中国科学: 信息科学 2015 年 第 45 卷 第 4 期: 480–500
www.scichina.com info.scichina.com
《中国科学》杂志社
SCIENCE CHINA PRESS
中国 CPU 与 DSP 专刊 · 论文

龙芯 GS464E 处理器核架构设计

吴瑞阳[①②③*], 汪文祥[①②], 王焕东[④], 胡伟武[①②④]

① 计算机体系结构国家重点实验室, 中国科学院计算技术研究所, 北京 100190
② 中国科学院计算技术研究所, 北京 100190
③ 中国科学院大学, 北京 100049
④ 龙芯中科技术有限公司, 北京 100190
* 通信作者. E-mail: wuruiyang@ict.ac.cn

收稿日期: 2014–10–16; 接受日期: 2015–01–08
国家科技重大专项 "核高基"(批准号: 2009ZX01028-002-003, 2009ZX01029-001-003, 2010ZX01036-001-002, 2012ZX01029-001-002-002, 2014ZX01020201, 2014ZX01030101)、国家自然科学基金 (批准号: 61221062, 61133004, 61173001, 61232009, 61222204, 61432016) 和国家高技术研究发展计划 (863 计划) (批准号: 2012AA010901, 2012AA011002, 2013AA014301) 资助

摘要 龙芯 GS464E 是龙芯公司最新推出的高性能处理器核架构. 在本文中, 将介绍 GS464E 架构的核心特性. 相比于之前的 GS464 架构, 重点强化了访存性能和分支预测准确率, 实现了 MIPS DSP 指令集和虚拟机支持, 增大了处理器中各项队列的项数, 并增大了 Cache 容量和 TLB 容量. 访存子系统拥有 3 级 Cache 结构, 每一级都采用 LRU 替换策略, 可以支持多核缓存一致性协议. 经过上述强化设计, GS464E 处理器核已成为一个创新性的高性能处理器核架构.

关键词 处理器核 多核处理器 分支预测 访存性能 缓存一致性

1 引言

多年来, 龙芯高性能处理器芯片一直采用龙芯 2F [1] 芯片设计的 GS464 处理器核 IP(intellectual property), 该 IP 使用 64 位数据宽度, 支持 4 发射和乱序发射、动态流水线技术, 并支持多核一致性扩展. 该处理器核历经多个版本的流片, 龙芯 3A[2,3]、龙芯 3B[4~6] 等多款芯片都使用了这个架构. 然而, 在多年的使用中, GS464 架构的一些性能问题逐渐显现出来, 尤其是低效率的流式访存性能和较低的分支预测准确率. 因此, 针对上一款处理器核架构的缺陷, 调研了 IBM 公司的 Power7 [7]、Intel 公司的 Ivy Bridge [8] 以及 ARM 公司的 Cortex-A9[1)] 等多款世界尖端芯片的设计规格, 龙芯公司于 2012 年开始研发 GS464E 处理器核 IP. 该处理器核的设计采用多项创新技术, 重点解决积垢已久的性能瓶颈. 使用该处理器核的龙芯 3A1500 四核处理器已于 2014 年流片.

GS464E 处理器核架构的主要特点有:

• 该处理器核重点提升了单线程的执行性能. 在本次设计中, 优化了分支预测准确率, 采用了双访存部件的设计, 使用近期最少使用 (least recently used, LRU) 替换算法的一级数据 Cache, 设计了一套激进的预取策略, 降低定点指令相关带来的延迟, 并采用更大的发射队列、重定序队列及更大的牺牲 Cache (Victim Cache). 上述设计大幅提高了单线程程序的性能.

1) A9 processor. http://www.arm.com/products/processors/cortex-a/cortex-a9.php.

引用格式: 吴瑞阳, 汪文祥, 王焕东, 等. 龙芯 GS464E 处理器核架构设计. 中国科学: 信息科学, 2015, 45: 480–500, doi: 10.1360/N112014-00292

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

• 片上共享 Cache (shared cache, SCache) 在维持于每个 SCache 体 1 MB 容量不变的情况下, 设计为 16 路组相联, 使用 LRU 替换策略, 并采用标识部分 (tag) 和数据部分 (data) 分开读取的设计以控制其功耗. 新的 SCache 设计提高了多核多任务下的片上末级 Cache 性能.

• 定义了 LoongISA 指令集结构, 该指令集将应用于龙芯公司即将设计的多款处理器. 该指令集包含了 MIPS DSP 指令集[2)], 实现了虚拟机支持, 并设计有 X86 及 ARM 的二进制翻译加速指令. 为了在 16 KB 每路的一级 Cache 的情况下支持 4 KB 的页大小, 还加入了硬件抗别名 (anti-alias) 支持.

• 在处理器核与片上末级 Cache 之间、以及片上末级 Cache 和内存系统之间, 都采用 AXI 接口, 与之前的 GS464 处理器系列兼容, 可以很容易地更换另一种处理器核或 SCache, 从而形成一款新的芯片设计.

GS464E 处理器核的流片采用 40 nm CMOS 工艺. 在该工艺下, 每个处理器核的大小为 4.30 mm × 5.22 mm, 面积约为 22.72 mm^2; 每个 SCache 大小为 3.00 mm × 4.29 mm, 面积约为 12.86 mm^2. 由 4 个处理器核、4 个 SCache, 以及 HT 高速总线控制器和内存控制器等外部设备组成的多核芯片大小为 17.66 mm × 14.08 mm, 面积约为 248.60 mm^2.

GS464E 处理器核可以用于搭建一个共享 SCache 的一致性多核芯片, 使用基于目录的 MESI 缓存一致性协议. 按照当前的目录实现方案, 最多可以支持在一个芯片中集成 32 个处理器核和 32 个 SCache.

2 GS464E 处理器核设计

图 1 显示了 GS464E 处理器核的 floorplan 图, 如图中所示, 处理器核被分为几个主要部件: 标注为 FETCH 的取指部件、标注为 REGMAP&ROQ 的寄存器重命名与指令重定序部件、标注为 FIX 的定点部件、标注为 FLOAT 的浮点部件、标注为 MEMORY 的访存部件、标注为 CACHE2MEM 的缓存失效队列及 256 KB 大小的 16 路组相联 Victim Cache.

相比于 GS464 的处理器核, 每个部件的内部设计都有大幅度的改进:

• 取指部件. 取指部件的功能是通过分支预测持续获得指令流, 并进行指令译码. 在 GS464E 处理器核中, 通过重新设计, 消除了分支指令预测跳转后的取指空泡; 通过加入循环缓冲器 (loop buffer), 使得最多 56 条指令组成的循环程序执行时不需访问指令 Cache.

• 寄存器重命名. 寄存器重命名部件的功能是进行各种寄存器的重命名, 用于动态流水线中的乱序发射. GS464E 处理器核中的寄存器重命名表扩展到 128 项定点物理寄存器和 128 项浮点物理寄存器, 此外, DSP 控制寄存器、HILO 寄存器以及浮点比较结果寄存器都单独进行重命名.

• 指令重定序. 指令重定序部件的功能是将乱序发射的指令进行重定序, 并进行分支预测错与指令例外的处理. 指令重定序队列 (reorder queue, ROQ) 从 64 项增大到 128 项; 分支指令队列从 8 项扩大到 24 项.

• 定点部件. 定点部件用于进行定点计算. 在 GS464E 处理器核中, 通过激进的计算结果提前反馈 (forward) 逻辑设计, 存在寄存器数据相关的 2 条定点指令之间的延迟 (load-to-use) 从 2 个时钟周期减少到了 1 个; 此外还加入了数字信号处理 (digital signal processing, DSP) 功能部件.

• 浮点部件. 浮点部件用于进行浮点计算. 浮点比较结果寄存器现在可以单独进行重命名, 因此浮点比较以及使用浮点比较结果的指令可以乱序发射和动态流水.

2) The MIPS DSP module for the MIPS64 architecture. http://www.mips.com.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

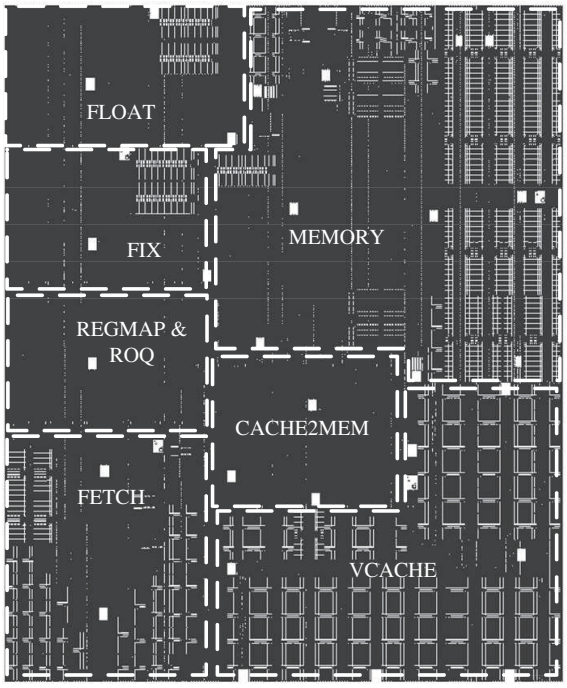


图 1　GS464E 处理器核的平面布置图

Figure 1　Floorplan of GS464E processor core

• 访存部件. 访存部件用于处理访存指令. 访存指令专用发射队列大小为 32 项, 可以将访存指令乱序发射到 2 个访存功能部件, 每个功能部件均可执行读取 (load) 或存储 (store) 指令; 访存重定序队列的项数从 24 项提升至 64 项. 一级数据 Cache 采用了 LRU 替换策略, 并将 Cache 行大小扩展为 64 字节长. 地址翻译快速查找表 (translation lookaside buffer, TLB) 的项数也有大幅扩充, 采用了 64 项可变大小页外加 1024 项固定大小页的双重 TLB 设计.

• 缓存失效队列. 缓存失效队列由指令 Cache 失效请求与数据 Cache 失效请求所共用, 用于处理缓存失效并重填 Cache 行. 在 GS464E 处理器核中, 其项数从 8 项提升至 16 项, 并实现了激进的指令和数据预取引擎, 该预取引擎会根据缓存失效队列中的信息自动生成预取请求, 并通过缓存失效队列来处理这些预取请求. 该预取机制不会导致一级 Cache 污染.

