# 龙芯 1 号微处理机性能模拟器

张仕健[1,2],张福新[1,2],唐志敏[2]

[1](中国科学院 计算技术研究所 系统结构研究室,北京 100080)
[2](中国科学院 研究生院,北京 100039)
E-mail:zsj_bj@ict.ac.cn

**摘 要**:性能模拟器是现代微处理器结构设计过程中性能评估的重要工具.它要求灵活性好、运行速度快和准确度高,然而,实现这样一个模拟器除了工作量大之外,还需要相当的设计技巧.通过改造 SimpleScalar 的 sim-outorder,开发了一个针对龙芯 1 号微处理器结构的性能模拟器,既减小了开发的工作量,又实现了灵活性、速度及准确度三者之间的平衡.实验数据表明,该性能模拟器平均运行速度在 200 KIPS 以上,IPC 平均偏差在 10%以内.

**关 键 词**:性能模拟器;性能评估;微处理器结构
**中图分类号**:TP302 **文献标识码**:A **文 章 编 号**:1000-1220(2006)12-2317-04

## Performance Simulator for Godson1 Microprocessor

ZHANG Shi-jian,ZHANG Fu-xin,TANG Zhi-min

[1](*Computer Architecture Laboratory*,*Institute of Computing Technology*,*Chinese Academy of Sciences*,*Beijing* 100080,*China*)
[2](*Graduate School of Chinese Academy of Sciences*,*Beijing* 100039,*China*)

**Abstract**:A performance simulator is an important tool to evaluate performance during the microprocessor architecture design. It aims to be flexible, fast and accurate. However, completing such a performance simulator is a hard work, because it is not only a heavy task, but also needs wise schemes. A performance simulator for Godson1 microprocessor has been developed based on the sim-outorder simulator in the SimpleScalar toolsets. In this way, the heavy task is cut down, while flexibility, speed and accuracy can be balanced. It is shown by the experiment that the performance simulator can run at 200 KIPS with IPC deviation less than 10%.

**Key words**:performance simulator;performance evaluation;microprocessor architecture

## 1 引 言

随着微处理器性能的提高,微处理器的结构也变得越来越复杂.当代高性能微处理器一般采用了多发射、乱序执行、猜测执行、分支预测和 cache 等微结构.这些复杂的微结构,使得设计者在设计一款高性能微处理器时需要进行仔细地权衡,不可能再仅仅凭借经验、直觉了[1].性能模拟器可帮助设计者方便地评估各种设计方案,快速选定合理的微结构.在现代微处理器结构设计过程中,性能模拟器的地位显得越来越重要.一般来说,一个性能模拟器需要对三个方面进行折衷:灵活性、速度及准确度.高度的灵活性可便于对不同的微结构进行评估.模拟器速度快,可以执行更大规模的负载,在更短的时间内给出评估结果.准确度高保证了评估结果和实际情况不会有太大的偏差.这三个方面相互之间通常是矛盾的.准确度越高,需要的约束就越多,速度随之降低,灵活性也会下降;灵活性好,需要支持不同的配置,速度却会变慢[2].

龙芯 1 号信号级模拟器(Godson1_ICT)是龙芯 1 号微处理器的详细模拟.它准确度高,验证了设计的正确性,但是它灵活性差,运行速度也很慢,平均只有 1k IPS 左右,不适合评估微结构.sim-outorder 是 SimpleScalar 工具集中的详细模拟器.它灵活性好,可用命令选项或文本方式配置结构参数,运行速度也很快,平均为 300k IPS[8].但是,由于 sim-outorder 模拟的微处理器结构和实际微处理结构可能相差很大,因此它的准确度不高,平均 IPC 偏差可达 40%[7].本文通过改造 SimpleScalar 的 sim-outorder,开发了一个针对龙芯 1 号微处理器结构的性能模拟器(Godson1_sim).它既具备 SimpleScalar 良好的灵活性,同时 IPC 平均偏差小于 10%,平均运行速度大于 200k IPS.

