

基于芯粒的多核处理器目录控制器的敏捷验证

罗莉^{1,2}, 冯权友^{1,2}, 周理^{1,2}, 铁俊波^{1,2}, 龚锐^{1,2}, 潘国腾^{1,2}

(1. 国防科技大学计算机学院, 湖南 长沙 410073; 2. 先进微处理器芯片与系统重点实验室, 湖南 长沙 410073)

摘要: 芯粒已成为芯片设计的新趋势, 随着微处理器结构向多核化、众核化发展, 高速缓冲存储器 (Cache) 一致性协议也日趋复杂。在多芯粒集成的微处理器中, Cache 一致性协议的验证更是一个具有重要应用价值的技术问题。基于目录的 Cache 协议是最广泛应用的硬件实现方法, 针对自主研发的分布式目录控制器 (DCU), 研究多芯粒集成的处理器目录控制器的敏捷验证技术, 随机测试采用否定选择算法 (NSA) 的优化机制。实验结果表明: 随机测试激励减少 40% 的条件下, 功能覆盖率较未使用否定选择算法的随机测试提高了 28 个百分点; 随机测试激励和定向测试激励相结合, 功能覆盖率最终达到 100%。同时设计一致性检查器, 监视一致性协议事务的生命周期, 用于快速定位设计错误, 追踪问题的精确场景。实践中, 该检查器定位的协议错误占设计错误的 90%, 有效提高了验证效率和验证质量。

关键词: 芯粒; 高速缓冲存储器一致性协议; 分布式目录控制器; 否定选择算法; 敏捷验证

中图分类号: TP302

文献标志码: A

DOI: 10.19678/j.issn.1000-3428.0070593

Agile Verification of Directory Control Unit for Chiplet-based Multi-core Processors

LUO Li^{1,2}, FENG Quanyou^{1,2}, ZHOU Li^{1,2}, TIE Junbo^{1,2}, GONG Rui^{1,2}, PAN Guoteng^{1,2}

(1. College of Computer Science, National University of Defense Technology, Changsha 410073, Hunan, China;

2. Key Laboratory of Advanced Microprocessor Chips and Systems, Changsha 410073, Hunan, China)

【Abstract】 Chiplets have emerged as a new trend in chip design. With the evolution of microprocessor architectures toward multi-core and many-core implementations, the complexity of Cache coherence protocols has increased substantially. For microprocessors integrating multiple chiplets, the verification of Cache coherence protocols is a technically critical issue with significant practical value. The directory-based Cache coherence protocol is the most widely adopted hardware-implementation approach. Focusing on a self-developed distributed Directory Controller Unit (DCU), this study investigates agile verification technologies for DCU in multi-chiplet integrated processors. An optimized mechanism based on a Negative Selection Algorithm (NSA) is employed in random testing, where functional coverage was improved by 28 percentage points under a 40% reduction in random test stimuli. By combining random and directed test stimuli, the functional coverage ultimately reaches 100%. Additionally, a coherence checker is designed to monitor the entire lifecycle of the coherence protocol transactions. This enables the rapid localization of design errors and accurate tracing of problematic scenarios, with 90% of the design bugs successfully identified. This approach effectively enhances verification efficiency and quality.

【Key words】 chiplets; Cache coherence protocol; distributed Directory Control Unit (DCU); Negative Selection Algorithm (NSA); agile verification

0 引言

近年来, 芯粒已成为芯片设计的新方向和新趋势。在后摩尔时代先进工艺的巨大成本压力下, 为了分担风险以及延续摩尔定律, 芯粒复用因其能有效提高芯片良率和降低设计成本, 已成为芯片敏捷研发的重要方式之一。

目前, 国产高性能微处理器的发展, 一方面遇到

后摩尔时代集成电路自身的发展瓶颈, 另一方面受到国外对国内先进工艺的限制, 其在性能的提升上遇到工艺墙、功耗墙、存储墙和可扩展性墙, 性能提升速度放缓。基于芯粒构建多核处理器架构, 能够解决国产高性能多核处理器性能扩展的难题。

高速缓冲存储器 (Cache) 是微处理器中的关键部件。随着处理器结构向多核化、众核化发展, Cache 一致性协议也日趋复杂, 如何保证一致性协

基金项目: 国家自然科学基金青年基金项目 (62203457); 并行与分布式处理国防科技重点实验室开放基金 (WDZC20235250112)。

作者简介: 罗莉 (CCF 会员), 女, 研究员、博士, 主研方向为高性能计算机体系结构、高性能通用微处理器设计与验证; 冯权友, 副研究员、博士; 周理, 助理研究员、博士; 铁俊波、龚锐、潘国腾, 副研究员、博士。

收稿日期: 2024-11-07

修回日期: 2025-02-24

E-mail: li_luo@nudt.edu.cn

议的正确性,一直是工业界和学术界关注的研究热点。随着处理器核数增多,基于目录的 Cache 一致性协议成为最广泛应用的实现方法。与基于总线侦听的协议相比,目录协议采用点对点的方式,精确发送一致性请求,具有良好的可扩展性和可靠性^[1-2]。多芯粒集成微处理器中,基于目录的 Cache 协议实现和验证更是一个极具挑战性的难题。