• Victim Cache. Victim Cache 是片内的第二级 Cache, 当一级 Cache 失效时被查询. 在 GS464E 处理器核中, 其总容量扩大为 256 KB, 相联结构从 8 路组相联改为 16 路组相联, 并采用了 LRU 替换策略, 提高了 Victim Cache 的命中率.

指令在流水线中的执行动态流程及各级流水线宽度如图 2 所示.

下面将详细描述各个部件的设计. Vimtim Cache 的设计将在片上 Cache 结构一章进行介绍.

2.1　取指部件

GS464E 处理器核中的取指部件通过高准确率的分支预测机制, 获取持续的指令流并进行译码, 处理后的指令序列提供给后续的流水级. 取指部件主要包括一级指令 Cache、指令 TLB、分支预测器、指令队列以及译码部件. 在取指部件中的指令都是按顺序的, 并不存在乱序执行的情况, 其输出的指令译码信息也是按照程序指令序来排列的. 图 3 是取指部件的流水级示意图.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

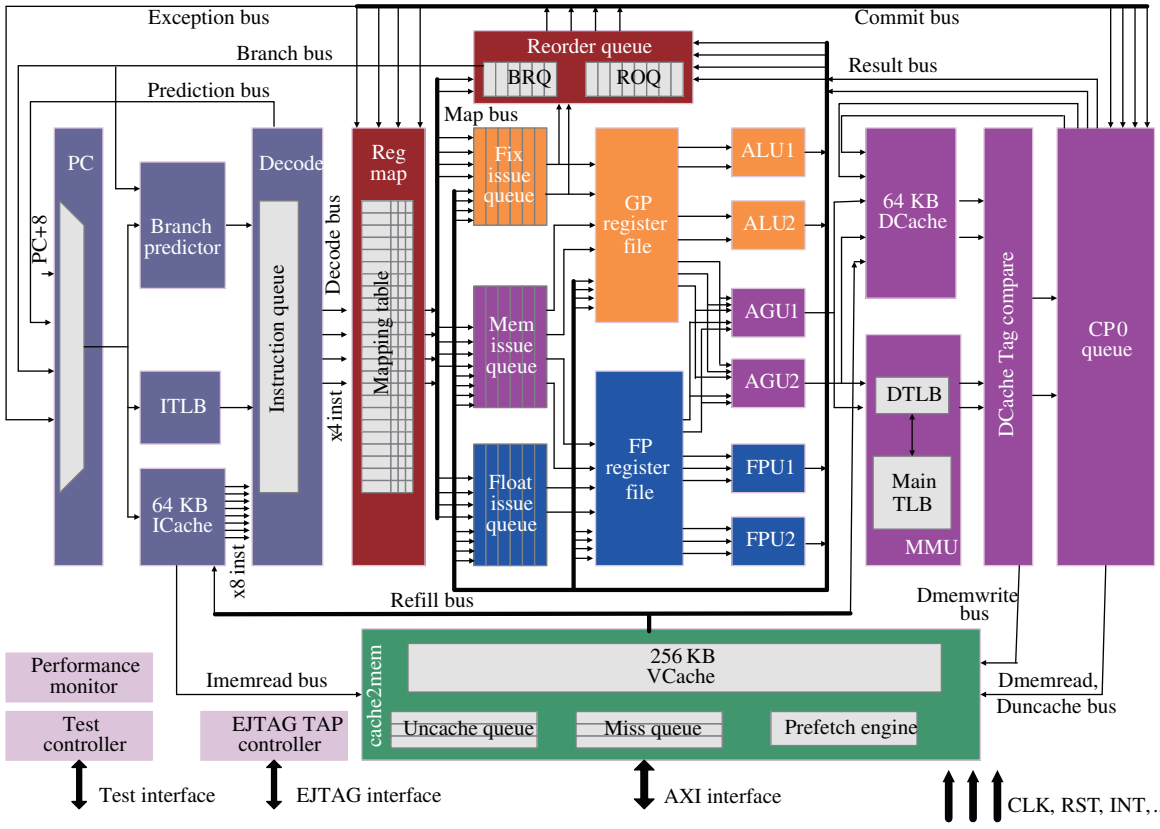


图 2 (网络版彩图) 流水线示意图

Figure 2 (Color online) Outline of processor pipeline

2.1.1 访问指令 Cache

每个 GS464E 处理器核包含 1 个 64 KB 容量、4 路组相联的指令 Cache, 其 Cache 行大小为 64 字节, 指令 Cache 的 Tag 和 Data 部分同时访问并在下一时钟周期进行命中的判断与选择. 其中 Data 部分划分为 8 个体 (bank), 每次取指令时只用读取所需要的几个体, 降低了 Cache 访问时的动态功耗. 指令 Cache 使用虚地址索引和实地址标识的索引方案.

取指部件的虚实地址转换通过 64 项全相联结构的指令 TLB 完成, 指令 TLB 中的内容是页表的子集. 其中, 每项存放 1 个页表项, 每个页表项均可支持 4 KB~1 GB 之间不同大小的页.

当指令 Cache 命中时, 取指部件每个时钟周期最多可以从指令 Cache 中取出 8 条指令, 并送往下一个流水级进行分支预测. 而当指令 Cache 不命中时, 取指部件将生成请求并访问缓存失效队列, 访存失效队列负责进行缓存缺失处理, 并将结果返回给指令 Cache, 其具体流程在缓存失效队列一节中介绍. 从缓存失效队返回的结果将整体填回指令 Cache. 指令 Cache 最多可容忍 3 个 Cache 行不命中, 即有 3 个未完成 (outstanding) 的 Cache 行不命中时, 依旧可以从正确的地址取指令. 当取指请求恰好落在从缓存失效队列返回的 Cache 请求数据上时, 可以直接取出相应指令进入下一个流水级. 此外, 指令 Cache 的预取由缓存失效队列中的硬件预取引擎统一完成, 不在取指部件中单独进行.

指令 Cache 的 Data 部分中还包含预译码信息, 每条指令的预译码信息为 8 bit 大小, 主要用于指令切分、分支类型判定和部分指令例外的判定. 预译码信息通过在缓存失效队列重填指令 Cache 时计

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

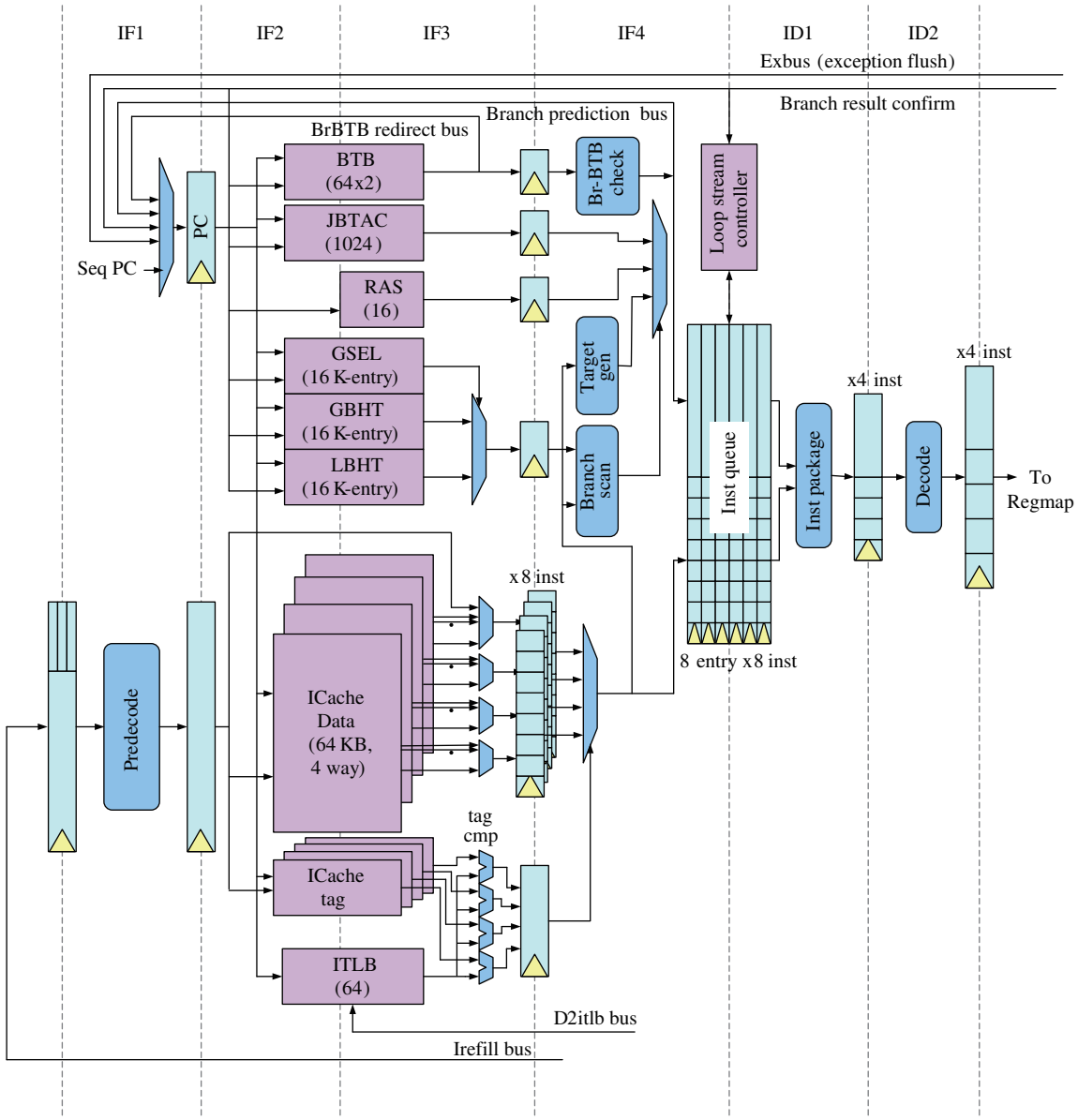


图 3 (网络版彩图) 取指部件流水线图

Figure 3 (Color online) Pipeline of instruction fetch unit

算得出, 并存储在指令 Cache 中. 使用预译码信息而非指令码来进行分支预测可以降低物理设计的复杂度.

