doi:10.3772/j.issn.1002-0470.2016.08－09.001

基于处理器硅前性能验证平台的基准程序库设计方法①

张华亮②[*,**,***] 刘宏伟[*,***] 刘天义[*,**,***]

([*]中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190)

([**]中国科学院大学 北京 100049)

([***]龙芯中科技术有限公司 北京 100190)

摘 要 提出了一种新的适用于处理器的硅前性能验证平台的基准程序实现方法。方法的主要思想是利用现成的广泛使用的测试程序集合,通过降低工作负载,采用基于基本块的划分、归并方式,将多个基于相同特征点的代码片段作为一个基准检测点,这些抽象的检测点构成了基准程序库。该方法将复杂的处理器内部行为的一致性判断转换为性能的宏观统计分析,充分利用了已有的权威测试基准集,无需重新编写性能验证平台的基准程序,既扩大了验证程序的规模,又节省了大量的劳动,同时可以针对验证样本通过分析系统自动展开验证工作,减少了人工核对的工作量。

关键词 基准测试程序集,硅前性能验证,处理器验证,仿真加速器,基本块,验证平台,基准程序库

0 引 言

近年来,工业4.0时代的到来以及智能设备的推广和广泛应用,市场需求推动处理器设计向着高性能、低功耗、高可靠性、高集成度等方向不断发展,同时也带动了处理器验证技术的飞速进步。在一款处理器的设计研发当中,硅前性能验证的工作决定着设计空间的探索方向和功能单元的性能相关参数的调整范围,同时也决定着对处理器进行性能评估。性能验证工作贯穿处理器设计开发的多个阶段。基准测试程序集(benchmark set)是进行计算机性能验证的重要工具。随着验证技术的不断发展和验证平台性能的大幅度提高,对基准测试程序集也需要做相应的改进。在这种情况下本文围绕着基准测试程序集展开工作,提出了一种新的基准测试程序集设计实现方法。

本文针对目前的处理器硅前验证平台的特点,提出了一种基于程序基本块特征和统计方法的基准程序数据库方案。在该方案下,开发者可以利用现成的已有的测试基准集,通过调整或少量修改输入数据规模,降低测试的工作负载,使得原来大规模的性能验证程序可以在基于硬件加速的仿真器验证平台得以运行,同时采用基于程序特征的基本块划分方式,将大型程序的整体性能分析工作降解为对局部热点程序块的观测。该方法从宏观角度提出性能分析单元,基于相似特性的程序片段组成检测点,进一步构成程序数据库。把验证程序重构成一个新的性能评估基准集合,同时提供了处理器性能的评估方法。本实验基于龙芯3A1500[1]多核处理器的验证平台,但是不局限于MIPS[2]架构和仿真器的验证平台。在验证阶段,通过该基准程序库我们发现并

① "核高基"科技重大专项课题(2014ZX01020201),国家自然科学基金(61432016)和863计划(2013AA014301)资助项目。

② 男,1982年生,博士生;研究方向:高性能计算机体系结构;联系人,E-mail:zhanghualiang@ict.ac.cn

(收稿日期:2016-04-06)

修正了十多个处理器的性能及功能缺陷。

1 相关工作

性能验证基准程序集可用于验证处理器的设计性能指标,它是面向验证平台,针对处理器具体设计结构、功能单元、策略算法的一组指令单元或完整程序。

SPEC2000[3],SPEC CPU2006[4] 等目前流行的标准化性能测试工具,可以对处理器进行全面、系统的测试和评估。该工具适合处理器的硅后性能测试阶段,但是由于目前的性能验证平台和精确模拟器的性能限制(仿真主频低、虚拟内存偏小等原因),导致其运行时间过长,在工程实践中无法作为性能验证程序直接使用。

为了解决在软件仿真平台上快速验证处理器的性能指标及分析处理器的内部流水线及功能部件,Alpha 公司提出了 microbench 的基准测试集,马可[5]在此基础上扩展了该测试集,包含了定点部件、浮点部件、访存行为、cache 操作、分支预测等验证模块。这个基准测试集可以从微观角度分析处理器的多项性能指标。此外 unixbench[6],lmbench[7] 等也是比较经典的测试基准集。这些程序主要针对于单一的结构单元或策略算法。测试流程简单,行为模式单一,与流片后的系统性能评价的 SPEC2000 等程序的复杂行为的工作负载差异较大。所以该类测试程序无法满足系统评估处理器的功能要求。