本文研究了基于芯粒的多核处理器目录控制器(DCU)的敏捷验证。针对自主研发的分布式目录控制器,描述所设计的 4 芯粒集成的多核处理器目录控制器敏捷验证环境,详细介绍提高敏捷验证效率的关键技术:基于否定选择算法(NSA)的随机测试优化机制和一致性监视检查器。在此基础上,给出实验结果。最后,对本文加以总结,并指出进一步的工作方向。

1 相关研究

芯粒技术提高了芯片的可扩展性,避免了随着晶圆规模增大出现的良率快速降低的问题,同时降低了芯片设计成本。通过多个芯粒构建处理器,不同芯粒采用不同工艺的方式,降低了成本,提高了良率,提供了更丰富的产品形态,能够最大化地复用芯粒架构,实现系统良好的可扩展性与兼容性。Intel 于 2021 年发布的 Sapphire Rapids 处理器包含 4 个同构“芯粒”^[3],通过全局 Mesh 网络实现众核扩展。AMD 早在 2018 年就在 Zeppelin、Rome 和 Turin 等 EPYC 系列处理器中采用芯粒技术^[4-6]。在 AI 计算领域,NVIDIA 基于芯粒方案设计了 AI 计算芯片 Simba^[7],Simba 采用 16 nm 工艺,每个芯粒最高算力为 4 TOPS,通过 36 个芯粒的集成,实现最高 127.8 TOPS 的算力。芯粒已成为芯片设计的新方向和新趋势,2018 年 10 月,多家半导体企业和 EDA 工具开发企业成立开放专用域架构(ODSA)组织^[8],旨在制定芯粒开放标准,2022 年推出了芯粒互联白皮书 UCle^[9],提供可靠的数据传输和链路管理。此后,《芯粒互联接口标准》^[10]和《集成芯片与芯粒技术白皮书》^[11]陆续发布,国内芯粒技术发展进入了标准化和生态化建设的关键阶段。

Cache 是现代微处理器存储层次中的关键部件,硬件支持的 Cache 一致性协议随着处理器核数增多变得愈加复杂,协议的状态空间呈指数级增长,甚至出现爆炸现象,如何保证 Cache 一致性协议的正确性,一直是工业界和学术界研究热点^[12]。同时,对这些复杂的协议进行验证也变得十分棘手。

目录具有良好的可扩展性和可靠性,是最广泛应用的实现方法^[13-19]。文献[20]提出了硬件实现的 Cache 一致性模型验证方法,利用基于约束图的检查方法检测缺陷,通过 Gem5 模拟器测试 2 核、4 核的 MESI 目录协议有效性,并评估硬件实现的面积和功耗。文献[21]提出了一种片上信号记录方法,利用 Cache 一致性协议规范来观察判断一致性事务的信号状态,信号状态不匹配就会标记错误,应用于 MESI 目录协议的 MIPS 4 核处理器,有效检测设计错误。文献[22]提出了一种使用商空间(quotient space)的动态生成测试激励,在 Gem5 模拟器上实现 Cache 一致性协议的各状态转换覆盖,相比基于状态空间广度优先的生成测试激励方法,测试激励数目减少 50%。文献[23]介绍了片间直连 Cache 一致性协议,讨论目录和 Token 协议设计关键技术,利用 Simics 和 GEMS 的系统模拟器,分析了两者在运行性能、带宽需求、可扩展性等方面的优缺点。目前学术上研究目录 Cache 一致性协议的验证,一般在结构级模拟器上进行建模,由于状态空间爆炸问题,结构级模拟器建模适用于 4 核、多核处理器等小规模系统验证,因此面向众核处理器的系统验证,基于仿真的验证方法是一个值得探索研究的领域。

在芯粒时代,基于仿真的 Cache 一致性协议的敏捷验证更是一个具有重要应用价值的技术问题。本文针对自主研发、硬件实现的分布式目录控制器,提出了一种基于芯粒的多核处理器目录控制器的敏捷验证方法,采用否定选择算法对随机测试进行优化。实验结果表明,测试激励减少 40%的条件下,功能覆盖率提高了 28%;随机激励和定向激励最终实现功能覆盖率 100%。本文还提出一种一致性协议专用检查器,用于快速定位错误,追踪问题的精确场景。实践中,利用该检查器定位的协议错误占总设计错误的 90%,可有效提高查错效率。

2 基于芯粒的多核处理器

多核、众核微处理器的常用实现方式是同构集成对称多处理器(SMP)系统,即集成多个成熟的、结构相同、功能强大的通用处理核心。本文设计的目标处理器芯片的基本芯粒单元(Die)为 16 核 SoC 结构,如图 1 所示,其采用 SMP 结构,包括处理器核(Core)、Cache、片上网络(NoC)、目录控制器、存储器(MCU)、跨芯粒接口(LIU),即 Die2Die link Unit。