2.1.2 分支预测

GS464E 处理器核采用了多种不同的机制来预测分支指令的跳转方向和跳转目标, 并通过分支目标缓冲器 (BrBTB) 来处理分支指令之后取指空泡的情况. 分支预测机制在每个时钟周期最多可以处理 4 条分支指令, 但只有最后一条可以是预测为跳转的分支指令, 预测为跳转的分支指令的后续指令将在下一个时钟周期进行处理.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

GS464E 处理器核采用 3 个时钟周期延迟的 “取指 — 分支” 循环迭代设计, 即第 1 个时钟周期根据程序计数器 (program counter, PC) 的值访问指令 Cache; 第 2 个时钟周期指令 Cache 查询完毕, 取出了最多 8 条指令; 第 3 个时钟周期逐条解析指令, 预测分支指令的跳转方向和跳转目标, 如果取出的多条指令中存在预测为跳转的分支指令, 则需要根据此类指令中最早的那条来更新 PC 值.

由于取出指令并预测出后续指令 PC 需要花费 3 个时钟周期, 其中有 2 个时钟周期无法确保能取出有意义的指令, 为避免性能损失, GS464E 处理器核实现了分支目标缓冲器 (BrBTB), 直接根据取指所使用的 PC 值预测后续指令的 PC 值. 如果取出的指令中没有预测为跳转的分支指令, BrBTB 也没有预测出需要跳变的 PC 值, 则取指部件从当前 PC 向后顺序取指令. BrBTB 为 128 项全相联结构, 使用当前时钟周期的 PC 值来索引下个时钟周期的预测 PC, 在其预测正确的情况下, 即使有需要跳转的分支指令, 取指流水线也不会断流.

BrBTB 的功能是根据当前 PC 来预测下一个时钟周期的取指 PC, 用于消除分支指令之后的取指空泡, 但是分支预测工作还是需要解析不同的指令来进行不同处理的. 分支指令跳转方向的预测使用一套组合分支历史表 (BHTs), 包括 1 个 16384 项的全局转移历史表 (global branch history table, GBHT), 1 个 16384 项的局部转移历史表 (local branch history table, LBHT) 和 1 个 16384 项的全局选择历史表 (global branch select table, GSEL). 每个时钟周期取出的最多 8 条指令可以同时查找这套分支历史表. 上述 3 个分支历史表中的每项都是 2 bit 饱和计数器, 其最高比特位在 GBHT 和 LBHT 中用于指明跳转方向, 而在 GSEL 中则用于决定选用 GBHT 还是 LBHT 的预测方向.

间接跳转指令的跳转目标预测分为两种情况: (1) 函数返回跳转指令 (MIPS 指令集中的 JR r31 指令) 采用 16 项的返回地址栈 (return address stack, RAS) 进行预测. 当分支预测阶段发现函数调用跳转指令时, 将该指令延迟槽之后指令的 PC 值压入 (push) 至 RAS 中. 当分支预测阶段发现函数返回跳转指令时, 则弹出 (pop) RAS 栈顶所存放的 PC 值作为跳转目标的预测值. 为了防止在错误推测路径上执行对 RAS 栈顶指令或栈顶内容的错误修改, RAS 的栈顶指针和当前栈顶的 PC 值都有其临时备份, 用于自纠正. (2) 除上述函数返回跳转指令之外的的间接跳转指令使用项数为 1024 的跳转目标缓存器 (jump branch target address cache, JBTAC) 进行预测, 该缓存器使用取指 PC 及跳转历史信息进行索引.

当指令被取出后, 会根据预译码信息的指示, 按照不同的指令类型, 分别使用上述 3 种机制中的某一种机制进行分支预测. 经过分支预测阶段后, 指令被存入大小为 64 项的指令队列中. 在 GS464E 处理器核中, 基于该指令队列的资源还实现了循环缓冲器 (loop buffer). 循环缓冲器会监测进入指令队列的指令流 PC 特征, 当发现指令流中包含一个不大于 56 条指令的单层循环时, 将停止从指令 Cache 取指, 而是直接从指令队列中取出指令送到译码阶段. 当循环次数达到并退出循环时, 循环缓冲器会被清空.

译码功能部件在每个时钟周期会从指令队列头部取出 4 条指令进行译码. 译码过程的行为较简单, 就是将指令码翻译为方便功能部件处理的内部码, 标识出指令类型、所需要操作的寄存器号以及指令码中可能包含的立即数, 用于接下来的寄存器重命名阶段.

2.2 指令分配与提交部件

指令分配与提交部件负责指令在处理器核中的分配与定序, 具体包括寄存器重命名、指令分配、指令发射、指令重定序、指令提交及分支与例外处理这几个功能.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

2.2.1 寄存器重命名

译码后的指令首先进行寄存器重命名, 随后根据操作类型分配至不同的发射队列中. 寄存器重命名是动态流水线中进行乱序发射的关键技术, 将为指令中指定的逻辑寄存器各自分配一个物理寄存器.

在 GS464E 处理器核中, 需要进行重命名的寄存器有: 通用定点寄存器、通用浮点寄存器、HILO 寄存器、DSP 控制寄存器、浮点比较结果寄存器. 这些需要重命名的寄存器各自进行映射, 通过物理寄存器映射表 (physical register map table, PRMT) 来保存物理寄存器和逻辑寄存器之间的关系. 用于通用定点寄存器重命名的为 128 项 64 bit 宽的寄存器堆, 用于通用浮点寄存器重命名的为 128 项 64 bit 宽的寄存器堆, 用于 HILO 寄存器重命名的为 16 项 128 bit 宽的寄存器堆, 用于 DSP 控制寄存器重命名的为 32 项 32 bit 宽的寄存器堆, 用于浮点比较结果寄存器重命名的为 32 项 32 bit 宽的寄存器堆.

2.2.2 发射队列

GS464E 处理器核中有 3 个独立的发射队列: 16 项的定点发射队列、24 项的浮点发射队列以及 32 项的访存发射队列. 其中定点发射队列负责源操作数和目标操作数均为通用定点寄存器、HILO 寄存器或 DSP 控制寄存器的运算指令和分支指令, 浮点发射队列负责源操作数和目标操作数均为通用浮点寄存器或浮点比较结果寄存器的运算指令和分支指令, 访存发射队列除了负责所有定点、浮点的访存操作指令外, 还包括控制寄存器操作指令 (CP0 指令) 以及在定点与浮点寄存器间交互数据的指令.

从控制功耗的角度出发, 3 个发射队列均采用移动指针而非移动队列中存储内容的管理策略, 每个时钟周期需要计算当前应当发射指令的指针. 只有源寄存器都已经就绪的指令才能被发射, 而指令在寄存器重命名阶段时就会先检查其源寄存器是否已经就绪. 若其源操作数还没有准备好, 则该指令在进行指令分配以及在发射队列中的时候, 都要将自身的源寄存器号同结果总线或 forward 总线的目标寄存器号相互比较, 以确定本条指令所需的源寄存器是否就绪. 此时所使用的寄存器号都是经过寄存器重命名后的物理寄存器号. 上述的 3 个发射队列都采用乱序发射机制, 指令的所有源操作数只要准备好就可以发射. 当存在多个发射候选时, 最早进入的指令具有最高的优先级.

2.2.3 指令提交和重定序队列

在 GS464E 处理器核中, 指令被顺序译码和重命名、乱序发射和执行, 但是要有序提交 (commit). 重定序队列负责指令的有序结束, 它从寄存器重命名模块获取程序指令序信息, 并有序地保存流水线中所有已经完成寄存器重命名但未提交的指令. 指令在功能部件执行完毕并写回 (writeback) 后, 重定序队列按照程序指令序顺序提交这些指令.

重定序队列最多可同时容纳 128 条指令. 重定序队列每个时钟周期最多可以提交队列顶端的 4 条已经处于写回状态的指令. 指令的提交信息会送往寄存器重命名模块, 用于修改重命名的状态, 同时还需要通知访存重定序队列, 因为 store 指令需要提交后才能修改存储器的内容.

GS464E 处理器核实现了精确例外, 当取指令、译码或执行时发生例外时, 例外信息被送至重定序队列保存下来, 只有例外指令成为重定序队列头时, 才进行例外报告与处理. 硬件进行的例外处理工作包括: 将例外原因、例外指令的 PC 值等例外信息记录到有关的 CP0 寄存器中, 并根据例外类型把例外处理程序的入口地址送到程序计数器中.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

2.2.4 分支指令队列与分支预测的错误恢复

分支指令在寄存器重命名后进入重定序队列和发射队列的同时还会顺序地进入分支指令队列. GS464E 处理器核中的分支指令队列最多可以容纳 24 条分支指令, 该队列会记录分支指令进行分支预测时的预测结果.

分支指令和其他指令一样都需要在功能部件执行, 但是其结果总线额外包含分支结果, 分支结果会写回到分支指令队列. 这些结果包括 JR 和 JALR 指令的目标地址, 以及条件转移指令的转移方向, 利用这些结果可以判断这条指令的分支预测是否准确. 不论预测成功与否, 分支指令的执行结果都会反馈到取指部件, 用于修正相关的分支预测器, 以帮助后续分支指令的预测.

预测错误的分支指令和在它之后取进来的指令都需要取消. 分支指令队列负责发送分支错误取消总线, 根据队列中记录的程序指令序, 准确地进行分支取消操作, 同时将正确的 PC 值送到程序计数器中.

2.3 定点部件与浮点部件

GS464E 处理器核中的定点部件包含 1 个多端口定点寄存器堆以及 2 个完全相同的定点运算部件. 定点寄存器堆共 128 项, 一共有 8 个读端口和 4 个写端口. 每条定点运算流水线中均包含: 用于执行定点加、减、比较、陷阱指令的算术逻辑单元 (arithmetic logic unit, ALU), 用于执行移位、循环移位和比特提取与截断指令的循环桶形移位器, 位操作单元, 前导零计数器, 分支处理单元, 除法部件和乘法器, 每种运算单元的数量均为 1 个. 所有频繁执行的指令均在 1 个时钟周期内执行完毕, 且通过激进的 forward 机制设计, 使得存在寄存器相关的多条指令可以背靠背连续发射. 乘法器采用全流水设计, 可进行 64 bit×64 bit 有符号或无符号乘法运算, 运算延迟为 3 个时钟周期, 全流水设计使得每个乘法器在每个时钟周期都可接受 1 条新的指令, 并产生 128 bit 的乘积结果.