## 2 模拟器相关研究

模拟器的相关研究主要体现在模拟器的建模技术、加速技术及验证技术三个方面.模拟器有三种建模方式:基于 trace 的、基于执行的及全系统的模拟.基于 trace 的方案需要先执行原测试程序得到程序踪迹,然后模拟器根据这些踪迹模拟指令时序.由于模拟器输入的是已经执行的指令踪迹,对

收稿日期:2005-09-22 收修改稿日期:2005-11-16 基金项目:中国科学院知识创新工程项目高性能通用 CPU 芯片研制项目(KGCX2-109)资助;中国科学院知识创新重大项目高性能通用 CPU 芯片研制项目(KGCX1-SW-09)资助. 作者简介:张仕健,男,1975 年生,博士研究生,研究方向为微处理器体系结构与模拟仿真;张福新,男,1976 年生,博士,研究方向为微处理器体系结构与性能分析;唐志敏,男,1966 年生,研究员,博士生导师,研究方向为高性能计算机系统结构、MPP 系统和数字信号处理.

于猜测执行的指令，基于 trace 的方案很难处理. Matt Reilly 等人对这一问题进行了研究，并试图采用一个仿真器 Aint 处理猜测执行的指令，但是他们的模拟器速度只有 20kIPS 左右，有关准确度的具体数据却没能给出[2]. Mayan Moudgill 等人为 PowerPC 建立的性能模拟器也是基于 trace 的，他们通过控制 Aria 生成 taken 和 not taken 两种版本的踪迹支持指令猜测执行，运行速度 100k IPS 左右，CPI 偏差在 10%以内[3]. 基于执行的模拟器是一边执行指令产生处理机状态，一边生成指令的时序. 它没有 trace 信息的存储问题，也容易支持指令的猜测执行，近年来比较流行. Todd Austin 等人构建的 SimpleScalar 是一个典型的基于执行的建模平台. 它支持多种参数配置，具有良好的灵活性，运行速度也很快，sim-outorder 运行速度为 300k IPS[4]，但是它模拟的处理器结构过于简单，和实际微结构有较大的差别. 基于 trace 的模拟和基于执行的模拟，输入的都是用户态程序，没有考虑操作系统、DMA 及 I/O 外设对系统性能的影响，为了进一步提高准确度就需要全系统的模拟了. SimOS 是一个常用的全系统模拟器，它对系统中各部件实现了不同层次的抽象，抽象层次越低，精确度越高，速度越慢，适合于操作系统、大型数据库等大负载情况下系统性能的研究[5]. 加速技术主要有以 SMARTS 为代表的统计采样技术和以 SimPoint 为代表的程序块特征化技术. SMARTS 将一个测试程序分成多个相同长度的指令段，每个指令段又分为功能模拟指令区、详细模拟预热指令区和详细模拟采样指令区. 在 99.7%的可信域内，SMARTS 的执行速度大约是详细模拟的 30 倍[6]. SimPoint 的方案是先分析程序基本块，选取有代表性的基本块，每个选取的基本块代表了多个具有相同行为的块实例. 这样，详细测量每个有代表性的基本块，计算每个块的实例个数，就可以统计出整个程序的行为. SimPoint 的模拟速度是 SMART 的 1.8 倍，但是它比 SMART 有更大的 CPI 偏差[6]. 模拟器的验证是为了提高模拟器准确度. Bryan Black 等人对 PowPC 604 进行了验证，分析了误差的来源，得到了 4%的平均时钟偏差，但是他们采用的是小规模的常驻 Cache 的测试程序[3]. Desikan 等人对一个 Alpha 21264 的模拟器进行了验证，将运行 SPEC2000 测试程序的 IPC 平均偏差从 40%减少到了 20%[7].

## 3 龙芯 1 号性能模拟器

龙芯 1 号性能模拟器需要平衡灵活性、速度及准确度三个目标. 灵活性可以借鉴 SimpleScalar 良好的用户接口和多样的配置参数；速度上可采用一些设计技巧，减少程序的时空开销；为了提高准确度，需要修改 SimpleScalar，模拟龙芯 1 号微处理器结构，并对修改后的模拟器进行详细的验证.