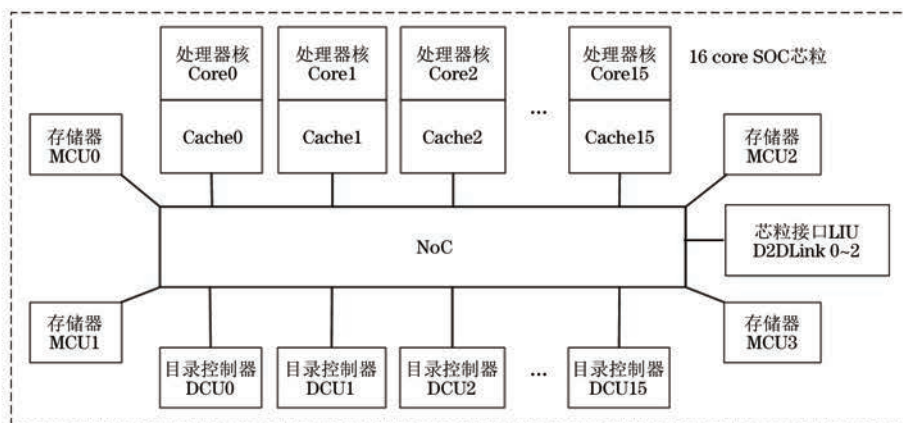


图 1 Die:16 核 SoC 芯粒结构

Fig.1 Die:16-core SoC chiplet structure

图 1 所示的 Core 是 CPU 核,完成指令的调度与执行;Cache 是高速缓存,包含私有 L1 Cache 和共享的 L2 Cache;NoC 提供 Cache 之间、Cache 和存储器之间的互连通路;DCU 记录每个数据块 Cacheline 在各芯粒的 16 个 Cache 中的副本使用情况,采用监听请求与响应的方式维护全局数据一致性,失效时才产生存储器的访问请求;目录控制器采用分体设计,形成分布式目录;MCU 外接 DDR SDRAM 存储器,实现存储器的读写访问控制。LIU 实现多芯粒间可靠、实时、完整、保序的数据传输,3 个直连接口支持 4 芯粒集成封装 64 核处理器,多芯粒集成 32 核、64 核处理器示意图如图 2 所示。LIU 是面向本项目芯片架构的定制型 IP 设计,支持本项目芯片指令集架构的专用一致性协议,UCIe^[9]协议层支持已经广泛使用的协议 PCIe 6.0、CXL 2.0、CXL 3.0,CXL 协议虽然能实现一致性互连,但与 X86 指令集架构深度绑定,与本项目芯片的特定指令集架构不兼容。

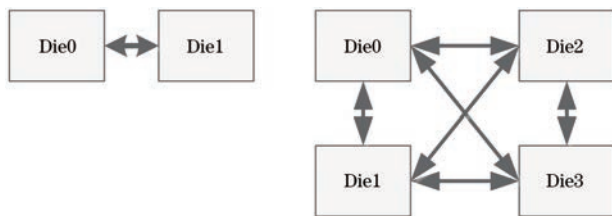


图 2 多芯粒集成 32 核、64 核处理器示意图

Fig.2 Schematic diagram of multi-chiplet integrated 32/64-core processor

在目标芯片中,Cache 数据块有 5 种状态:MOESI(Modified, Owned, Exclusive, Shared, Invalid)。为提高一致性协议事务处理的吞吐率,对指令集规范存储模型中定义的共享型、支持乱序处理、可缓存型的物理地址空间进行分片,确保每一个目录控制器管理固定地址空间的 Cache 块(Cacheline)的数据一致

性,不同的目录控制器所管理的空间不重叠;物理地址空间中的任一 Cacheline,其对应的宿主目录控制器在全芯片中是唯一的。多芯粒结构下,宿主目录控制器可能不在本芯粒内,可以位于另一芯粒上。目录控制器的功能是支持全局数据共享,跟踪记录各个 Cache 数据的使用,失效时产生到 MCU 的访问请求,同时通过监听请求与响应的方式处理 Cache 一致性协议报文。

在多芯粒系统中,读事务的一致性处理典型流程如图 3 所示。当芯粒 Die0 的 Cache0 不命中,需要访问位于 Die1 中的 Cacheline 时,目标地址属于芯粒 Die1 的 DCU0 管理,发出共享读请求 ReadShared 事务,会先由 NoC、LIU 传送到 Die1 的 DCU0,ReadShared 进入 Die1 的 DCU0 目录处理流水线,触发 Cache 一致性协议操作。如果 DCU0 的流水线命中,ReadShared 最终引起监听取数操作(Snoop),芯粒 Die1 的 DCU0 将监听返回的结果 CompData 通过片上网络、LIU 转发返回给请求发起方 Die0 的 Cache0。

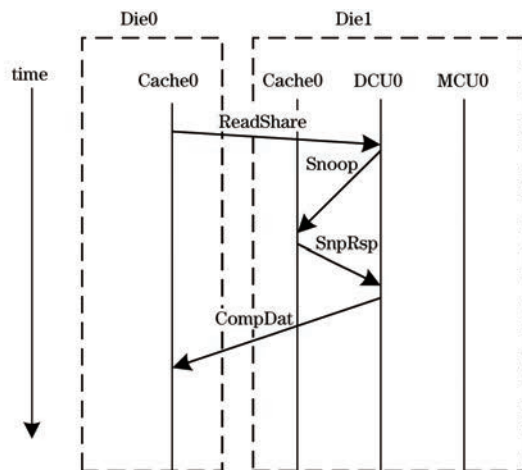


图 3 跨芯粒的读操作协议流程

Fig.3 Cross-chiplet read operation protocol flow

在多芯粒系统中,写事务的一致性处理典型流程如图 4 所示。芯粒 Die0 的 Cache0 发出写请求,当目标地址属于芯粒 Die1 的 DCU0 管理时,写请求 WriteUnique 会先由 NoC、LIU 传送到 Die1 的 DCU0,WriteUnique 请求进入 Die1 的 DCU0 目录处理流水线,执行 Cache 一致性协议操作,当系统中存在多副本时,目录控制器启动监听作废操作(Snoop),Die1 的 DCU0 收到所有监听作废响应 SnpRsp 后,才能发出写响应操作,芯粒 Die0 的 Cache0 收到写响应后,发出 WrtDat 通过 NoC、LIU 转发 Die1 的 DCU0,Die1 的 DCU0 执行一致性协议操作,发起访存写 McuWrt 操作,写回 MCU0。

