通用处理器的高带宽访存流水线研究

张 浩 林 伟 周永彬 叶笑春 范东睿
（中国科学院计算技术研究所 北京 100190）

摘 要 存储器访问速度的发展远远跟不上处理器运算速度的发展，日益严峻的访存速度问题严重制约了处理器速度的进一步发展. 降低 load-to-use 延迟是提高处理器访存性能的关键，在其他条件确定的情况下，增加访存通路的带宽是降低 load-to-use 延迟的最有效途径，但增加带宽意味着增加访存通路的硬件逻辑复杂度，势必会增加访存通路的功耗. 文中的工作立足于分析程序固有的访存特性，探索高带宽访存流水线的设计和优化空间，分析程序访存行为的规律性，并根据这些规律性给出高带宽访存流水线的低复杂度、低延迟、低功耗解决方案. 文中的工作大大简化了高带宽访存流水线的设计，降低了关键路径的时延和功耗，被用于指导 Godsonx 处理器的访存设计. 在处理器整体面积增加 1.7%的情况下，将访存流水线的带宽提高了一倍，处理器的整体性能平均提高了 8.6%.

关键词 高带宽；访存流水；高速缓存；TLB
中图法分类号 TP302 **DOI 号**：10.3724/SP.J.1016.2009.00142

High-Bandwidth Memory Accessing Pipeline of General Purpose Processor

ZHANG Hao LIN Wei ZHOU Yong-Bin YE Xiao-Chun FAN Dong-Rui
(*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

Abstract There is a near-exponential increase in processor speed and memory capacity. However, memory latencies have not improved as dramatically, and access times are increasingly limiting system performance. Low load-to-use latency is a key to approach high memory performance, and increasing the bandwidth of memory pipeline always works. But high bandwidth brings more complexity and needs more power. The authors' work was based on the analysis of the applications, and intend to find the head room of the performance of the memory pipeline. The authors find some useful characters of memory operations was found and give an optimized design of high bandwidth memory pipeline, which has low complexity, low latency and low power. The decisions are used to instruct the design Godsonx processor, although the bandwidth of memory access is doubled and the performance is increased by 8.6%, the extra area is only 1.7% of the original design.

Keywords high bandwidth; memory pipeline; cache; TLB

1 引 言

根据摩尔定律，微处理器的速度每十年增长十倍以上，而常用存储器(DRAM)的访存时间却只是缩短到原来的一半. 因此，相对于处理器来说，访存延迟以每十年 5 倍的速度增长. 从 1986 年到 2000 年，处理器速度的年增长率为 55%，而存储器速度的年增长率仅为 10%，按照这种速度推算可知，处理器与存储器以及通信性能上的差距不断增加，

收稿日期：2006-11-27；最终修改稿收到日期：2008-07-17. 本课题得到国家自然科学基金(60736012)、国家"九七三"重点基础研究发展规划项目基金(2005CB321600)资助. **张 浩**，男，1980 年生，博士，主要研究方向为体系结构、片上网络交换路由等. E-mail：zhanghao@ict.ac.cn. **林 伟**，男，1979 年生，博士研究生，研究方向为大规模并行体系结构及处理器实现. **周永彬**，男，1981 年生，博士研究生，研究方向包括片上网络、片上存储系统以及并行程序在众核处理上的优化. **叶笑春**，男，1981 年生，博士研究生，研究方向包括为高性能计算机体系结构、算法并行与优化. **范东睿**，男，1979 年生，博士，副研究员，研究方向是高性能计算技术及先进处理器体系结构.

造成了系统结构的失衡，形成了阻碍性能提升的“内存墙”[1-2]. 日益严峻的访存速度问题正在成为影响处理器性能的瓶颈，严重制约了处理器速度的进一步发展.

降低 load-to-use 延迟是提高处理器访存性能的关键，高速缓存 Cache 的引入就是为了缓解这一问题[3-4]，通过位于处理器主流水线和存储器之间的这层存储逻辑，处理器可以在大多数情况下避免直接访问内存，提高了访存的性能，延缓了“内存墙”问题. 目前几乎所有的主流通用处理器都在流水线中包含高速缓存逻辑，并且高速缓存逻辑往往被作为独立的流水级. 在其他条件确定的情况下，增加访存通路的带宽是降低 load-to-use 延迟的最有效途径，但增加带宽意味着增加访存通路的逻辑复杂度，并且访存通路上的 RAM 逻辑都需要提供双端口. 这些复杂度上的增加势必会增加访存通路的时延和功耗.