GS464E 处理器核中的浮点部件包含 1 个多端口浮点寄存器堆以及 2 个完全相同的浮点运算部件. 浮点多端口寄存器堆共 128 项, 一共有 8 个读端口和 4 个写端口. 每条浮点运算流水线中均包含: 格式转换单元 (用于执行 “定点转为浮点”、“单精度浮点转为双精度浮点” 这类格式转换指令), 浮点比较单元, 浮点除法单元, 浮点开方与求倒数单元, 以及浮点乘 — 加熔和 (fused multiply–add) 单元 (用于执行浮点加、减、乘、乘加、乘减指令), 每种运算单元的数量均为 1 个. 其中格式转换单元采用 3 级全流水设计, 浮点乘 — 加熔和单元采用 4 级全流水设计.

2.4 访存部件

GS464E 处理器核的访存部件用于执行访存指令及定点 — 浮点交互的指令. 相比于之前的芯片, GS464E 采用了回滚式访存重定序机制, 大大降低了物理设计的难度, 还实现了可以在多核情况下使用的存储填充机制. 新设计的访存部件包含两条访存功能流水线, 每条流水线都可以流水地执行 load 或 store 操作.

如图 4 所示, 访存部件包含以下的子部件: 访存指令发射队列 mmqueue、访存专用定点寄存器堆 mr、两个访存地址生成部件 memaddr, 一级数据 Cache 部件 DCache、一级数据 TLB 部件 DTLB、两个访存 Tag 比较部件 dtagcmp、访存重定序队列 cp0queue, 以及二级 TLB 部件 sec_tlb. 只有一份的子部件由两条访存流水线共享使用.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

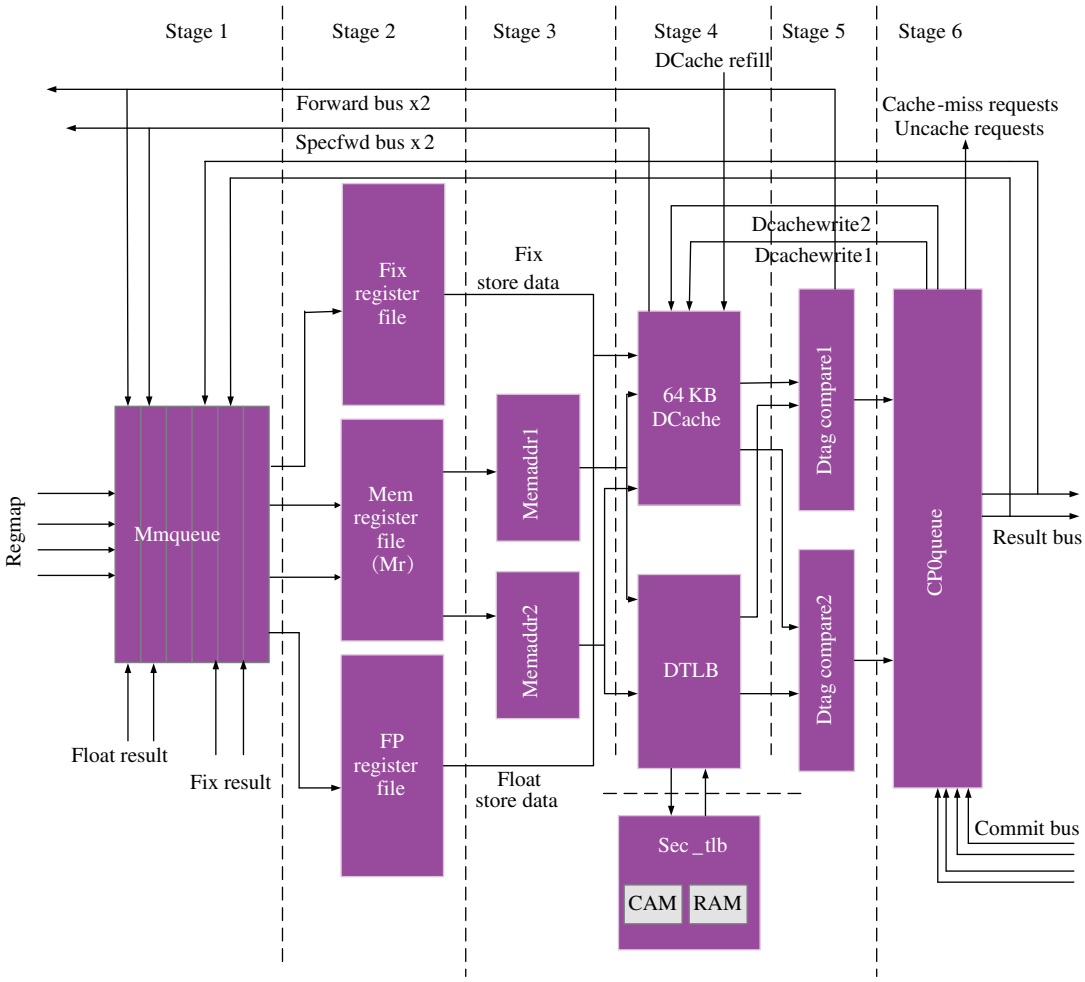


图 4　(网络版彩图) 访存部件流水线图

Figure 4　(Color online) Pipeline of memory unit

2.4.1　访存流水线

访存指令在访存流水线中的执行过程在图 4 已有所展示. 本节详细介绍访存流水线每一级所实现的功能.

访存指令发射队列 mmqueue 接收寄存器重命名部件送来的指令信息, 并将当前已经就绪的、在程序指令序上最老的访存指令发射到访存流水线上. 大部分访存指令都可以在两个访存部件中的任意一个执行, store 操作和 load 操作一样, 会等待地址和数据同时就绪后, 才发射到流水线中.

两条访存流水线在拥有 4 个读端口的寄存器堆 mr 中读取所需的操作数; 一些特殊指令需要发送额外请求去定点部件或浮点部件, 以获取所需的源数据, 例如, 浮点 store 指令 SDC1 所需要存储的数据只存放在浮点寄存器堆中. 所有类型的访存指令的地址肯定会存放在 mr 中, 不需要从定点部件或浮点部件取.

两条访存流水线分别拥有一个地址计算模块 memaddr. 地址计算模块根据访存指令的类型, 利用 mr 读出的寄存器内容计算访存指令的虚拟地址 (effective address, EA 或虚地址). 虚地址低位送到 DCache 模块进行索引查找, 整个虚地址送到 DTLB 模块进行虚实地址翻译.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

一级数据 Cache 是由 4 路组相联、每一路 16 KB 大小的双端口随机存储器 (random access memory, RAM) 构成的, 两个端口分别对应一条访存流水线. 每个端口根据地址的低位读取 RAM 中存放的 tag 和 data, 并将读出的结果送至 dtagcmp 部件准备进行 tag 比较. 同时, store 指令需要存储的数据在此部件计算, 同样也送到 dtagcmp 部件.

一级数据 TLB 与一级数据 Cache 同时被查询. 一级数据 TLB 部件使用 2 个 64 bit 虚地址作为输入, 查询双端口的全相联组织的 DTLB, 得到 2 条访存指令的虚地址所对应的 48 bit 物理地址 (physical address, PA 或实地址), 并送至 dtagcmp 模块准备进行 Tag 比较. 此外, 大部分 CP0 相关的特权指令, 如 MTC0 及 MFC0 都是在 DTLB 模块执行的.

两个实地址比较模块 dtagcmp 模块分别对应一条访存流水线. 这个模块将 DCache 模块读出的 4 路 tag 与 DTLB 模块得到的实地址进行比较, 确定一级数据 Cache 是否命中, 以及命中在哪一路. 命中路的数据会被取出, 和 DCache 送来的 store 指令所要存储的数据一起计算, 得到访存指令的执行结果, 并送到访存重定序队列 cp0queue.

访存重定序队列 cp0queue 接收程序指令序信息, 接收 dtagcmp 模块送来的指令地址与数据信息, 并通过回滚机制来维护乱序发射的访存指令之间的正确执行序, 该回滚机制将在下一节进行介绍. Cp0queue 除了访存指令重定序工作之外, 在每个时钟周期还需要写回最多 2 条可写回的 load 指令和最多 2 条可写回的 store 指令、将最多 1 条访存失效的指令送往缓存失效队列、将最多 2 条已经提交且被允许写一级数据 Cache 的 store 指令发送到一级数据 Cache 的写端口. Cp0queue 可以被视作访存指令的归宿, 所有访存指令都在这里进行缓存失效处理和写回.

目标寄存器为定点寄存器的 load 类型访存指令在从 DCache 流水级到 dtagcmp 流水级传递时, 会进行猜测提前反馈操作, 称为 specfwd. 在从 dtagcmp 流水级到 cp0queue 传递时, 会重新发送上一个时钟周期的 specfwd 信息, 形成 fwdbus 总线. 两条访存总线的最多 4 条用于提前反馈的信息会被送至定点发射队列和访存发射队列, 并置猜测就绪标记, 使得指令可以猜测地发射. 如果猜测 forward 猜测错误, 则会发送对应该访存流水线的猜测取消信号 spec_cancel, 猜测错误的条件包括: 一级数据 Cache 不命中、或者一级 DTLB 不命中、或者该指令在 cp0queue 被回滚. 猜测取消信号将会取消对应的 specfwd 以及 fwdbus 总线带来的所有后果, 已经猜测发射的相应指令也会被取消, 并回到发射队列重新等待发射机会.

在上述 forward 机制的作用下, 可以得知访存指令在理想情况下的延迟: 访存指令到定点指令以及访存指令到访存指令的 load-to-use 延迟为 4 个时钟周期, 访存指令到浮点指令的 load-to-use 延迟为 6 个时钟周期.