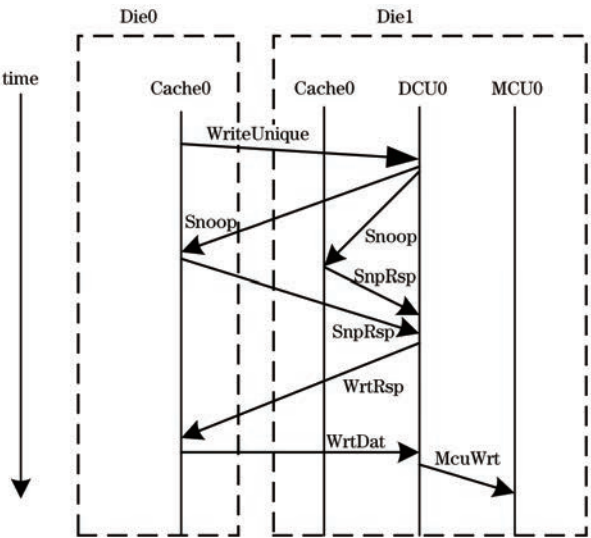


图 4 跨芯粒写操作协议流程
Fig.4 Cross-chiplet write operation protocol flow

在目录协议中,目录控制器记录了所有 Cache 中数据副本的使用情况,并根据当前收到的报文命令和目录状态进行跟踪与修改,维护整个处理器中各 Cache 之间数据一致性。目录实现方式分为集中式和分布式,集中式目录设计简单,但目录控制器周围的网络流量常常是访问热点,进而影响一致性传输延迟,造成网络的功耗热点。为了提高多核处理器目录并行处理能力,本文的目录控制器采用了分布式目录实现方式^[20],目标系统的 Cache 一致性事务均匀分散到多个目录控制器,每个目录控制器在 NoC 中具有唯一的网络地址标识,目录信息保存在采用组相联方式管理的存储体中(测试配置为 24 路组相联组织)。为进一步提高并行处理性能,每个目录控制器分为两个存储体(bank),形成两条事务处理的流水线,双存储体按地址 addr[6]交叉访问。每个 Cache 的数据块 Cacheline 回填前,目录控制器都分配一个目录项,记录该 Cacheline 的使用情况。

DCU 每个目录项信息如图 5 所示,主要包括 tag、busy、valid、vector、ecc 校验等信息。其中,目录的标志位 tag,即访存物理地址的一部分,通常是高位地址段;busy 定义了目录项的忙闲状态,如监听请求没有完成,或者不能处理相同地址的新请求,则设置忙状态;valid 是目录项有效位,若目录项有效,表示至少有一个 Cache 有数据副本;vector 代表共享者向量,定义了有数据副本的 Cache 的编号;vector 宽度为 64 位,可支持 4 个芯粒集成的 64 核处理器;ecc 校验位是目录项的海明校验位。

valid	busy	tag	vector	ecc
-------	------	-----	--------	-----

图 5 DCU 目录项 entry 的定义
Fig.5 Definition of a directory entry in DCU

3 目录控制器的敏捷验证环境

目标芯片中一个芯粒的 16 个目录控制器是相对独立的,DCU0 管理的只是映射到存储器 MCU0 地址空间的一致性信息。所以,目录协议实现的功能验证可以针对单个目录控制器进行测试。针对 DCU0 构建模拟验证环境,如图 6 所示,在 4 个芯粒集成 64 核处理器的验证环境中,Cache 是自主 IP 设计,敏捷验证环境选用了 Cache 的模型替代,它是一个时钟精确的功能模型,模拟实现了处理器中 Cache 的功能。MCU 模型也是时钟精确的功能模型,用于模拟访存的读、写事务。验证环境中的一致性协议检查器是针对 DCU0 的一致性检查工具,用于监视映射到存储器 MCU0 的访存地址空间一致性协议事务的生命周期。

DCU0 可以接收处理来自本芯粒的 16 个 Cache 或来自远程芯粒的 48 个 Cache 的请求。DCU0 输入、输出的 Cache 一致性信息都是以 NoC 报文格式传输,可分为 4 类:Cache 读写请求,监听请求,读写响应,监听应答。DCU0 中目录失效时,形成 MCU 访存请求;目录命中时,会形成 Cache 的监听请求。DCU0 也以 NoC 报文的形式接收,其中包括 MCU 存储器的读响应和各个被监听 Cache 的应答信息。

设计敏捷验证环境面临两大问题:一是测试激励的质量,二是有效定位设计错误。本文采用基于否定选择算法的随机测试提高功能覆盖率,同时设计了一致性协议检查器,监视一致性协议事务生命周期,及时定位设计错误,在有限的研制周期内完成验证。