CoreMark[8] 测试程序模拟了队列操作、排序算法、矩阵运算等操作,是一个很符合实际应用情景的测试程序。每次测试循环时间较短,简单修改后它就可以运行在性能验证平台上评估处理器的性能。但是此类行为复杂,工作负载适中的程序数量偏少,无法满足验证平台上基准测试程序的规模及数量上的需求。如果以软件工程的方式大规模开发此类程序需要大量的时间及人力资源消耗,增加处理器的研发及验证成本。

随机指令生成方式是一种经常采用的验证程序生成方式,该方式广泛应用于功能验证的流程中,也可以应用在硅前性能验证的简单测试模式中。对应中大规模的测试程序,IBM 公司在 POWER7[9] 的验证过程中采用时钟精确的软件模拟器和基于硬件加速的仿真器 AWAN 分别执行 SPEC 测试程序的片段,通过性能分析工具调试性能缺陷。本文方案与该方法相比无需时钟级精确的性能模拟器,文中方法更具普适性,考虑到实际中不同模型的实现细节及程序的运行环境,本文方案中不要求对标平台的执行指令流完全一致,所以该方法在实际使用中更加灵活。

2 方法的可行性分析

在龙芯 3A1500 处理器的硅前性能验证流程中,我们希望用于验证平台的程序基准集满足以下几个特点:(1) 基准测试与真实的系统应用程序的行为模式尽量一致;(2) 基准程序无需大规模的软件工程方式实现,尽量利用现有的成熟的测试程序;(3) 基准程序数量较大,满足大规模测试的要求;(4) 基准程序的工作负载构成具备多样性;(5) 基准程序与通用的硅后系统测试程序的热点片段功能相近,访存模式相似,以便于从硅前性能验证的结果推测流片后的系统性能指标;(6)基准程序规模适中,方便人工排查,调试处理器相关模块的性能异常及缺陷。

处理器的硅前性能验证环境通常由测试基准集、形式验证程序、软件模拟器、FPGA 验证平台,算法策略抽象模型等模块组成。验证方法有模拟验证、FPGA 验证、形式化验证、基于仿真器的全系统验证等方式。

本实验采取了基于仿真器与软件模拟器相结合的方式验证。采用基于处理器结构抽象的软件模拟器与实现处理器详细设计方案的仿真器为对标平台。在精简版的操作系统上共同运行相同的测试程序,采用相同的工作负载,通过运行结果的性能参数对比判断处理器实现方式的正确性。考虑到具体的测试用例:以 SPEC2000 的定点测试程序中,如果我们采取 164. gzip 程序为基准程序, test 集合作为输入数据,那么在仿真器平台上等待该程序完成并获取性能参数(运行时间,IPC, 分支预测率,cache 失

效率,TLB 失效率等)需要大约2 个小时。在这种情况下,如果以单个程序作为性能观测的时间窗口长度,在程序完全运行结束后,如果仿真器的性能结果与模拟器的数据对比差异较大时,通过人工核查的办法进行数据分析,那么对应的指令流数量多达 2G 多条,在实际工程中几乎不可行。同时在该时间段内只完成了一个测试用例,测试效率低下。

为了实现人工核查的程序段在一个可控的范围内,我们将一个完整的测试程序的执行从逻辑上划分成多个基本块向量。以一个向量块样本为性能验证的基本单元,从而缩短了观察的指令长度,降低了后期人工指令分析的复杂度。由于基准测试的多个样本由相同程序的持续的指令流构成,相比多个由测试程序片段实现的检测点模式,该方案减少了大部分的程序加载和处理器功能单元预热的步骤,提高了验证过程的效率。该方法同时也增加了测试用例的数量,使得性能验证过程中的测试用例更多,更利于发现处理器的结构设计和逻辑实现过程中的性能缺陷。

在实际的验证过程中,将同一验证程序分别运行在基于硬件加速的仿真加速平台上和全系统软件模拟器上,采用基本块向量划分的方法分别采集两个平台的处理器性能数据,包括 IPC,分支预测率,指令分类信息,cache 命中率,TLB 缺失信息,各级流水线的相关信息等。