### 3.1 对 SimpleScalar 的改造

sim-outorder 是 SimpleScalar 中最复杂、最详细的模拟器. 它将指令重命名到 RUU 队列上模拟一个五级动态流水线. 它的流水线结构，如图 1 所示，由取指、分派、发射执行、写回和提交五个阶段组成. 取指阶段从 I-cache 中取指令，放入 dispatch 队列，并根据取指预测器预测下一条指令的 PC. 在取指过程中，如果 I-cache 不命中，则阻塞流水线直到 I-cache miss 完成. 在分派阶段，指令出 dispatch 队列，进入 RUU 队列或 load/store 队列，进行指令译码和寄存器重命名. 在该阶段如果发现取指阶段分支预测错误，则设置猜测执行模式；如果发现指令的操作数都准备好，则指令进入 ready queue 队列. 发射执行阶段跟踪寄存器和内存地址的相关性，从 ready queue 队列中取指令发射到相应的功能部件中执行，并进入写回事件队列. 写回阶段从写回事件队列中取出执行完的指令，并通知所有等待它运算结果的指令. 在该阶段如果发现写回的指令是在猜测模式下，则取消流水线中所有正在执行的指令，并通知取指模块从正确的分支路径上取指. 提交阶段将 RUU 队列中已写回的结果顺序提交给结构寄存器，回收指令占用的 RUU 或 load/store 队列[8].

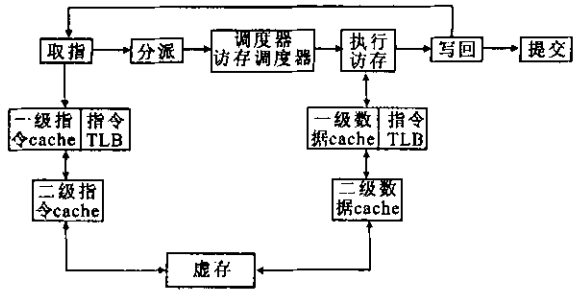


图 1 sim-outorder 流水结构

为了保证内存状态的一致性，sim-outorder 对访存指令做了一些特殊的处理. 在分派阶段，load/store 指令需进入 load/store 队列. 在 load/store 队列中，只有当前面所有的 store 指令地址都已知时，load 指令才能发射执行. 如果 load 的访存地址和前面的 store 地址相等，则 load 指令直接读取该队列中 store 指令的值返回，否则，load 指令访问存储系统返回[8]. 在提交阶段，store 指令的值写入数据 cache.

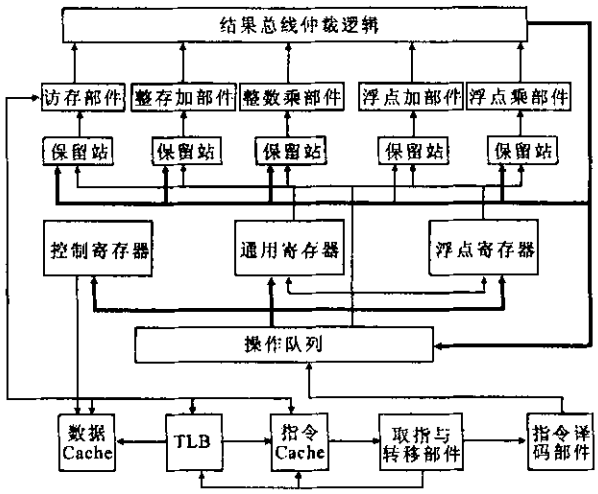


图 2 龙芯 1 号微处理器结构

龙芯 1 号微处理器的流水线是基于操作队列复用的单发射五级动态流水线，如图 2 所示，它由取指、译码及重命名、发

射、执行写回及提交五个阶段组成[9]. 在取指阶段,取指部件从 I-cache 中取回一条指令,送入 IR 寄存器,并采用静态 taken 预测下一条指令的 PC. 取指时,如果 I-cache 不命中,则阻塞流水线直到 I-cache miss 完成. 在译码及重命名阶段,译码部件将 IR 中的指令译码成内部统一格式,将目标寄存器重命名到操作队列上,并进入操作队列. 发射阶段先从操作队列中取一个操作并读取相关寄存器的内容,然后进入相应的保留站中等待. 执行写回阶段先从保留站中取出源操作数都准备好的指令,再进入相应的功能部件中执行,并将执行结果通过结果总线写回操作队列和保留站. 提交阶段将操作队列中已写回的结果提交给结构寄存器,如果提交的指令发生了例外,则进行例外处理.