本文的工作立足于分析程序固有的访存特性，探索高带宽访存流水线的设计和优化空间. 通过分析总结出程序访存行为的规律性，并根据这些规律性给出高带宽访存流水线的低复杂度、低延迟、低功耗的高带宽访存解决方案. 本文的工作大大简化了高带宽访存流水线的设计，降低了关键路径的时延和功耗，被用于指导 Godsonx 处理器的访存设计(Godsonx 是基于 Godson-2[5] 设计的 X86 处理器). 采用优化高带宽访存设计后，相对原始的宽度为 1 的访存流水线，处理器的整体面积仅增加了 1.7%，处理器的平均性能提高了 8.6%.

本文首先介绍高带宽访存流水线的相关研究，然后介绍试验环境，并分别给出 Cache、TLB 的优化策略，评价了改进后的性能提升，最后对全文工作进行总结.

2　相关研究

处理器设计者对性能的要求是无止境的，对指令级并行性的追求也不会停止，而片上存储系统是影响指令并行性开发的关键，因此对片上存储系统性能的要求也会持续. 程序中大约有三分之一的指令是访存指令或者涉及访存操作[6]，对于 4 发射的超标量处理器来说，具有平均每拍处理两条访存操作的能力是必需的. 面对这样的需求，在处理器中采用高带宽的访存流水线非常必要. 这就要求采用多端口的 Cache. Cache 位于流水线的关键路径上并且逻辑复杂[7]，因此有必要研究一种简化的多端口 Cache，同时满足对 Cache 带宽的要求.

目前，多端口 Cache 的设计方法主要有 4 种[8]：传统的复杂的理想多端口 Cache、分时访问的多端口 Cache、多份复制的单端口 Cache 组成的多端口 Cache 以及采用多体交叉策略的多端口 Cache. DEC Alpha 21264 采用的是分时多端口 Cache 设计技术[9]，Cache 的工作频率是其他流水级频率的两倍，在主流水线的一拍内可以先后处理两条访存操作，借此实现双端口的效果. 随着处理器主频的不断提高，SRAM 已经成为流水线时延的关键路径，这种策略已被淘汰. DEC Alpha 21164[10] 采用的是由多份复制的单端口 Cache 组成的多端口 Cache，使用两个内容完全相同的单端口 Cache 实现了双端口 Cache. MIPS R10000 采用了双体的 DCache[3]. 两个同时被处理的访存操作必须对应不同的体. 对于比较平衡和调度良好的访存操作流，这种策略可以提供良好的访存并行性和较高的带宽，但如果访存操作流并不平衡，频繁发生的体冲突将严重影响访存的性能. 虽然各项多端口 Cache 设计技术都有其缺点，多体交叉技术仍然是应付不断提高的指令级并行性的关键方法，目前主流的微处理器都倾向于采用这种策略[3,11].

在主流的处理器的设计中，快速地址转换部件 TLB(Translation Look aside Buffer)作为访存部分的核心部件，是流水线中不可或缺的一环[5,12-13]. 为了降低访存延迟，主流处理器一般将 Cache 和 TLB 放在相同的流水级，采用虚地址索引的 Cache，访存操作同时访问 Cache 和 TLB[14-15]，所以高带宽的访存流水线设计同样会对 TLB 的设计产生影响.

随着多端口 TLB 在先进通用处理器中的应用，由高带宽的访存流水线带来的虚实地址转换压力得到了缓解，研究人员开始把越来越多的注意力放在多端口 TLB 的设计和优化上. 文献[16]给出了一种 L0 TLB 的设计策略，整个 L0 TLB 只有一项，利用连续访存操作的页面局部性，简单的 TLB 结构可以同时处理多个访存操作的虚实地址转换，文献[17]中也介绍了相似的设计. 文献[18]中介绍了 selective filter-bank TLB，使用几个较小的缓冲区来处理大多数的访问，用以降低 TLB 的延迟. Yeager Kenneth 介绍了另一种 TLB 结构，TLB 中的一项被几个连续的页面公用，这种设计思路和 MIPS 处理器类似[19]. Interleaved TLB 使用互连结构，根据不同的地址将多个虚实地址转换请求分配到多个 bank[20].