2.4.2 访存重定序机制

访存重定序机制是用于维护乱序发射的访存指令之间执行顺序的机制. 在之前的 GS464 处理器核设计中, 访存重定序采取的是数据传递机制, 使用数据传递机制进行访存指令重定序的方法为: 当一条指令经过发射路径到达重定序队列时, 这条指令需要从所有执行序在它之前的、与其相关的 store 指令处获取对应数据; 同时, 如果这条指令是 store 指令, 那么这条指令还要将它自己的数据传递给所有的已经在重定序队列中、与其相关的、且执行序在它后面的指令. 简单说来, 该机制可以描述成 “取前给后”.

在物理设计上, 实现这样的数据传递机制代价很大: 该指令可能从多条指令处取得数据, 可能从任意指令处拿取任意字节的数据; store 指令可能把自己的数据传递给多条指令, 可能将数据传递给任意指令的任意字节位置. 这个机制的物理设计过于复杂, 严重限制了队列的规模, 致使 GS464 处理器

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

核中的访存重定序队列只有 24 项大小. 在 GS464E 处理器核中实现了 2 个访存功能部件, 同一时刻可能有 2 条指令进入访存重定序队列, 数据传递机制的实现代价更是难以接受.

为了提高指令并行度、将访存重定序队列项数增大, 在 GS464E 处理器核中, 只保留了指令是否相关的判断, 取消了数据传递的功能, 通过指令回滚到发射队列重新发射的机制来保证执行的正确性. 其具体操作方式如下所述: 当一条指令经过发射路径到达重定序队列时, 发现队列中有执行序在它之前的与其相关的 store 指令, 那么该指令回滚到访存发射队列 mmqueue, 并等待该 store 指令写入一级数据 Cache 后再次发射; 如果一条 store 指令经过发射路径到达重定序队列时, 发现队列中已有执行序在它之后的与其相关的指令, 那么将这些指令回滚到 mmqueue, 让这些指令重新发射. 其中, 指令相关的判断会根据指令的操作类型、访存地址等信息进行精确的判断, 例如, 对 A 地址的半字存储 SH 操作并不会使得对 A+2 地址的半字取 LH 操作回滚. 简单说来, 该机制可以描述为 “等前打后”.

通过采取 “等前打后” 的访存重定序策略来代替 “取前给后” 的策略, 略增加了发射队列和访存重定序队列的设计复杂度, 在一些情况下会带来少许性能降低, 但是却大大降低了物理设计的难度, 可以在同样的设计主频下使用更大的访存重定序队列, 增加访存指令并行度, 可以更好的容忍访存延迟. 同时, 也为支持更大的访存宽度, 如 256 bit 向量访存指令, 提供了可能性.

2.4.3 一级数据 Cache 的组织架构

GS464E 处理器核中的一级数据 Cache 采取 4 路组相联结构, 总大小为 64 KB. 其中, 每一路大小为 16 KB, 分为 256 个 Cache 行, 每一行为 64 字节. 一级数据 Cache 使用虚地址索引, 实地址标识, 每一个 Cache 行需要 48 bit 的 tag 来存储实地址信息, 以及 512 bit 的 Data 来存储数据, tag 部分和 Data 部分都使用 ECC 校验码来进行冗余保护. 一级数据 Cache 采用 LRU 替换策略, 当一个 Cache 行被查询命中时, 或因缓存失效而被新填入时, 会被调整到最难被替换的优先级; 如果一个 Cache 行因多核一致性导致的外部请求被无效, 该 Cache 行会被调整至最容易被替换的优先级.

一级数据 Cache 是一个双端口的缓存结构, 每一个时刻最多可以实现 2 条访存指令的查询操作. 同时, store 指令从访存重定序队列发出写 Cache 操作时, 也通过这 2 个端口进行写操作, 其优先级低于访存指令的查询操作. 除此之外, ECC 校验出错时的自纠正操作和缓存失效重填请求都需要从第一个写端口进行写入, 写入操作将按照特定优先级进行, 并对低优先级的操作产生阻塞.

2.4.4 虚实地址翻译机制

GS464E 处理器核中使用了一个软件维护、特殊指令加速的虚实地址翻译查找表 (TLB) 设计, 用于将 64 bit 的虚地址转换为 48 bit 的实地址. 该 TLB 设计支持从 4 KB~1 GB 之间共计 10 种页大小, 支持 MIPS 提供的 RI 和 XI 两种安全保护机制, 还额外设计了 K 域, 可以对内核模式下的内存访问进行保护限制. 此外, 为了提高软件处理 TLB 缺失的速度, 还加入了几条用于辅助 TLB 失效处理的指令.

虚实地址翻译工作在硬件实现上是由两级 TLB 来完成的. 一级数据 TLB 对软件透明, 每次访存操作执行时被查询; 软件可见的是二级 TLB, 软件修改 TLB 的指令 (如 TLBWR) 会直接修改二级 TLB. 二级 TLB 对一级数据 TLB 是包含关系, 一级数据 TLB 的缺失会自动从二级 TLB 中查找并取回, 硬件会自动维护包含关系. 此外, 指令 TLB 也是根据二级 TLB 中的内容填入的.

一级数据 TLB 的规模为 32 项, 二级 TLB 则是由 2 个部分组成, 包括 64 项全相联的可变页大小的地址可寻址存储器 (content addressable memory, CAM) 部分, 以及 8 路组相联、每一路 128 项, 共

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

计 1024 项的固定页大小的 RAM 部分, 这两级 TLB 中的每一项都可以装载 1 个标准 MIPS 双页. 一级数据 TLB 是双端口的, 分别对应于 2 个访存功能部件, 一级 TLB 查询发生不命中时, 会选择在程序指令序上较老的那条指令来查询只有 1 个端口的二级 TLB. 二级 TLB 查询需要花费 2 个时钟周期, 如果命中, 那么命中的结果会写入一级数据 TLB, 其替换策略为随机替换; 如果不命中, 其结果也会通知访存重定序队列. 需要查询二级 TLB 的指令经过访存重定序队列中的回滚机制, 重新回到发射队列, 再次发射时, 二级 TLB 的查询结果已经返回, 因此可以判断是否真正的发生了 TLB 失效例外.

二级 TLB 中可变页部分 (CAM 部分) 和固定页部分 (RAM 部分) 会被同时查询, 但是软件进行 TLB 写入时, 只有其中一个会被修改: 如果写入项的页大小与提前配置好的固定页相同, 则 TLB 指令会将 TLB 表项写入固定页部分, 否则, 将写入可变页部分. 64 项的 CAM 部分和 8 路组相联的 RAM 部分的替换策略都采用随机替换策略.

2.4.5 Store fill 机制

GS464E 处理器核的一级数据 Cache 是一个写回 (writeback) 式的 Cache, 所有的 store 操作都需要对一级数据 Cache 进行写入. 当 store 指令发生缓存失效时, 需要将 store 指令所需的 Cache 行从内存搬运至一级数据 Cache, 再进行写入. 然而在很多情况下, store 指令都倾向于填满整个 Cache 行, 因此这个搬运过程就显得多余, 还带来了不少的时间与功耗开销. 为了降低这个开销, GS464E 处理器核中引入了存储填充 (store fill) 机制.

该机制的工作原理为: 当 store 指令在一级数据 Cache 中发生缓存失效时, 其访存失效请求会将待存储的数据一并送至位于缓存失效队列中的 store fill 缓冲区, 同时, 该访存失效请求暂缓访问 SCache. 如果接下来的多条 store 请求在该缓冲区中成功地拼满了 Cache 行, 那么只需向 SCache 请求空白的 Cache 行即可. 收到空白行请求的 SCache 不会访问内存, 在进行多核一致性处理后直接返回, 缓存失效队列负责将缓冲区中拼满的数据重填回一级数据 Cache. 在 store fill 进行期间, 相应的 store 指令可以退出流水线, 释放队列空间.

如果较长时间没有拼满一个 Cache 行, 或遇到了属于同一个 Cache 行的 load 操作, 或遇到内存屏障类操作 (如 SYNC 或 CACHE 指令), store fill 机制就会进行退出处理, 向 SCache 请求一个正常的 Cache 行, 待结果返回后, 在缓冲区进行数据拼凑操作, 将拼凑后的结果填回一级数据 Cache. 硬件会记录 store fill 成功与否的近期历史, 并对 store fill 机制进行自动调整. GS464E 处理器核中实现的 store fill 机制可以适用于多核处理器的情况, 减少了 store 指令导致的内存读取, 并允许 store 指令提前退出流水线, 提升了处理器的整体执行性能.

2.5 缓存失效队列

缓存失效队列位于一级 Cache 与 SCache 之间, 负责对 Victim Cache 进行访问和管理, 实现缓存失效请求的处理和重填工作, 并负责硬件预取请求的生成和处理. 在 GS464E 处理器设计中, 对缓存失效的处理过程进行了多项创新性的改动, 包括基于缓存失效队列的无污染硬件预取引擎设计以及可以支持多核情况的 store fill 机制.

2.5.1 缓存失效队列的组织结构

缓存失效队列共有 16 项, 会进入该队列的请求包括访存重定序队列发来的访存失效请求、取指部件发来的指令失效请求以及 SCache 发来的多核一致性请求, 此外, 硬件预取引擎会将生成的数据和指令预取请求也送给缓存失效队列, 由该队列进行处理. 所有进队请求的地址均为实地址.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

缓存失效队列中的缓存失效请求会先查询 Victim Cache, 如果其结果命中, 则会将结果重填回一级 Cache, 并将一级 Cache 替换出的 Cache 行填入 Victim Cache 中之前被取出的位置; 如果 Victim Cache 查询不命中, 则会访问 SCache, 待 SCache 的结果返回后, 填入一级 Cache, 将一级 Cache 替换出的有效数据根据 LRU 算法写入 Victim Cache, 并将 Victim Cache 替换出的有效脏数据写回到 SCache. 缓存失效请求总是会先查询 Victim Cache, 查询不命中时再查询 SCache, 这种串行访问的设计降低了缓存一致性维护的复杂度.

缓存失效队列中由 SCache 发来的多核一致性请求会对一级 Cache 和 Victim Cache 进行查询, 根据查询结果和一致性请求的类型, 对 Cache 行进行无效、写回等操作, 并将结果返回 SCache.

缓存失效队列中的硬件预取请求会对 SCache 进行查询, 根据查询结果, 将预取来的数据暂存在队列中. 如果有缓存失效请求和硬件预取请求操作同一个 Cache 行, 那么预取到的数据会交递给这个缓存失效请求. 缓存失效队列会定期清除掉那些一直没有被使用到的预取数据. 这样的预取处理方式不会带来任何一级 Cache 污染.