在验证平台的基本块信息的分析比较过程中,我们发现上述两个平台的分析结果并非完全一致,无法简单地进行基本块向量信息的一一对应及性能指标原始数值的比较。原因如下:(1) 由于处理器设计的策略算法的原因,资源分配、指令调度、cache 替换等算法会采用随机策略,导致程序每次运行的处理器状态不同,进而影响程序的性能参数。(2) 在全系统模拟的环境中,两个系统的硬件时钟中断发生时间通常不一致,导致测试程序的基本块向量信息受到操作系统中断处理操作的扰动,不完全相同。(3) 在系统库函数 glibc 对程序的堆栈数据进行分配时,为了防止恶意软件对程序进行缓冲区溢出漏洞的攻击,及黑客软件对程序数据进行破解,每次程序运行时的堆栈地址的空间布局是随机分配的,程序数据分配到了不同的地址可能影响到 cache 及访存行为。这些问题在微观的简单指令流验证过程中是可以忽略的,因为微指令集的每个测试功能单一,行为模式固定,无需操作系统,环境简单。但是在宏观的基准程序测试环境下,上述的多种因素导致参与性能对比的指令流及系统性能指标不完全一致,需要进一步处理。

我们采用宏观的基于大数据的统计规律,将之前对基准测试程序的实验数据确定性判断,改进为基于相似行为程序基本块集合的统计分析。这样可以减少系统及测试环境对验证程序结果的影响,降低性能分析的工程复杂度。同时该方法对验证程序的执行透明,无需对测试基准程序进行人工剪裁或功能改写等工作。

3 基准程序库的实现方案

为了满足验证环境的需求,我们在软件模拟器验证环境和仿真加速平台上实现性能检验的功能,软件模拟器采用基于虚拟计数器的统计模块记录程序信息。仿真加速器采用 DPI 编程接口统计处理器内部状态信息。

在验证程序的执行过程中,我们把一个程序的执行路径描述为指令基本块的顺序执行。一个基本块定义为一段只有一个入口和一个出口连续的指令序列。实验中记录基本块的指令长度与入口地址。截取 5M 长度的指令流作为一个测试样本。

性能文件中每个样本的运行快照 Profile 由两部分组成,一部分是指令相关数据:统计各个基本块在本段指令流中的迭代次数,该数据与基本块指令的长度乘积作为一个新元素,将指令流中所有的基本块统计出来,这些元素按数值大小排序后组成该样本的指令相关列表 L。由于该列表元素过多,为了后继数据处理速度的提高,尽量在减少维度的同时,保持样本间的数据关系,我们采用随机投射(random project algorithm)算法,将指令列表的数据维度减少到 30,记录更新后的指令向量为 L_i。样本的另一部分数据为该段指令的性能参数,这些数据包括 IPC、分支预测率、数据 cache 访问的失效率、

TLB 缺失次数、流水线吞吐量、流水线队列的堵塞次数等。这些性能数据组成一个向量记为 L_s。至此我们将验证程序的运行情况抽象地表示为两个数值列表。

我们将软件模拟器平台得到的两个列表作为输入数据。在性能分析模块中,算法首先对同一测试程序中所有样本的指令相关向量使用 Ward 系统聚类法。Ward 法又称利差平方和法,是一种较好的层次聚类方法。该方法采用欧氏距离作为样本的距离度量,算法的核心思想是使得同类的样本离差平方和较小,同时类与类之间的偏差平方和较大。在该算法下,一个类中样本的离差平方和的计算公式如下:

$$S_t = \sum_{i=1}^{n} (L_i^t - \bar{L}^t)'(L_i^t - \bar{L}^t) \quad (1)$$

其中, S_t 为一个类的离差平方和, $\bar{L}^t$ 为该类中的向量的平均值。以 SPEC CPU2000 中的 164. gzip 程序为例,通过该方法我们将程序运行样本分为 200 个类,每个类由相关的程序特性向量信息与处理器性能数据组成。我们把每个类作为一个“宏观”上的基准检测点,类中的每个样本的执行可看作验证系统对该检测点的一次访问。所有测试程序经过模拟器平台的运行及聚类处理后得到基准检测点,这些检测点组成基准程序库。计算每个类的中心点,为后继的归类步骤提供数据。

对于加速仿真平台得到的测试样本,进行基于指令特征向量 L_i 的归类。计算每个样本与聚类中心的欧氏距离,将该样本归类到距离最近的类中。

所有的样本归类完毕后,计算两个对标平台同类样本的性能平均值,按照以下顺序重点分析验证程序:(1) 对标同类 IPC 平均值相差 6% 以上的;(2) 加速仿真平台上性能向量中每项指标最差的类;(3) 该验证程序中包含样本数量最大的 3 个类,即验证程序的局部性热点。选择上述三者离距离中心点最近的 2 个片段并记录对应基本块向量的仿真起始时间。

在仿真加速平台上重新运行性能测试程序。在之前记录的仿真时间的节点处收集处理器的内部模块的信号时序信息,对照处理器的详细设计方案分析波形文件的信号时序关系。对处理器结构进行修改、调整或直接展开新一轮的性能验证工作。性能验证的方法流程如图 1 所示。

