成差异列表;根据比对的差异结果,自动修复目标表等^[12]。

2.5 集成数据加载组件

集成数据加载将从数据源系统中抽取、转换后的数据加载到数据仓库系统中。数据加载策略要考虑加载周期及数据追加策略两方面的内容。根据高校教学、管理业务的实际情况,加载周期要综合考虑业务分析需求和系统加载的代价,对不同的子系统的数据采用不同的加载周期,但必须保存同一时间数据的完整性。

集成数据加载组件主要功能是解析元数据描述 XML 文件。XML 文件解析模块将经数据转换层转换完成的元数据描述 XML 文件,通过对象-关系映射进行解析,最后将转换结果存入中心数据库中。

数据的追加策略根据数据的抽取策略和业务规则确定,本系统采用三种类型:直接追加、全部覆盖、更新追加。对于流水数据采用直接追加方式;如果抽取数据本身已包括数据的当前和所有历史状况,则对目标表采用全部覆盖方式;对于需要连续记录业务的状态变化,并用当前的最新状态同历史状态数据进行对比的情况可采用更新追加的方式。

3 结束语

根据高校数据集成系统的需求,构建了 ETL 系统

模型和实现方案。该 ETL 系统基于适配器和中间件技术、具有多线程并发运行、增量抽取、易扩展、支持跨平台的功能。该系统可以很好地解决数据集成中多数数据融合、数据不一致和数据同步更新等问题。

参考文献:

- [1] 张上游,王 燕. 数字化校园建设中数据分析的研究[J]. 西南民族大学学报,2009,35(4):871-873.
- [2] 王晓虹,王国仁,于勇前,等. 电信闭环决策支持系统的研究与实现[J]. 计算机应用研究,2008,25(4):1247-1249.
- [3] 秦学勇,刘 栋. 数据仓库的可扩展性研究与设计[J]. 计算机技术与发展,2009,19(5):65-67.
- [4] 李 颖,郝克刚,葛 玮. 基于电信数据仓库系统的 ETL 研究与设计[J]. 计算机应用与软件,2009,30(1):178-180.
- [5] 许 力,牟晓光,马云存. 并行 ETL 过程的研究与实现[J]. 计算机工程与应用,2009,45(13):170-172.
- [6] 宋 杰,王大玲,鲍玉斌,等. 一种元数据驱动的 ETL 方法的研究[J]. 小型微型计算机系统,2007,12(12):2167-2173.
- [7] Panos V, Alkis S, Manolis G, et al. A generic and customizable framework for the design of ETL scenarios[J]. Information Systems,2005, 30(7):492-525.
- [8] Alkis S, Dimitrios S, Malu C. Representation of conceptual ETL designs in natural language using Semantic Web technology[J]. Data & Knowledge Engineering,2010,69(1):96-115.
- [9] 黄怀毅,杨路明. 一种轻量级架构的 ETL 系统设计与实现[J]. 计算机技术与发展,2008,18(6):202-205.
- [10] 戴 浩,杨 波. ETL 中的数据增量抽取机制研究[J]. 计算机工程与设计,2009,30(23):5552-5555.
- [11] Alkis S, Panos V, Timos S. State-Space Optimization of ETL Workflows[J]. IEEE Transactions on Knowledge & Data Engineering,2005, 17(10):1404-1419.
- [12] 包从剑,李星毅,施化吉. 可扩展和可交互的数据清洗系统[J]. 计算机技术与发展,2007,17(7):84-86.

(上接第 185 页)

(11):18-19.

- [2] 吴英攀,于立新. 基于层次化验证平台的存储器控制器功能验证[J]. 微电子学与计算机,2009(2):25-28.
- [3] 王世好,王歆民,刘明业. 嵌入式系统软硬件协同验证中软件验证方法[J]. 计算机研究与发展,2005,42:514-519.
- [4] 马 宁,李 玲,田 泽,等. ARINC659 总线协议芯片的仿真验证[J]. 计算机技术与发展,2010,20(1):205-206.
- [5] 杨海波,田 泽,蔡叶芳,等. FC IP 软核的仿真与验证[J]. 计算机技术与发展,2009,19(9):168-172.
- [6] RapidIO Interconnect Specification Rev2.0,03[S]. [s.l.]: Copyright RapidIO Trade Association,2008.

- [7] 梁小虎. 高速串行总线 RapidIO 与 PCI Express 协议分析比较[J]. 航空计算技术,2010(3):164-167.
- [8] 黄振中,柴小丽. 基于 Vxworks 的 PCI-RapidIO 桥驱动设计[J]. 计算机工程,2010(3):239-240.
- [9] Serial RapidIO User Guide v5.1 UG503[M]. [s.l.]:Xilinx, 2008.
- [10] Serial RapidIO User Guide v4.3 UG247[M]. [s.l.]:Xilinx, 2008.
- [11] 郭 蒙,田 泽,蔡叶芳,等. 1553B 总线接口 SoC 验证平台的实现[J]. 航空计算技术,2008,38(6):99-101.
- [12] 田 靖,田 泽. AFDX-ES SoC 虚拟仿真平台的构建与应用[J]. 计算机技术与发展,2010,20(8):192-198.