表 1　模拟器验证前后 IPC 偏差对比表(%)

| | gcc | parser | Perl | vortex | Ammp | art | sixtrack | mean |
|---|---|---|---|---|---|---|---|---|
| non-validated Godson1_sim | 24.0 | 10.6 | 34.2 | 5.6 | -4.6 | 20.4 | -6.2 | 15.1 |
| validated Godson1_sim | 2.5 | 6.6 | 6.1 | 2.4 | -5.4 | 16.3 | -16.1 | 7.9 |

表 2　实验环境

| | |
|---|---|
| CPU | Intel Pentium 4 CPU 2.80GHZ |
| Cache | 512K |
| Memory | 2G |
| Input | SPEC CPU2000 benchmark, test 规模,最多运行 5 亿个时钟周期 |
| Compiler | gcc version 3.3 ,O2 优化 |

通过以上比较,可以发现龙芯 1 号微处理器的流水线结构和 Sim-outorder 的流水线结构在取指、译码及重命名、写回、提交四个阶段比较相似. 这样,龙芯1号性能模拟器可以参考 sim-outorder 中已经定义的 RUU 队列、重命名机制、事件队列及 cache 结构等资源,减少了开发的工作量.

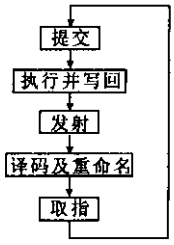

图 3　流水线模拟流程图

对 sim-outorder 的改造主要有以下几个方面:

· 定义龙芯 1 号的功能部件及每个部件的发射延时、操作延时.

· 在每个功能部件中增加保留站,指令从 RUU 队列发射到相应的保留站中等待,取消了 sim-outorder 中的 ready queue 队列.

· 在提交阶段检查分支指令是否猜测错误,如果猜测错误,则清空流水线中所有正在执行的指令,再从正确的路径上重新取指.

· 实现对存储系统的 blocking 访问,访存指令只有在前面的访存操作返回后才进行,访存指令和其他指令一样统一放在 RUU 队列取消了 sim-outorder 中的 load/store 队列. 只有当 RUU 队列中 store 指令是第一条指令时,store 指令才发射执行,并将结果写入数据 cache.

· 支持的龙芯 1 号 branch 指令后面的延迟槽.

### 3.2　灵活性

龙芯 1 号性能模拟器充分借鉴了 SimpleScalar 参数可配置的特点. 用户通过一个配置文件可以很方便地修改微处理器各部件的属性. 例如:可以配置各功能部件的数量、功能单元及保留站大小,也可以配置不同的取指预测机制,用户还可以通过条件编译实现不同流水结构和访存方式. 这样,就能够方便地探索不同的微处理器结构对性能的影响.

表 3　处理器模型参数配置

| | |
|---|---|
| Decode/issue/commit width | 1/1/1 |
| Fetch queue size | 1 |
| RUU size | 32 |
| Function Units | 1 Int ALU,1 Int Mult/Div,1 FP ALU, 1 FP Mult/Div,1 Load/Store Port |
| Reserve Station items forevery Function Unit | 2 |
| Branch Predictor | taken 静态预测器 |
| L1 ICache (hit latency) | 128 sets, 32 bytes blocksize, 2 associations,Random(4 cycles) |
| L1 DCache (hit latency) | 128 sets, 32 bytes blocksize, 2 associations,Random(1 cycle) |
| TLB(miss latency) | 48 sets,4k bytes blocksize,1 association , Random (1 cycle) |
| DRAM latency | first chunk 62 cycles,inter-chunk 3 cycles |