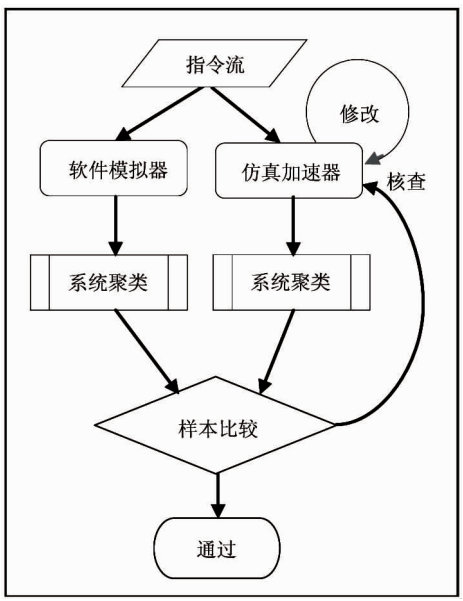

图 1 性能验证的方法流程图

4 实验平台和实验结果

本文所提出的基准程序库及验证方法在龙芯 3A1500 的研发过程中得到了实现和使用。基准程序库中的测试程序采用了目前广泛使用的 SPEC CPU2000,SPEC CPU2006 等中等规模的测试基准集中的程序。

本实验中的验证平台包括 Sim-Godson 软件模拟器、EVE 仿真加速器[10]平台、Verdi[11]的波形调试分析工具。受限于实验平台的物理条件及验证时间的制约,输入的数据集合大多采用 test 集合。下面分别介绍各个实验平台和其实验结果。

龙芯 3A1500 处理器是龙芯公司基于 464E[12]的 IP 核产品,它包括 4 个处理器核心,是一款支持超标量、动态流水线、多发射,乱序执行的高性能通用处理器,功能部件拥有两个定点运算单元、两个浮点运算单元以及两个访存功能部件。本实验的处理器结构模型参数详见表 1。

表 1　3A1500 处理器的基准参考模型

结构模块	主要参数
计算单元	定点部件 2 个,浮点部件 2 个
存储单元	访存部件 2 个
L1 Dcache	64K,4 路组相连,块大小 64Byte
L1 Icache	64K,4 路组相连,块大小 64Byte
TLB	一级 32 项,二级 64 项 CAM, 1024 项 RAM
分支单元	组合分支历史表(BHTs),128 项 BrBTB, 16 项 RAS, 1024 项 JBTAC
队列单元	16 项定点队列,24 项浮点队列,32 项访存队列
发射单元	4 发射,128 项 ROB

Sim-Godson 是面向龙芯 GS464E IP 核开发的一款性能分析模拟器。该模拟器采用功能模拟和时序模拟功能分离的执行驱动方式。执行引擎负责解释执行指令,性能分析模块只实现分析指令在流水线中流动所需的微体系结构,不关心指令执行的数据结果。这使得模拟器的运行速度得到提升。

EVE 仿真加速器是 Synopsis 公司推出的一款基于 FPGA 的系统仿真验证平台,可以创建时钟精确的处理器结构功能模型,提供完整信号可视性的调试功能。验证过程中采用的操作系统为精简版的 Linux,内核版本号为 2.6.36。在该平台上采用处理器模型,虚拟器件,虚拟速度适配器和预编译的存储器模型搭建事务级的验证环境。

图 2 显示为每个检测点样本的记录信息。(a)图片对应基本块向量的宏观性能统计数据,(b)图片对应该段程序的各级流水信息。同时记录的还有基本块向量信息。在结束了随机指令验证以及基于微基准程序验证流程后,处理器的硅前性能验证结果如图 3 所示,其中发现了软件模拟器 Bug 4 处,处理器实现功能错误 2 处,处理器性能缺陷 4 个(包括算法的实现错误),功能模块参数调整 3 个,以及发现编译器优化方案 2 个(地址连续的 load 指令合并以及指令顺序调整),glib 函数库优化多个(与预取相关的字符串操作指令等)。