Cache、TLB 是通用处理器访存流水线中最为核心的两个部件，这些部件的设计直接影响访存流水线的性能，后面会分别从单个部件的角度对高带宽访存流水线的优化进行研究.

3 实验环境

本文的研究工作以 Godsonx 模拟器作为平台. Godsonx 是一款 4 发射的超标量 X86 处理器. 因为 X86 指令集缺乏结构寄存器[21]，X86 程序中的很多临时数据往往需要借助内存来存放，并且参数传递大量依赖栈来完成，这些导致 X86 处理器的访存操作非常密集[6]，而 Godsonx 的原始设计中访存带宽为 1，所以设计高带宽的访存通路对 Godsonx 来说尤其重要.

Godsonx 模拟器是基于 Godson2 模拟器实现的，是一个完全兼容 IA-32 指令集的全系统信号级模拟器，详细模拟了真实处理器的所有结构细节，可以看成 RTL 代码的高级语言版本. 表 1 给出了 Godsonx 模拟器的详细配置.

表 1 Godsonx 处理器配置

功能部件	2 个定点部件、2 个浮点部件、2 个访存地址部件
流水线深度	12 级
访存队列	Store 16 项、Load 12 项
Reorder Buffer	32 项
指令窗口	16 项 ALU、16 项访存
TLB	DTLB 64 项、ITLB 32 项
指令 Cache	32KB、8-way、32Byte/line
数据 Cache	64KB、4-way、32Byte/line
片上 L2 Cache	512K、4way、32byte/line
访存延迟	L1：1-cycle、L2：6-cycle、Memory：100-cycle

本文使用 SPEC CPU 2000 程序作为分析对象. SPEC CPU 2000 是用来测试 CPU 性能的标准测试程序组，其分值被认为是衡量处理器性能的一个标准[22]. 本文设计中的时延、面积信息则是基于 GodsonX 的 RTL 代码，使用 Synopsys 的 Design Compiler(2006.06 for amd64)软件完成.

4 高带宽访存流水线的优化

虽然可以通过增加访存流水线的宽度来满足新型处理器对于访存性能的要求，但一味地增加访存流水线的宽度并不能使得处理器的性能线性增加. 对于一个 4 发射的超标量处理器，在 DCache 大小为 32Kbytes 的情况下，增加第 2 个访存端口可以将处理器的性能平均提高 25%. 持续增加端口并不能得到相应的性能提升，增加第 3 个和第 4 个端口只能将处理器的性能分别提高 4%和 1%[23].

4.1 简化的多端口 Cache 设计

为了理解为何设计简化的基于多体策略的多端口 Cache，我们必须了解 Cache 所见到的访存行为的规律. 在本节中，我们分析了程序访存行为的规律，在此基础上提出了我们的设计，并介绍了简化访存队列选择逻辑的技术. 整个设计没有对处理器的性能产生明显影响.

4.1.1 针对 Cache 的访存行为特征分析

程序中被执行的指令有大约 38%是访存指令或直接涉及到访存[6]. 如图 1 所示，对于 SPEC CPU2000 基准测试程序，访存指令在被提交的指令中占很大一部分. 同时，先进处理器中广泛采用的多发射乱序流水线技术也要求采用多端口 Cache 设计.