### 3.3　运行速度

龙芯 1 号性能模拟器采用了很多程序设计技巧,减少程序的时间复杂度和空间开销,来提高运行速度. (1)在 TLB 和 Cache 的设计中,使用软件缓存和 hash 查找,降低查找的时间复杂度. (2)采用反向调用流水级的方式模拟流水线的执行过程,减少数据的复制开销. 龙芯 1 号信号级模拟器采用 two-list 算法来模拟流水线,每个寄存器使用两个变量:reg 和 reg.bak. 在一个时钟周期,模拟器先从 reg 变量中取值进行运算,再将运算结果写入 reg.bak 变量,在时钟结束时,又将 reg.bak 的值复制回 reg,这样数据复制的开销很大. 实际上,当流水线工作时,在每个时钟的上升沿,本级流水从上级流水的输出锁存器中得到数据进行运算,在时钟结束前,本级流水的结果输出到输出锁存器中. 在一个时钟周期, 数据在流水线中只前进一个流水级. 采用反向调用流水级的方式,可以简单有效地模拟数据在流水线中的传输过程. 图 3 是流水线模拟的流程图,它可以保证本级流水的数据在被写入之前已被下级流水读取,大大降低了数据复制带来的开销. (3)减少内存的动态分配. 系统初始化时就分配足够的内存,并保存在一个链表表示的资源池中,系统运行时再从资源池中分配和回收. 保留站和事件队列充分使用了这一方案.

采用这些设计技巧后,龙芯 1 号性能模拟器在 Intel Pentium 4 CPU 2.8GHZ 机器上运行时,平均速度在 200k IPS 以上,采用 SMART 加速技术后平均速度可达 1M IPS.

### 3.4 准确度

为了提高准确度,性能模拟器以信号级模拟器为参考进行了对比、验证.首先,编写一些反映微结构特征的测试程序,然后比较这些程序在性能模拟器和信号级模拟器中执行的流水情况,如果差异比较大,分析并找出差异的原因,修改性能模拟器,再进行比较,直到它们之间的差异减少到可以接受的程度.在验证的过程中,造成差异的原因很多是由于规范错误造成的,即对模型的功能理解不正确,模拟的行为是错误的[3].

表 4 模拟器 IPC 比较

| | Gzip | mcf | gcc | parser | Perl | vortex | apsi | Ammp | Art | equake | sixtrack | mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Godson1_ICT | 0.3841 | 0.1880 | 0.1109 | 0.3790 | 0.1064 | 0.2730 | 0.1626 | 0.2329 | 0.0921 | 0.1179 | 0.0955 | 0.1948 |
| Sim-outorder | 0.4312 | 0.3347 | 0.1461 | 0.4272 | 0.1495 | 0.2903 | 0.3215 | 0.2139 | 0.3644 | 0.2154 | 0.2960 | 0.2900 |
| %difference | 12.3 | 78.0 | 31.7 | 12.7 | 40.5 | 6.3 | 97.7 | -8.2 | 295.7 | 82.7 | 209.4 | 48.9 |
| Godson1_sim | 0.3755 | 0.1978 | 0.1137 | 0.4039 | 0.1129 | 0.2796 | 0.1342 | 0.2204 | 0.1071 | 0.1419 | 0.0801 | 0.1970 |
| %difference | -2.2 | 5.2 | 2.5 | 6.6 | 6.1 | 2.4 | -17.5 | -5.4 | 16.3 | 20.4 | -16.1 | 1.1 |

如信号级模拟器在猜测取指或访存发生 cache miss 时,需要占用内存端口,但是不会修改 cache 的内容,性能模拟器却修改了 cache 的内容.经过验证,性能模拟器的准确度得到进一步的提高,同时,由于施加了更多的约束,性能模拟器的执行速度会有不同程度的下降.表1是模拟器验证前后IPC偏差对比表,non-validated Godson1_sim 是未验证的模拟器,

表 5 模拟器速度比较(KIPS)

| | Gzip | mcf | gcc | parser | Perl | vortex | apsi | Ammp | art | equake | sixtrack | mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Godson1_ICT | 2.11 | 1.14 | 0.68 | 2.25 | 0.65 | 0.91 | 0.99 | 1.40 | 0.56 | 0.71 | 0.58 | 1.1 |
| Sim-outorder | 267.18 | 172.91 | 220.63 | 307.31 | 252.13 | 413.13 | 268.84 | 167.92 | 228.01 | 314.87 | 371.89 | 271.3 |
| Speed up | 126.6 | 151.7 | 350.2 | 136.6 | 387.9 | 454.0 | 271.6 | 119.9 | 407.2 | 443.5 | 641.2 | 246.6 |
| Godson1_sim | 291.53 | 181.16 | 201.60 | 408.02 | 231.87 | 285.59 | 137.83 | 276.86 | 93.28 | 373.36 | 93.50 | 234.1 |
| Speed up | 138.2 | 158.9 | 296.5 | 181.3 | 356.7 | 313.8 | 138.1 | 197.8 | 166.6 | 525.9 | 161.2 | 212.8 |

validated Godson1_sim 是已验证的模拟器,从表中可以看出,验证后的模拟器 IPC 偏差从 15.1%下降到 7.9%.