(a) 处理器的综合性能　　(b) 处理器的流水线性能

图 2　基于检测点的一个样本对应的处理器的性能数据

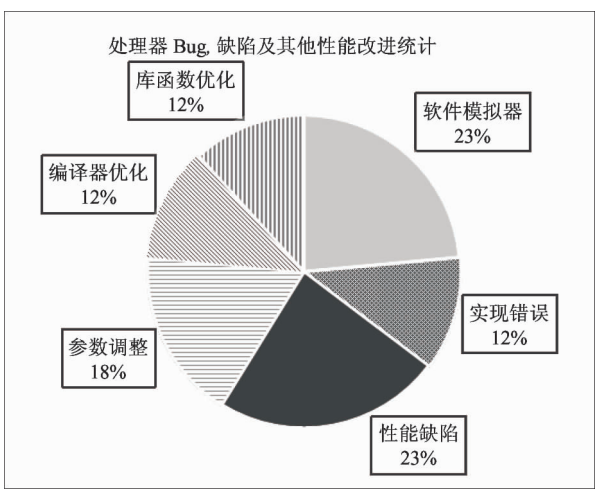

图 3　基于基准程序库的性能验证统计结果

5　结　论

基准测试集是处理器进行验证工作的重要工具。目前业界有多种方法实现硅前性能验证工作的基准测试集,但是通用的基准程序与验证工作的规模不匹配,随机生成的验证向量与实际硅后的性能测试程序相差较大。本文针对硅前验证平台的特点,提出了一种面向硅前性能验证的基准程序库,该方法使用已有的成熟的基准测试程序,不需要人工编写新的基准程序,减少了硅前验证的工作量。通过国产处理器龙芯 3A1500 的硅前性能验证工作,

该基准测试库可以很好地发现处理器的性能 Bug 及缺陷,该方法同时具有良好的可扩展性。因此,本文提出的基准测试程序库是一种有效的硅前性能验证的工具。

参考文献

[1] Wu R L, Wang W X, Wang H D, et al. Design of loongson gs464e processor architecture. *Scientia Sinica Informationis*, 2015, 45 (4): 480-500

[2] MIPS Technologies Inc. MIPS64 Architecture for Programmers Volume I: Introduction to the MIPS64 Architecture, 2005

[3] Henning J L. SPEC CPU2000: Measuring CPU performance in the new millennium. *Computer*, 2000, 33(7): 28-35

[4] Li S, Cheng B, Gao X, et al. Performance characterization of SPEC CPU2006 benchmarks on Intel and AMD platform. In: Proceedings of the International Workshop on Education Technology & Computer Science, Wuhan, Hubei, 2009. 116-121

[5] 马可. 微处理器性能分析模型的建立和研究:[博士学位论文]. 合肥:中国科技大学计算机科学技术系, 2007. 12-19

[6] Shouvik B, Daniel A M. Predicting the effect of memory contention in multi-core computers using analytic performance models. *IEEE Transactions on Computers*, 2015, 64 (8): 2279-2292

[7] Kruger C P, Hancke G P. Benchmarking Internet of things devices. In: Proceedings of the 2014 IEEE International Conference on Industrial Informatics, Porto Alegre, Brazil, 2014. 611-616

[8] Poovey J A, Conte T M, Levy M, et al. A benchmark characterization of the EEMBC benchmark suite. *IEEE Micro*, 2009, 29(5): 18-29

[9] Srinivas M, Sinharoy B, Eickemeyer R J, et al. IBM POWER7 performance modeling, verification, and evaluation. *IBM Journal of Research & Development*, 2011, 55 (3): 4:1-4:19

[10] Synopsys. EVE. http://www.eve-team.com: Synopsys, 2010

[11] Synopsys. Verdi. https://www.synopsys.com: Synopsys, 2011

[12] Gao X, Chen Y J, Wang H D, et al. System architecture of Godson-3 multi-core processors. *Journal of Computer Science and Technology*, 2010, 25(2): 181-191

Design of processors' benchmark database using the testbench for pre-silicon performance verification

Zhang Hualiang[*,**,***], Liu Hongwei[*,***], Liu Tianyi[*,**,***]

([*] Key Laboratory of Computer Architecture, Institute of Cumputing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100049)

([***] Loongson Technology Corporation Limited, Beijing 100190)

Abstract

A method to construct a middle scale testbench for processors' performance debug during the pre-silicon verification stage was proposed. This approach makes use of the prevalent processor benchmarks and replaces the complete test with light workloads. The instruction streams are clustered into several checkpoints according to basic block partition and categorization, and all of these checkpoints make up a novel benchmark. This approach converts the complex performance diagnostic method, which is based on waveform observation of the elaborate architecture, to statistical analysis of checkpoints. This method does not require a new medium scale testbench, so extra work and cost can be avoided. Moreover this can prompt automated verification process with less manual inspection.

Key words: benchmarks, pre-silicon performance verification, processor verification, simulation accelerator, basic block, testbench, benchmark database