请求同一个 Cache 行的缓存失效操作会在缓存失效队列中进行合并, 但只限于同属于数据 Cache 失效或指令 Cache 失效. 缓存失效队列还有专门的缓冲区用于存储访存失效 store 指令的数据, 以支持 store fill 机制, 该机制的原理已在上一章中做过解释.

2.5.2 硬件预取引擎

GS464E 处理器核中使用了最多同时支持 4 个数据访问流和 1 个指令访问流的流式硬件预取引擎, 数据和指令的预取相对独立. 其中, 数据流式预取引擎可以支持升序和降序模式, 使用访存失效请求来建立和维护访问流信息; 而指令预取只能支持升序的预取, 使用指令失效请求来建立和维护流信息. 当失效请求为连续的 2 个 Cache 行时, 将会建立访问流, 每个访问流都配有 3 bit 的倒数计数器, 用于计算这个访问流是否已过期. 缓存失效请求如果在某个访问流上延续, 那么配属于这个访问流的计数器会刷新到最大值. 访问流建立后, 就可以触发预取, 预取请求的触发包括下列两种情况.

如果缓存失效请求在缓存失效队列中和预取请求合并, 证明预取是有益的, 将会根据此次失效的地址查询预取引擎中已经建立的访问流信息. 指令失效请求会查询指令预取引擎, 而访存失效请求会查询数据预取引擎. 如果访问流查询命中, 则根据访问流的升序或降序信息, 产生 1 个预取请求, 预取请求的地址是失效地址加上特定增量; 如果访问流查询不命中, 则寻找计数器为零或计数器最小的访问流, 将其顶替.

如果未能和预取请求合并的缓存失效请求在 Victim Cache 中查询不命中, 则证明需要进行预取, 此时会根据此次失效的地址查询预取引擎中的访问流信息, 如果访问流信息查询命中, 则根据访问流的升序或降序信息, 产生预取请求, 预取请求地址是失效地址加上特定增量; 访问流查询不命中则不进行操作.

每次触发预取时, 预取地址相对于失效地址的增量会随着这个访问流的历史触发次数而变化, 被触发过的次数越多, 其预取增量越大. 这个增量值的上限则由已建立的访问流的个数决定: 当只有 1 个访问流存在时, 增量上限被设定为 4 个 Cache 行大小; 当超过 1 个访问流时, 该增量被设定为 2 个 Cache 行大小. 指令预取的增量上限一直设定为 4 个 Cache 行. GS464E 处理器核使用增量预取策略, 相对于每次触发就预取多条的策略, 在访问流预测正确的情况下效果是相同的, 但是访问流预测错误时, 增量预取策略会少一些无效的预取请求.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

2.6 可调试性设计

由于芯片规模越来越大, 验证工作不可能达到 100% 的覆盖率, 因此处理器流片后的硅后调试是所有处理器都需要面对的问题. GS464E 处理器核中采用了 MIPS EJTAG[3)] 硅后调试系统, 符合 EJTAG3.1 规范, 可以在流片后对芯片进行在线调试. EJTAG 可以支持程序计数器采样、指令断点、数据断点、从调试代码段取调试指令、从调试代码段重启处理器等多种调试功能.

除此之外, GS464E 芯片中还新增加了可调试性设计 (design for debug, DFD) 模块, 可以在某些触发条件下, 将时钟停止. 时钟停止后, 可以再配合芯片测试时使用 JTAG 扫描链 [9] 来获取当前的处理器内部所有触发器的状态. 时钟停止的触发条件分为很多种, 每一种都是软件可配置的, 例如当处理器核对外的 AXI 总线上的一个读请求的地址等于配置值时触发, 或某个处理器核对一个配置好的地址进行 store 操作时触发. 条件触发后, 调试主机会收到信号并开始工作, 调试完成后, 可以令时钟继续, 进行下一步调试工作.

3 片上 Cache 层次结构

3.1 三级 Cache 层次

GS464E 处理器核使用了三级片上 Cache 结构, 其中位于每个处理器核内部的私有 Cache 包括一级数据 Cache 和一级指令 Cache, 以及数据和指令共用的 Victim Cache, 第三级 Cache 为 SCache, 是由所有处理器核共享使用的.

3.1.1 一级 Cache 的设计

一级指令 Cache 为 4 路组相联设计, 共计 64 KB 容量, Cache 行大小为 64 字节, 替换策略为随机替换. 一级数据 Cache 和一级指令 Cache 的相联结构、Cache 容量及 Cache 行大小相同, 但采用 LRU 替换策略. 两个一级 Cache 的功能及其读写控制在上文都有过介绍. 一级 Cache 和 SCache 维护包含 (Inclusive) 关系, 在一级 Cache 中的 Cache 行必定在 SCache 中有其对应的备份.

3.1.2 Victim Cache 的设计

Victim Cache 的组织方式为 16 路组相联, 每一路为 16 KB 大小, 共计 256 KB 容量, Cache 行大小为 64 字节, 使用 LRU 替换策略. Victim Cache 位于一级 Cache 之下, 由缓存失效队列进行管理, 当一级 Cache 发生缓存失效时被查询. 查询 Victim Cache 时, 采用的是先读取 tag 再读取命中路的 data 的方式, 使用少量延迟来换取功耗的降低. 在缓存失效请求查询 Victim Cache 仍然失效的情况下, 由缓存失效队列负责向 SCache 发出失效请求.

当一个 Cache 行从一级数据 Cache 或一级指令 Cache 中替换出来时, 会被填入 Victim Cache, 而 Victim Cache 替换出的数据将会写回到 SCache. 指令 Cache 行与数据 Cache 行都可以存放在 Victim Cache 中, 以每一个 Cache 行 tag 域中的一个额外比特来做区分. Victim Cache 与两个一级 Cache 维护排外 (exclusive) 关系, 处于一级 Cache 中的 Cache 行必定不存在于 Victim Cache 中; Victim Cache 与 SCache 维护包含 (inclusive) 关系, 也即处于 Victim Cache 中的 Cache 行必定在 SCache 中有其对应的备份.

3) EJTAG specification. http://www.mips.com.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

3.1.3 SCache 的设计

SCache 为片上末级 Cache, 由所有处理器核共享使用, 一个四核处理器就会拥有 4 个 SCache 以供 4 个处理器核使用. 每个 SCache 为 16 路组相联, 容量为 1 MB, Cache 行大小为 64 字节, 使用 LRU 替换算法. 每个 SCache 使用大小为 16 项的管理队列来查询和维护. 相比于一级 Cache 和 Victim Cache, 每个 SCache 的 Cache 行还额外包含 64 bit 的目录域, 用于记录该 Cache 行被哪个处理器核持有, 以及被处理器核持有的是指令行还是数据行. 在这个目录的设计规模下, 最多支持一个芯片集成 32 个处理器核. 在区分数据行和指令行之外, SCache 并不区分一级 Cache 和 Victim Cache, 因此一级数据 Cache 持有的 Cache 行或 Victim Cache 持有的数据 Cache 行都会被视作该处理器核持有数据 Cache 行.

在 GS464E 处理器核配属的 SCache 中, 加入了基于缓存一致性的硬件抗别名设计. 当程序使用的页的大小比一级 Cache 中每个 Cache 路的容量更小的情况下, 就会出现别名的情况, 同一个物理地址会被放置在一级 Cache 中的多个索引上. 为了保证映射到同一个物理地址的多个虚地址之间的数据是一致的, SCache 在 tag 域中加入了额外的 2 bit 的页染色域, 利用 SCache 对片内两级 Cache 的包含关系, 保证这个 Cache 行只能存在于满足当前页染色域的特定索引位置上. 当一个访存请求要求页染色域的另一个值时, 会发生缓存失效, SCache 发现该请求页染色不命中之后, 会通过一致性请求, 写回并无效处于一级 Cache 或 Victim Cache 之中的 Cache 行备份, 修改页染色域的值为新值后, 再行返回, 以此保证每个物理地址只对应一个实例. 基于缓存一致性的硬件抗别名设计是 GS464E 处理器核的创新性功能, 在保持一级 Cache 大容量的前提下, 实现了对较小页的高效硬件支持.

SCache 中还为使用原子操作指令 (例如, LL/SC 指令) 进行多核的同步加入了特殊设计: 当一个处理器核使用原子操作指令获得一个 Cache 行时, SCache 在短时间内将禁止把这个 Cache 行转让给其他处理器核. 该时间间隔为随机数, 随机的范围可以通过软件进行配置. 通过上述设计, 多个处理器核同时争抢同一个内存地址时, 可以在时间上串行开来, 减少了竞争. 这个特殊设计可以提高多核同步的性能, 并可以防止因每个处理器核都占有时间过短、无法修改共享数据而导致的活锁现象.

在 SCache 中查询失效的请求会向下一级存储设备发出访存请求. 下一级存储设备包括内存, PCI, SPI 或其他外部设备.

3.1.4 各级 Cache 访问延迟

直接在一级数据 Cache 命中的 load 指令延迟为 4 个时钟周期. 相比于直接在一级数据 Cache 命中, 如果该 Cache 行在一级数据 Cache 中失效但是在 Victim Cache 命中, 则会给这条访存指令带来额外 18 个时钟周期的延迟. 如果是 Victim Cache 也失效但是在 SCache 中命中的访存指令, 相比一级 Cache 命中的情况要多 50 个时钟周期的延迟, 其中访问 SCache 需要的时间为 11 个时钟周期, 其他延迟包括请求在片内传递的延迟, 以及为支持处理器核降频设计所需要的异步队列所带来的延迟.

3.1.5 可配置的预取

GS464E 处理器核中设计了块式软件预取指令, 可以依照用户配置, 进行特定的数据预取操作. 软件预取指令根据预取的目的地, 分为预取到一级数据 Cache 和预取到 SCache 两种. 一条软件预取指令最多可以配置为预取 256 个块, 每个块可以包含最多连续 64 个 128 bit 的数据, 块与块之间的间隔最大为 64 KB.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

块式软件预取指令使用虚地址作为操作数, 配置信息为另一个操作数, 使用 TLB 进行虚实地址翻译. 类似于 MIPS 指令集中的预取指令, 软件预取指令在用户态模式下可用, 不会触发执行相关的例外.