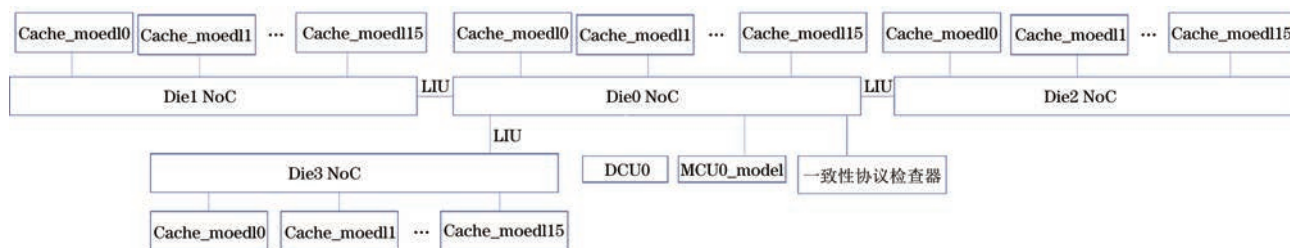


图 6 多芯粒集成的处理器的目录控制器敏捷验证环境

Fig.6 Agile verification environment for the DCU in a multi-chiplet integrated processor

3.1 否定选择算法的随机测试

测试激励是敏捷验证环境的驱动力,目录控制器测试一般采用随机测试,但完全随机程序会存在大量冗余测试,或者无效测试激励,降低验证效率。目录控制器验证的主要功能,一是对 Cache 一致性报文正确处理,二是目录功能正确性,包括目录命中、替换、目录忙命中、目录满命中、目录不命中等功能点。本文提出一种基于否定选择算法的随机测试,其自动产生有效的受约束随机测试激励,能够在减少冗余测试激励的同时,尽快提高功能点覆盖率,减少测试时间。

否定选择算法(NSA)是人工免疫系统的主要算法之一^[24-26],人体中使用抗体来区分自体和非自体抗原。否定选择算法的主要目标是区分自我样本和非自我样本,而只有自我样本是可用的。

基于否定选择算法的随机测试,首先针对多核处理器测试激励特征,进行否定选择算法的抗体符号编码,其次随机生成初代抗体,再次随机生成新抗体,比较新抗体与种群其他抗体差异性,大于差异阈值则加入种群,最后直至达到迭代次数或者种群数目达到设定范围。

采用符号编码方式进行抗体编码,每一位域代表一个特征矢量,定义测试向量以下 5 个方面的特征:1) sendid 的编号,表示了数据来自 64 个 Cache 中的哪个 Cache,可能是本地芯粒或者跨芯粒的 Cache;2) 报文的操作码(opcode),如 none-cache 读、共享读、none-cache 写、Cache 维护等命令;3) 测试所包含的每一条报文的地址相关性,如果相关,那么读写的地址是相同的,不相关地址选择随机,或者与前一个报文的地址连续递增;4) bank 编号,即目录存储体的编号;5) len 的编码,表征报文请求是 1、2、4、8、16、32 还是 64 Byte 或者无数据携带。

一个抗体 Gen 编码表示如图 7 所示,抗体编码 (4,5,1,1,0),定义 Cache4 发出了一条 op5 操作,访问地址与第一个抗体地址相同,访问地址映射到 bank1,读 8 位的数据。

4	5	1	1	0
---	---	---	---	---

图 7 一个抗体 Gen 实例

Fig.7 An example of antibody Gen

一个测试激励由多个测试向量,即抗体种群构成,抗体种群已产生 48 个抗体,如图 8 所示,第 1 个抗体 Gen0 是 Cache4 发出了一条操作码是 op5 的请求,地址是随机地址,访问目录 bank1,读取 1 Byte 的数据;第 2 个抗体 Gen1 是 Cache0 发出了一条操作码是 op1 的请求,地址与 Gen0 地址相同,地址相关,访问目录 bank1,读取 64 Byte 的数据;第 3 个抗体 Gen2 是 cache3 发出了一条操作码是 op4 的请求,地址与前一个抗体 Gen1 地址连续,访问目录 bank0,读取 16 Byte 的数据;……;第 48 个抗体 Gen47 是 Cache 47 发出操作码是 op0 的请求,地址与第 1 个抗体地址相同,目录 bank 地址是 1,读取 64 Byte 的数据。

Gen0	Gen1	Gen2	Gen47
4 5 0 1 0	0 1 1 1 5	3 4 2 0 4	47 0 1 1 5

图 8 抗体种群表示的测试向量

Fig.8 A test antibody population

本文所述抗体差异性利用欧氏距离定义,计算任意抗体 n_{ik} 和种群中抗体 n_{il} 之间的欧氏距离的函数表达式如下:

$$D_{n_{ik}} = \sum_{b=1}^F \sum_{a=1}^E (Gen_{n_{ik}}(a, b) - Gen_{n_{il}}(a, b))$$

$$n_{il} = 0 \text{ to } F-1$$
(1)

式中: $D_{n_{ik}}$ 表示任意抗体 $Gen_{n_{ik}}$ 与种群中抗体 $Gen_{n_{il}}$ 之间的欧氏距离; E 表示组成抗体的特征片段总数; F 表示当前种群的大小。