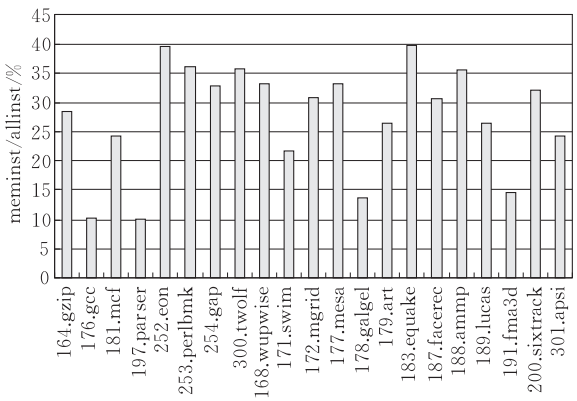

图 1 访存指令在指令序列中所占的比例

从图 2 可以看出，load 和 store 操作间的比例是比较固定的，load 操作的数目大概是 store 操作的两倍. 不考虑 Cache 缺失的情况，每条 store 操作需要先后两次访问 Cache，第 1 次是检查是否命中，第 2 次才真正将数据存入 Cache，而每条 load 操作

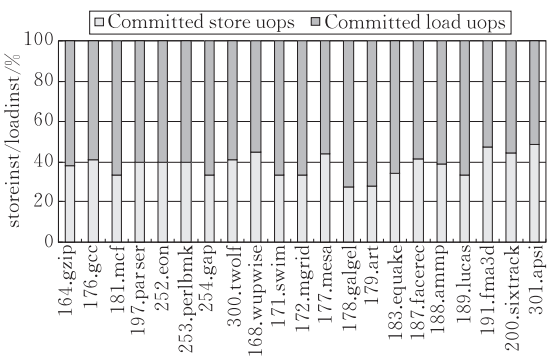

图 2 指令序列中 load 指令和 store 指令的比例

却只需要访问一次 Cache. 因此，我们可以得出结论：store 操作访问 Cache 的次数接近 load 操作访问 Cache 的次数.

4.1.2　简化的多端口 Cache 结构

DCache 采用多体策略，每拍允许两条访存操作同时访问，这两条访存操作可以使 load 和 store 任意组合，但前提是这两条访存操作不能对应相同的体，这一点是由访存队列保证的.

因为先进高性能处理器采用的超标量技术需要多端口 Cache，并且大于 2 的 Cache 端口数并不能带来明显的性能提升，我们将分析的重点定在设计双端口的 Cache. 由于 load 和 store 访问 Cache 的次数一样多，简化的双端口 Cache 技术将 load 和 store 分开来处理. 该 Cache 提供两个端口，一个被 load 使用，一个被 store 使用，load 和 store 各自使用自己的 Cache 端口，互不干扰. Load 和 store 的选择逻辑在 LSQ 中，每拍同时选取一条 load 和一条 store.

图 3 给出了简化双端口 Cache 设计的细节. 当 store 为检查是否命中而第 1 次访问 Cache 时，只有 Tag SRAM 被访问，并不访问 Data SRAM；相反，当 store 为存储数据而第 2 次访问 Cache 时，只有 Data SRAM 被访问. Load 操作同时访问 Tag SRAM 和 Data SRAM. 我们采用双端口的 Tag SRAM 和单端口的多体 Data SRAM 组成双端口 Cache，同时处理一条 load 和一条 store. 如果 LSQ 中的选择逻辑选出的是一条 load 和一条第 1 次访问 Cache 的 store，这两条操作可以分别被发送到各自的 Cache 端口；如果一条 load 和一条第 2 次访问 Cache 的 store 同时被 LSQ 选中，则必须检测是否发生了体冲突；如果发生体冲突，store 被取消，如果没有发生体冲突，这两条操作仍然可以分别被发送到各自的 Cache 端口. 用于体冲突检测的逻辑位于选择逻辑之后，这意味着首先选出一条 load 和一条 store，然后判断是否发生体冲突.