3.2 缓存一致性和内存一致性

GS464E 使用的缓存一致性 (Cache coherency) 为基于目录 (directory) 的 MESI 一致性, 共享块想要升级为独占 (exclusive) 块时, 需要先进行无效操作. 当 SCache 处理一个处理器核的缓存失效请求时, 有可能向另一个处理器核发出一致性请求, 包括无效请求、写回请求及无效并写回请求 3 种. 该一致性请求由被请求处理器核的缓存失效队列处理并给予回应.

GS464E 使用的内存一致性 (Memory consistency) 为处理器一致性, 该一致性类似于弱一致性, 仅维护被同步操作隔开的操作之间的顺序.

4 性能评估

4.1 实验平台

GS464E 处理器核在流片前进行了深度的性能评估与性能优化. 性能评估工作基于两个平台, 第一个是寄存器转换级 (register transfer level, RTL) 电路仿真性能评估, 在这个平台上可以运行一些微型程序或手写程序; 第二个更主要的平台是 EVE 仿真加速器, 该仿真加速器是 Synopsys 公司旗下一款硬件验证仿真加速平台, 已被全世界重要的半导体和电子系统公司所采用, 在此平台上可以启动 Linux kernel 并挂载文件系统, 并运行中小型测试程序.

测试平台所使用的处理器频率为 1 GHz, 除有特殊标注的程序外, 所使用的内存均为 DDR3-1000 双通道内存. 对比平台为龙芯 3A 四核处理器, 处理器频率为 1 GHz, 所使用内存为 DDR3-667 双通道内存. 部分测试程序在 Intel Ivy Bridge 处理器上也进行了实验, 所使用的处理器型号为 i7-3770, 主频为 3.9 GHz, 内存为 DDR3-1333 单通道内存.

系统级测试所使用的 Linux 内核版本为 2.6.36, 编译器版本与优化参数在测试程序一节介绍.

4.2 所使用的测试程序

在 RTL 级平台上运行的微型测试程序包括:

• Microbench. 一组用标准 C 编写的简单程序, 包含 C, E 和 M3 类, 分别侧重于分支预测器性能、运算部件流水线效率和访存通路延迟三方面测试, 该程序使用 GCC4.3 编译, 优化参数为 -O2.

• Memcpy. 循环调用 C 库中的 memcpy 函数, 测试 memcpy 的性能分数, 该程序使用 GCC4.3 中的 C 库函数.

• Dhrystone. EEMBC 公司设计的一个侧重字符串处理的小型程序, 该程序使用 GCC4.4 编译, 优化参数为 -O2.

• Whetstone. 一个侧重浮点性能的小型程序, 该程序使用 GCC4.4 编译, 优化参数为 -O2.

• Linpack kernel. Linpack 是线性系统软件包 (linear system package) 的缩写, 是当前最流行的用于测试计算机系统浮点性能的测试程序. Linpack kernel 测试是抽取 Linpack 所使用的 BLAS 库中的 dgemm 核心函数, 循环执行以测试其性能分数. 该核心函数对编译器不敏感, 所使用的是 GCC4.3 编译, 优化参数为 -O2.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

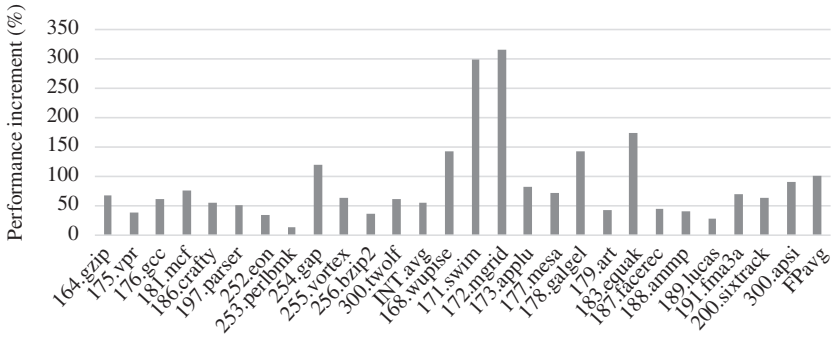


图 5 SPEC CPU 2000 测试结果

Figure 5 Performance increment of SPEC CPU 2000

• 原子操作测试. 一个手写的使用 LL 与 SC 指令进行多核同步的测试, 该测试是一个纯汇编指令写成的测试.

在 EVE 仿真加速器上运行的中小型测试程序包括:

• Stream. 这是一个业界广为流行的综合性内存带宽测量工具, 包括 Copy, Scale, Add 和 Traid 4 种测试, 该程序使用 GCC4.3 编译, 优化参数为 -O2.

• SPEC2000. SPEC CPU 2000 测试是由 SPEC 组织推出的 CPU 子系统的评估软件, 包括定点测试和浮点测试两个部分, 测试规模则包括 test ref train 3 种. 在 EVE 仿真加速器上运行的是最小的 test 测试集, 使用 GCC4.3 编译, 优化参数为 -O2.

• SPLASH2. 由斯坦福大学推出的标准共享内存并行测试程序 (Stanford ParalleL Applications for Shared Memory, SPLASH) 的第二版本, 用于测试以共享内存方式实现的并行程序的性能, 该程序使用 GCC4.3 编译, 优化参数为 -O2.

• Unixbench. 是 Linux 与 Unix 测试系统基准性能的经典工具, 包含进程、管道、运算等多种系统基准操作, 该程序使用 GCC4.3 编译, 优化参数为 -O2.

• Coremark. 由 EEMBC 公司设计的一款测试嵌入式 CPU 性能的基准测试, 包含冗余循环测试、矩阵运算等多种运算公式, 该程序使用 GCC4.9 编译, 优化参数为 -O2.

• Linpack HPL. 这个测试程序是 Linpack 测试的高度并行计算基准测试 (high performance Linpack), 该程序使用 GCC4.4 编译, 优化参数为 -O2.

• Stressapptest. 由谷歌公司推出的一款测试内存或硬盘负载的程序, 在 GS464E 处理器核性能评估时, 仅使用了其中内存负载测试部分, 该程序使用 GCC4.4 编译, 优化参数为 -O2.

• Bogomips. Bogomips 是在 Linux 内核启动时打印的一个分数, 是一个循环减一程序的运行结果.

4.3 性能评估结果

性能评估的结果将按照程序的特性, 在本节予以分类介绍.

SPEC CPU 2000 测试程序的性能测试结果如图 5 所示, 图中纵坐标表示 GS464E 处理器核相比于龙芯 3A 芯片提升的百分比. 结果显示, 定点测试程序的性能平均提升了 54.9%, 而浮点测试程序的性能平均提升了 100.6%. 性能提升的主要原因是由于 GS464E 处理器核使用了更大的队列, 因此可以更好的开发程序的并行性. 此外, 访存性能的提高也会提高 SPEC CPU 2000 的运行速度.

Linpack 性能的提升如表 1 所示. HPL 程序在分块较大的分块策略下, 其性能提升了 4 倍有余.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

表 1 Linpack HPL 测试结果

Table 1 Test result of Linpack HPL

Ns/NBs/NBMINs	GS464E (GFLOPS)	Loongson3A (GFLOPS)	Increment (%)
2048/2048/2048	0.6190	0.1327	366.5
2048/2048/64	1.334	0.7490	78.1

表 2 流式访存程序的测试结果

Table 2 Test result of stream-like programs

Benchmark	GS464E (MB/s)	Loongson3A (MB/s)	Increment (%)
Memcpy	7952.0	410.0	1939.5
Stream-copy	7977.4	338.3	2258.1
Stream-scale	8054.9	388.3	1974.4
Stream-add	7601.6	400.0	1800.4
Stream-triad	7906.4	400.9	1872.2
Stressapptest	1952.2	418.3	366.7

表 3 峰值带宽比例测试结果

Table 3 Test result for peak bandwidth

Processor	Memcpy (MB/s)	Memcpy peak bandwidth (%)	Stream copy (MB/s)	Copy peak bandwidth (%)
GS464E	6150	76.8	5500	68.7
GS464	410	7.7	338	6.3
Ivy Bridge	8500	84.4	6599	65.6

此外, 核心循环 dgemm 的效率从龙芯 3A 芯片上的大约 50% 提升到了 GS464E 处理器核中的大约 90%. 这主要是由于 GS464E 处理器核中加入了块式软件预取指令, 可以将所需要的数据在前面的循环中预取到缓存中. 相比之下, 在 Intel Ivy Bridge 处理器上得到的 dgemm 效率在 90%~97% 之间, 与 GS464E 上的执行效率相比并无太多优势.

性能评估时使用到的流式访存型程序包括 memcpy, stream 与 stressapptest. 其测试结果如表 2 所示. 流式访存的性能相比于龙芯 3A 芯片有了非常大的提升, 很多测试程序都可以提升接近 20 倍的性能. 这说明 GS464E 处理器核中实现的预取引擎是非常有效的.

如果按照访存带宽与内存峰值带宽的占比来比较, GS464E 已经可与 Ivy Bridge 处理器的性能相比. 测试结果见表 3. 由于本实验中 Ivy Bridge 处理器使用了单通道内存, 因此这次实验中所使用的 GS464E 处理器核也仅配套使用一个 DDR3-1000 的单通道内存条. 龙芯 3A 芯片虽然使用双通道内存条, 但是其内存路由设计导致运行 memcpy 及 stream 程序时无法使用第二个通道, 因此龙芯 3A 使用双通道内存条的结果等同于其使用单通道内存条的结果. 从这个实验可以看出, GS464E 处理器核在进行内存拷贝时可以利用 70% 以上的峰值内存带宽, Ivy Bridge 的处理器主频与内存频率之比更大, 理论上它应该可以获得更高的带宽占比, 但实际上其表现并没有明显优于 GS464E.