基于否定选择算法的随机测试伪代码如下所示。

算法 1 基于否定选择算法的随机测试

输入 $T = \emptyset$ // T 是初始化抗体种群集合,设置抗体种群的大小 M 和欧氏距离阈值 Z

输出 T // 输出最新优化的抗体种群 T

1. build(Gen0);

// 产生第 1 个抗体编码: 随机 Cache 发出随机操作码、

//随机地址映射到 DCU0、随机目录 bank、随机字节访问,配置合法的 NoC 请求报文

```
2. T={Gen0};
3. while (T≠∅) do
4.   F=calculate_sizeof(T); //计算抗体种群的大小 F
5.   build(Gennik); //产生抗体 Gennil 的编码: 随机
Cache 发出随机操作码、随机地址映射到 DCU0, 地址相关或者无关、随机目录 bank、随机字节访问, 配置合法的 NoC 请求报文
6.   for nil=0 to F-1 ;
7.     calculate (Dnik); //计算抗体 Gennik 的欧氏距离
8.   end for
9.   if Dnik<Z, goto step5;
10.  else T=T ∪ {Gennik} ;
11.  if nil<M goto step4 ;
12. end while
13. output T; //输出最新优化的抗体种群 T
```

3.2 一致性协议检查器

Cache 一致性协议异常复杂, 基于否定选择算法的随机测试激励, 并不能完全覆盖协议正确性的验证, 发现和定位设计漏洞非常耗时。为此, 设计了一致性协议监视检查器, 判定目录控制器是否按照 Cache 一致性协议要求进行状态的正确迁移, 同时在一致性相关状态与报文流程正确的基础上, 检查全局的数据一致性是否正确。

一致性协议检查器采用记分板方式工作。在 NoC 中设置探针, 收集 Cache 事务发生的整个生命周期事件, 包括: 监视 Cache 协议事务处理的基本流程; 监视目录控制器是否成功完成目录替换操作和目录内容更新; 监视报文类型及报文各个报文域段的合法取值、Cache 状态的迁移路径、非法

状态; 比对读写数据正确性、处理事务是否超时, 这样即可有效精准地定位错误的 Cache 事务类型和场景。

Cache 一致性协议和片上互连结构在设计实现时是紧密耦合的, NoC 连接了主存、Cache、DCU, 不论采用总线还是网络结构, 多核处理器的互连结构通常都采用报文通信方式, Cache 一致性消息通过片上互连报文进行传输。NoC 的报文结构包括事务号、源节点、目的节点, 操作码、地址或数据。NoC 的报文的操作码定义了 Cache 读写请求、监听请求、读写响应、监听应答。源节点表征具体发出读写请求是来自 64 个 Cache 中的哪个 Cache, 本地芯粒或者跨芯粒的 Cache、哪个 DCU 发出的监听请求。目的节点一般是主存、Cache 或者 DCU。

一致性协议检查器的记分板如表 1 所示, 其中登记事务的时间、执行态是进行或完成, 操作码是读写请求、监听请求或监听应答。当读写响应完成, 记分板表格登记该事务完成, 修改对应的执行状态; 利用事务号、源节点, 目的节点、地址做索引, 将对应的表格的执行态改为完成态。如果记分板某个地址读响应数据不是记分板中最后写入的值, 协议检查器报告错误。程序正常结束或程序执行时间超过预定时间时, 检查记分板是否有未完成事务, 如果该事务因超时没有完成, 则程序报告错误。当程序报告错误, 则检查记分板文件, 记分板表格精确定位了错误场景和发生时间。如果读写主存数据不一致, 即发生数据不一致时, 则需要调试查错。如果程序正常结束, 并且检查记分板文件时事务执行态都是完成态, 则确定 Cache 协议验证通过测试。

表 1 一致性检查记录表结构
Table 1 The recorder structure of the Cache coherence check

表序号	模拟时间	事务号	源节点	目的节点	操作码	访存地址	访存大小/Byte	访存数据	执行态
1	第 60 拍	0	Die0 Cache0	Die0 DCU0	写请求	3 000	1	127	进行
2	第 110 拍	1	Die1 Cache17	Die0 DCU0	读请求	3 000	64	127	进行
.....									

4 实验结果

本文采用仿真方式, 建立了基于芯粒集成的多核处理器目录一致性协议的敏捷验证环境。针对在研的一款处理器芯片, 验证功能点主要是 Cache 一致性报文的正确处理、目录功能的正确性, 目录功能包括目录命中、替换、目录忙命中、目录满命中、目录不命中等场景。

随机激励测试实现了 81% 的功能覆盖率, 利

用定向激励有效解决边界场景覆盖, 最终实现功能覆盖率 100%。采用否定选择算法的随机测试激励与完全随机测试激励的功能覆盖率对比如图 9 所示(彩色效果见《计算机工程》官网 HTML 版, 下同)。随机生成测试激励 10 000 个测试向量, 达到功能覆盖率 53%, 随着测试向量的增加, 覆盖率提升平缓; 采用否定选择算法的随机测试激励, 只需 6 000 个测试向量, 覆盖率结果已收敛, 可以达到功能覆盖率 81%。针对目录满命中、目

录存储器错误擦除操作等边界场景,采用定向测试激励,补足随机测试不能达到的 19% 的功能覆盖率,最终达到 100% 覆盖率。实验结果显示,基于否定选择算法的随机测试在测试激励减少 40% 的条件下,功能覆盖率较未使用否定选择算法的随机测试提高了 28 百分点。