## 4 实验及结果分析

我们以龙芯 1 号信号级模拟器 Godson1_ICT 为参考,比较了 sim-outorder、Godson1_sim 的运行速度及 IPC.由于 Godson1_ICT 运行速度只有 1K IPS 左右,运行完一个 test 规模的 SPEC 测试程序需要三四天的时间,因此我们对每个测试程序最多运行 5 亿个时钟周期.本实验环境如表 2 所示,龙芯 1 号微处理器参数配置如表 3 所示,实验结果分别在表 4 和表 5 中给出.从试验结果可以看出,sim-outorder IPC 的最大偏差可以达到 295.7%,平均偏差为 48.9%,平均运行速度为 271.3 k IPS;Godson1_sim IPC 最大偏差为 20.4%,平均偏差为 1.1%,平均运行速度为 234.1 k IPS.和 sim-outorder 相比,Godson1_sim 的运行速度虽稍有下降,但准确度大有提高.我们还发现,在龙芯 1 号性能模拟器运行结果中,有几个浮点测试程序 IPC 偏差较大,速度下降得也较明显.如运行 apsi、Art 和 sixtrack IPC 偏差都在 16%以上,速度也只有 100k IPS 左右.这是因为这几个测试程序参考的 IPC 本身就很小,即使 IPC 的绝对偏差很小,也会表现出较大的相对偏差,同时,它们的运行速度表明,进一步施加约束提高准确度的空间已经不大了.

## 5 结论与进一步的工作



龙芯 1 号性能模拟器模拟了龙芯 1 号微处理器的结构,实现了灵活性、运行速度及准确度三者之间的平衡.由于龙芯 1 号信号级模拟器和龙芯 1 号 RTL 模型在微结构上还有点差异,我们计划先在龙芯 1 号 RTL 模型上实现一个性能计数系统,统计在该模型上运行程序的行为特征,然后将龙芯 1 号性能模拟器和 RTL 模型作进一步的比较、验证.

**References:**


[1] Bose P, Conte T M. Performance analysis and its impact on design[J]. Computer, 1998, 31(5): 41-49.

[2] Reilly M, Edmonidson J. Performance simulation of an alpha microprocessor[J]. Computer, 1998, 31(5): 50-58.

[3] Moudgill M, Wellman J, Moreno J. Environment for powerPC microarchitecture exploration[J]. IEEE Micro, 1999, 19(3): 15-25.

[4] Austin T, Larson E, Ernst D. SimpleScalar: an infrastructure for computer system modeling[J]. Computer, 2002, 35(2): 59-67.

[5] Rosenblum M, Herrod S A, Witchel E, et al. Complete computer system simulation: the simos approach[J]. IEEE Concurrency, 1995, 3(4): 34-43.

[6] Wunderlich R E, Wenisch T F, Falsafi B, et al. SMARTS: accelerating micorarchitecture simulation via rigorous statistical sampling[C]. In: Proc of the 30th annual international symposium on computer architecture. New York: ACM Press, 2003. 84-97.

[7] Desikan R, Burger D, Keckler S W. Measuring experimental error in microprocessor simulation[C]. In: Proc of the 2001 symposium on software reusability: putting software reuse in context. New York: ACM Press, 2001. 266-277.

[8] Burger D, Austin T D. The simplescalar tool set[J]. version 2.0. ACM SIGARCH computer architecture news, 1997, 25(3): 13-25.

[9] Hu Wei-wu, Tang Zhi-min. Architecture of the Godson1 processor[J]. Chinese Journal of Computers, 2003, 26(4): 385-396.


**附中文参考文献:**


[9] 胡伟武,唐志敏.龙芯 1 号处理器结构设计[J].计算机学报,2003,26(4):385-396.