SPLASH2 的测试结果如图 6 所示. 图中纵坐标表示的是 GS464E 处理器核相比于龙芯 3A 芯片的性能提升百分比. SPLASH2 程序平均性能提升了 199.7%, 平均数算法为几何平均. SPLASH2 程序中有大量的内存同步操作, 也即原子操作, GS464E 处理器核在 SCache 中为原子操作所加入的特殊设计有效地提升了该程序的性能.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

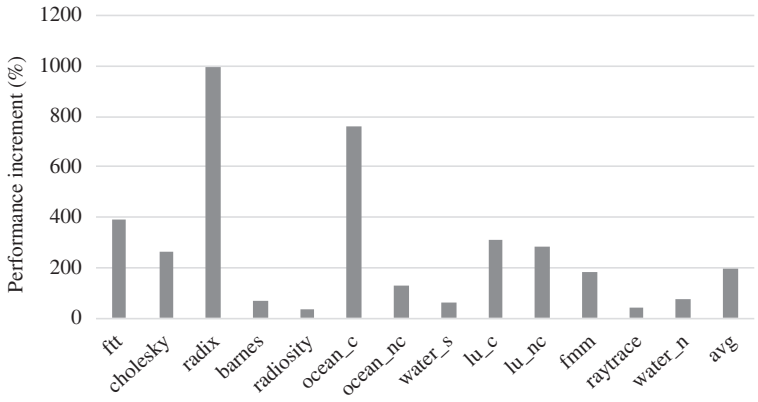


图 6　SPLASH2 测试结果

Figure 6　Performance increment of SPLASH2

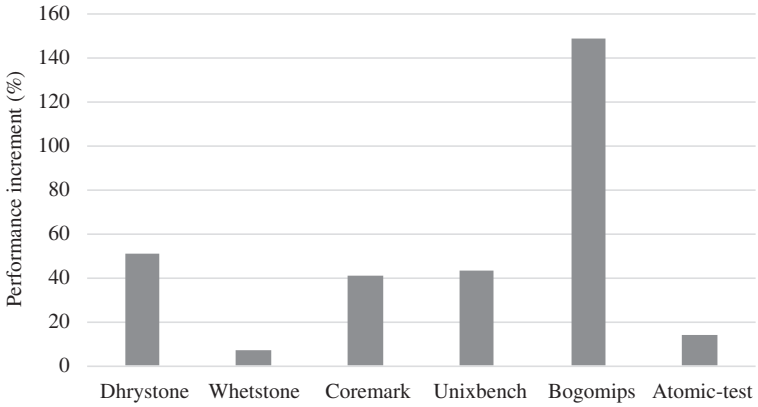


图 7　其他程序测试结果

Figure 7　Performance increment of other testbench

表 4　GS464E 与 Ivy Bridge 结果比较

Table 4　Result comparison for GS464E and Ivy Bridge

Processor	Whetstone (WMIPS/MHz)	Coremark (CMark/MHz)	Dhrystone (DMIPS/MHz)
GS464E	1.78	5.17	2.52
Ivy Bridge	1.51	5.44	5.42

性能评估时使用的其他程序的性能评估结果如图 7 所示. 纵坐标表示 GS464E 处理器核相对于龙芯 3A 芯片的性能提升. 分支指令较多的 Dhrystone 程序, 以及有少量访存操作的 Coremark 与 Unixbench 都有 40% 以上的性能提升, 而运算类测试程序也有少量性能提升. Bogomips 的分数则因为取指机制的优化和定点部件 forward 的加入, 从而提升了 150%.

其中一些的运算类程序在 Intel Ivy Bridge 处理器上也进行了测试, 其与 GS464E 的对比结果见表 4. 其中 Dhrystone 程序性能相对低下是因为 gcc 编译器在编译该程序中字符串比较的函数时, 生成了一些低效率代码. 通过手动修改汇编代码, 修改后的程序在 GS464E 处理器核上可以达到 3.18DMIPS/MHz 的分数. Ivy Bridge 处理器在 Dhrystone 程序上得分较高的主要原因也与编译器有

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

关, 是因为 gcc 直接使用了 SSE4 向量指令来进行字符串比较. 表中的 3 个测试程序规模较小, 都可以在一级数据 Cache 中命中, 因此主要测试的是处理器核流水线的设计水平. 测试程序的结果都是每 MHz 主频下运行的循环次数, 虽然因工艺差距及物理设计水平的差距, GS464E 无法达到类比于 Ivy Bridge 的主频, 不过通过这个结果依然可以看出 GS464E 的流水线设计水平可与世界先进水平相比.

经过性能评估可以发现, 相比于龙芯 3A 芯片, 新设计的 GS464E 处理器核在访存密集型程序上性能提升明显, 尤其是流式访存程序性能提升超过 10 倍. 除此之外, 各类定点、浮点测试也都有可观的性能提升. 当与 Intel Ivy Bridge 这样的世界先进处理器相比较时, GS464E 的执行效率也相差不多, Whetstone 程序每 MHz 主频的运行次数以及 stream copy 程序的带宽相比于内存总带宽的比例已经超过了 Ivy Bridge.

5 总结

GS464E 处理器核是一款高性能的处理器核架构, 拥有强大的运算能力和很高的访存带宽. 相比于之前的处理器核, 本次设计重点强化了分支预测和访存流水线, 消除了几个影响性能的关键延迟; 大幅度地提高了处理器核内部各项队列的项数, 提高了指令的并行度; 加入了激进的数据预取, 重点提高了流式访存的性能; 优化了各级片内缓存, 提高缓存命中率. 在 GS464E 的处理器设计中, 采用了多项创新, 包括基于缓存失效队列的预取引擎设计、可适用于多核情况的存储填充机制, 以及基于缓存一致性的硬件抗别名设计, 一些部件的设计也参考了业界公认的设计方案, 如使用 BrBTB 消除了分支指令之后的取指空泡, 使用双访存部件设计, 采用双重 TLB 设计等.

经过上述设计, GS464E 处理器核的性能比上一代处理器核产品有大幅提升, 在流式访存程序上, 更是获得了超过 10 倍的性能提升. 从程序运行效率的角度, 其结果已不输于 Intel Ivy Bridge 处理器. 总体而言, GS464E 的设计已经接近国际最先进水平, 是一款国内顶尖的拥有自主知识产权的处理器核产品.

参考文献

1 Hu W W, Zhang F X, Li Z S. Microarchitecture of the Godson-2 processor. J Comput Sci Technol, 2005, 20: 243–249
2 Hu W W, Wang J, Gao X, et al. Godson-3: A scalable multicore RISC processor with x86 emulation. IEEE Micro, 2009, 29: 17–29
3 Gao X, Chen Y J, Wang H D, et al. System architecture of Godson-3 multi-core processors. J Comput Sci Technol, 2010, 25: 181–191
4 Hu W W, Wang R, Chen Y J, et al. Godson-3B: A 1GHz 40W 8-core 128GFLOPS processor in 65nm CMOS. In: Proceedings of IEEE International Solid-State Circuits Conference Digest of Technical Papers. San Francisco: IEEE, 2011. 76–78
5 Hu W W, Zhang Y F, Yang L, et al. Godson-3B1500: A 32nm 1.35 GHz 40W 172.8 GFLOPS 8-core processor. In: Proceedings of IEEE International Solid-State Circuits Conference Digest of Technical Papers. San Francisco: IEEE, 2013. 54–55
6 Hu W W, Yang L, Fan B X, et al. An 8-core MIPS-compatible processor in 32/28 nm bulk CMOS. IEEE J Solid-State Circ, 2014, 49: 41–49
7 Sinharoy B, Kalla R, Starke W J, et al. IBM POWER7 multicore server processor. IBM J Res Dev, 2011, 55: 1–29
8 Damaraju S, George V, Jahagirdar S, et al. A 22 nm IA multi-CPU and GPU system-on-chip. In: Proceedings of IEEE International Solid-State Circuits Conference Digest of Technical Papers. San Francisco: IEEE, 2012. 56–57
9 Qi Z C, Liu H, Li X K, et al. Design for testability features of Godson-3 multicore microprocessor. J Comput Sci Technol, 2011, 26: 302–313

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

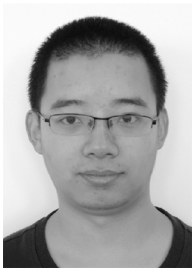
Design of Loongson GS464E processor architecture

WU RuiYang[1,2,3*], WANG WenXiang[1,2], WANG HuanDong[4] & HU WeiWu[1,2,4]

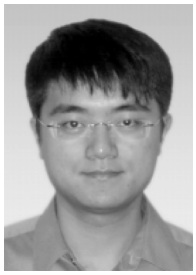
1 *State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190*, China*;
2 *Insitute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190*, China*;
3 *University of Chinese Academy of Sciences, Beijing* 100049*, China*;
4 *Loongson Technology Co. Ltd., Beijing* 100190*, China*
*E-mail: wuruiyang@ict.ac.cn

Abstract Loongson GS464E is the most recent processor architecture introduced by Loongson Technology. In this paper, we describe the key aspects of the GS464E architecture. Compared to the previous GS464 architecture, GS464E focuses on improving the performance of memory access and branch prediction, using enlarged queues, caches, and TLBs in the processor. Support for the MIPS DSP instruction set and virtualization is also provided. Specifically, the memory subsystems have a 3-level cache hierarchy each with an LRU replacement policy, and also support multi-processor cache coherence. With the aforementioned optimization features, GS464E has become an innovative, high-performance processor architecture.

Keywords processor core, multi-core processor, branch prediction, performance of memory access, Cache coherence



WU RuiYang was born in 1991. He is currently a Ph.D. candidate in the Institute of Computing Technology of the Chinese Academy of Sciences, Beijing. His main research interests include high-performance computer architecture and multi-core processor architecture.



WANG WenXiang was born in 1983. He received his Ph.D. degree from the Institute of Computing Technology of the Chinese Academy of Sciences, Beijing, in 2012. His main research interests include high-performance computer architecture and microprocessor architecture.

WANG HuanDong was born in 1982. He received his Ph.D. degree from the Institute of Computing Technology of the Chinese Academy of Sciences, Beijing, in 2011. His main research interests include multi-core processor architecture, on-chip network, memory, and IO systems.

HU WeiWu was born in 1968. He received his Ph.D. degree from the Institute of Computing Technology of the Chinese Academy of Sciences, Beijing, in 1996. He is currently a professor and Ph.D. supervisor in the Institute of Computing Technology. His research interests include high-performance computer architecture, parallel processing, and VLSI design.

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net