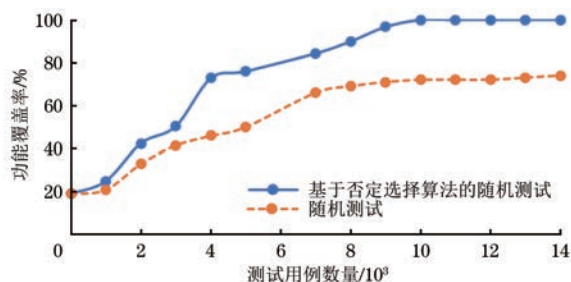


图 9 随机测试激励执行的功能覆盖率

Fig.9 The functional coverage status of random test

定向测试和随机测试的验证过程中累计发现 148 个设计错误,错误分布率如图 10 所示,其中,跨芯粒事务处理错误占比为 44%,目录与 NoC 接口、访存接口错误占比为 11%,目录替换 Buffer 错误占比为 8%,报文 Fifo 错误占比为 13%,目录流水线处理错误占比为 24%,跨芯粒事务处理的设计错误率最高。

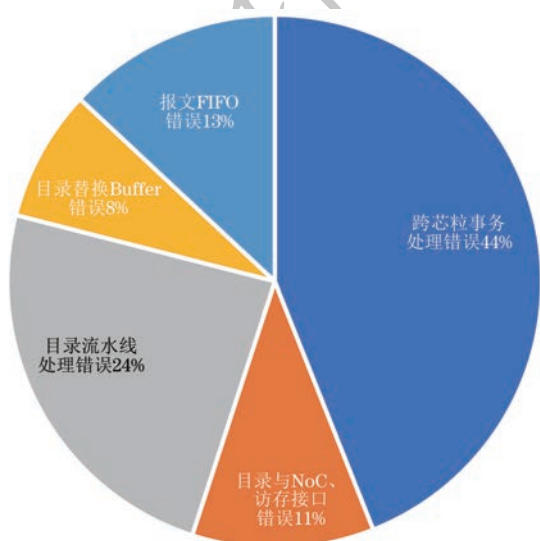


图 10 设计错误分布率

Fig.10 Design bugs rate

综上,在多芯粒集成的处理器的目录控制器验证过程中,随机激励测试实现了 81% 的功能覆盖率,利用定向激励有效解决了边界场景覆盖,最终实现功能覆盖率 100%。同时,利用 Cache 一致性检查器快速定位错误,该工具定位的错误占总漏洞数的 90%,有效提高了查错效率。

5 结束语

多核处理器的 Cache 一致性验证历来是处理器

系统研制的瓶颈。多芯粒集成的微处理器中,Cache 一致性验证更是一个具有重要应用价值的技术问题。本文研究了面向芯粒集成的多核处理器目录控制器的敏捷验证技术,采用否定选择算法优化随机测试,并设计了 Cache 一致性检查工具,监视一致性协议事务生命周期。实验结果表明,这些敏捷验证方法有效提高了验证效率和验证质量。

本文研究了面向芯粒集成的国产高性能微处理器的目录控制器敏捷验证方法。下一步将研究所提出的随机测试优化算法和一致性协议检查工具如何适用不同处理器架构,同时随着系统规模的扩大和功能的增加,如何确保可移植性、可维护性和自适应性。此外,今后将对多芯粒 Cache 一致性协议随机测试的自适应生成和仿真效率做进一步的优化研究。

参考文献

- [1] SORIN D J, HILL M D, WOOD D A. A primer on memory consistency and cache coherence [M]. [S. l.]: Morgan & Claypool Publishers, 2011.
- [2] SIMONI R, HOROWITZ M. Modeling the performance of limited pointers directories for cache coherence [C] // Proceedings of the 18th Annual International Symposium on Computer Architecture. New York, USA: ACM Press, 1991: 309-319.
- [3] NASSIF N, MUNCH A O, MOLNAR C L, et al. Sapphire Rapids: the next-generation Intel Xeon scalable processor [C] // Proceedings of the IEEE International Solid-State Circuits Conference. San Francisco, USA: IEEE Press, 2022: 44-46.
- [4] BURD T, BECK N, WHITE S, et al. "Zeppelin": an SoC for multichip architectures [J]. IEEE Journal of Solid-State Circuits, 2019, 54(1): 133-143.
- [5] VELTEN M, SCHÖNE R, ILSCHKE T, et al. Memory performance of AMD EPYC Rome and Intel Cascade Lake SP server processors [C] // Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering. New York, USA: ACM Press, 2022: 165-175.
- [6] SZUSTAK L, WYRZYKOWSKI R, KUCZYNSKI L, et al. Architectural adaptation and performance-energy optimization for CFD application on AMD EPYC Rome [J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(12): 2852-2866.
- [7] SHAO Y S, CLEMONS J, VENKATESAN R, et al. Simba: scaling deep-learning inference with multi-chip-module-based architecture [C] // Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. New York, USA: ACM Press, 2019: 14-27.
- [8] IBM Inc. IBM power systems facts & features enterprise, scale-out and accelerated servers with Power9 processor technology [EB/OL]. [2024-10-06]. <https://www.ibm.com/downloads/cas/EPNDE9D0>.
- [9] Universal Chiplet Interconnect Express. Universal Chiplet Interconnect express (UCIe): building an open chiplet ecosystem [EB/OL]. [2024-10-06]. https://www.uciexpress.org/_files/ugd/0c1418_c5970a68ab214fc97fab16d11581449.pdf.
- [10] 芯粒互联接口标准: T/ZGTXXH096—2024 [S]. 中国 Chiplet 产业联盟, 2024.

- Chiplet Interconnection Interface Standard: T/ZGTXXH096—2024[S]. China Chiplet Industry Alliance, 2024. (in Chinese)
- [11] 集成芯片前沿技术科学技术专家组. 芯片集成与芯粒技术白皮书[EB/OL]. [2024-10-06]. https://www.sgpjbg.com.cn/baogao/142807.html?bd_vid=8203137834455463678. Expert Group on Cutting-Edge Technologies of Integrated Chips. White paper on integrated chip and chiplet technologies[EB/OL]. (2024-10-06). https://www.sgpjbg.com.cn/baogao/142807.html?bd_vid=8203137834455463678. (in Chinese)
- [12] 胡森森, 计卫星, 王一拙, 等. 片上多核处理器 Cache 一致性协议优化研究综述[J]. 软件学报, 2017, 28(4): 1027-1047. HU S S, JI W X, WANG Y Z, et al. Survey on Cache coherence protocol and performance optimization for chip multi-processor[J]. Journal of Software, 2017, 28(4): 1027-1047. (in Chinese)
- [13] QIU Y D, JIAO J, ZENG X Y, et al. Tag-sharer-fusion directory: a scalable coherence directory with flexible entry formats[J]. IEEE Transactions on Parallel and Distributed Systems, 2023, 34(1): 262-274.
- [14] SHU W, TZENG N F. NUDA: non-uniform directory architecture for scalable chip multiprocessors [J]. IEEE Transactions on Computers, 2018, 67(5): 740-747.
- [15] ZEBCHUK J, FALSAFI B, MOSHOVOS A. Multi-grain coherence directories[C]//Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture. Davis, USA: IEEE Press, 2017: 359-370.
- [16] ROS A, ACACIO M E, GARCIA J M. A direct coherence protocol for many-core chip multiprocessors [J]. IEEE Transactions on Parallel and Distributed Systems, 2010, 21(12): 1779-1792.
- [17] SANCHEZ D, KOZYRAKIS C. SCD: a scalable coherence directory with flexible sharer set encoding[C]//Proceedings of the IEEE International Symposium on High-Performance Computer Architecture. New Orleans, USA: IEEE Press, 2012: 1-12.
- [18] FERDMAN M, LOTFI-KAMRAN P, BALET K, et al. Cuckoo directory: a scalable directory for many-core systems[C]//Proceedings of the IEEE 17th International Symposium on High-Performance Computer Architecture. San Antonio, USA: IEEE Press, 2011: 169-180.
- [19] ASADUZZAMAN A, CHIDELLA K K. A novel directory based hybrid cache coherence protocol for shared memory multiprocessors[C]//Proceedings of the IEEE International Symposium on Phased Array Systems and Technology. Waltham, USA: IEEE Press, 2017: 1-6.
- [20] KUMAR B, THAKUR S, BASU K, et al. A low overhead methodology for validating memory consistency models in chip multiprocessors [C] // Proceedings of the 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems. Bangalore, India: IEEE Press, 2020: 101-106.
- [21] KUMAR B, BHOSALE A K, FUJITA M, et al. Validating multi-processor cache coherence mechanisms under diminished observability[C]//Proceedings of the IEEE 28th Asian Test Symposium. Kolkata, India: IEEE Press, 2020: 99.
- [22] LYU Y D, QIN X K, CHEN M S, et al. Directed test generation for validation of cache coherence protocols [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 38(1): 163-176.
- [23] 王云霏, 王飙, 李媛. 多路直连 CMP 一致性协议量化研究[C]//第二十届计算机工程与工艺年会论文集. 西安: 中国计算机学会, 2016: 150-156.
- WANG Y J, WANG B, LI Y. Quantitative research on multi-path direct Connection Consistency Protocol (CMP) [C] // Proceedings of the 20th Annual Conference on Computer Engineering and Technology. Xi'an, China: CCF, 2016: 150-156. (in Chinese)
- [24] MOHI-ALDEEN S M, MOHAMAD R, DERIS S. Application of Negative Selection Algorithm (NSA) for test data generation of path testing[J]. Applied Soft Computing, 2016, 49: 1118-1128.
- [25] 陈文, 李涛, 刘晓洁, 等. 一种基于自体集层次聚类的否定选择算法[J]. 中国科学(信息科学), 2013, 43(5): 611-625. CHEN W, LI T, LIU X J, et al. A negative selection algorithm based on hierarchical clustering of self set [J]. Scientia Sinica (Informationis), 2013, 43(5): 611-625. (in Chinese)
- [26] 夏春艳, 张岩, 万里, 等. 基于否定选择遗传算法的路径覆盖测试数据生成[J]. 电子学报, 2019, 47(12): 2630-2638. XIA C Y, ZHANG Y, WAN L, et al. Test data generation of path coverage based on negative selection genetic algorithm [J]. Acta Electronica Sinica, 2019, 47(12): 2630-2638. (in Chinese)

文字编辑 金胡考
栏目编辑 宋 圆