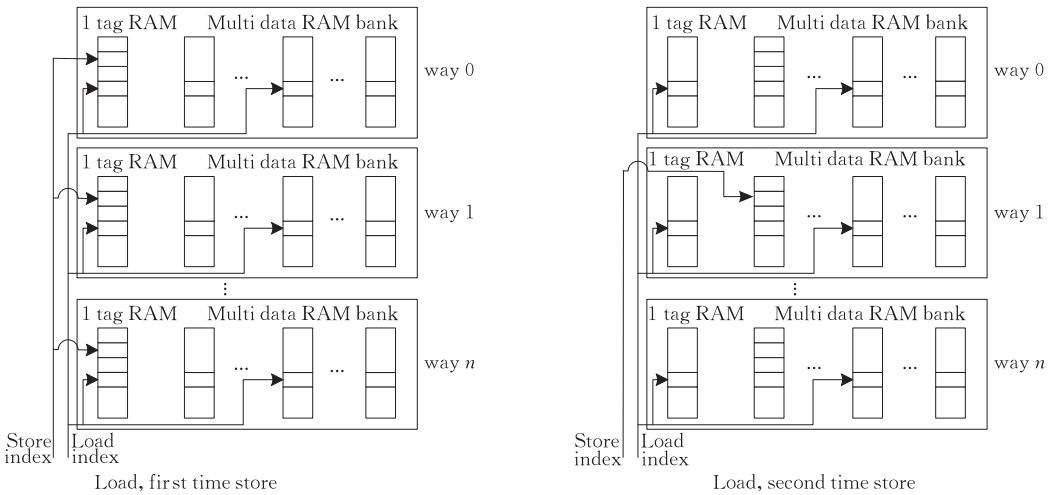

图 3　简化的双端口 Cache

4.1.3　LSQ 选择逻辑的时延

对于 LSQ 来说，由于分别选择一条 load 和一条 store，可以使用分离的 Load 队列和 store 队列取代统一的 LSQ，这使得 LSQ 的发射选择逻辑被简化，这一点可以从 LSQ 选择逻辑的延迟上看出来. 表 2 给出了 LSQ 选择逻辑的延迟对比，包括原始设计中 LSQ 选择逻辑的延迟以及采用简化双端口 Cache 设计后 LSQ 选择逻辑的延迟. LSQ 中每一项都包括 64 位的数据、64 位的地址以及大于 20 位的其他控制信息. LSQ 选择逻辑的时延从 1.24ns 下降到 1.03ns，减少了 16.9%. LSQ 位于流水线的关键路径上，所以降低 LSQ 的时延非常重要，如果 LSQ 一级的延迟是流水线各级中最长的，在理想情况下处理器的频率可以得到 20.4%的提升.

表 2　LSQ 发射选择逻辑的延迟

LSQ 选择逻辑	延迟时间/ns
原始设计	1.24
简化设计	1.03

4.2　单端口的 TLB 设计

在本节中，我们首先要回答这样一个问题：为什么在高带宽访存流水线中可以用单端口 TLB 来取代多端口 TLB？我们通过对程序的访存行为进行分析，从访存流水线中各模块的角度总结程序访存行为的规律性. 通过这些分析可以证明，在高带宽访存流水线中采用单端口 TLB 符合访存行为的规律，

下面将详细介绍新型的单端口 TLB 设计及优化技术.

4.2.1 针对 TLB 的访存行为特性分析

首先分析访存行为的页面级局部性.通过在 Godsonx 模拟器环境上统计对每个页面的平均访问次数,结果如图 4 所示.可以很明显地看出,访存操作在页面一级具有很好的局部性,对于 1 亿条连续执行的指令,平均一个页面被访问 60700 次,其中最少的程序平均每个页面被访问 13242 次,最多的程序平均每个页面被访问 98577 次.

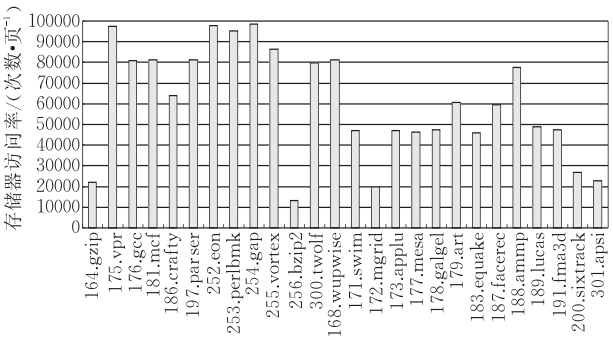

图 4 每页平均被访问的次数

虽然从整体看来程序访存行为有很好的页面局部性,但连续的访存操作序列并不具有明显的页面局部性.图 5 给出了对这一情况的详细分析结果,n 表示在程序顺序上连续访问同一页面的访存操作的数目,n 大于 8 的情况下的访存操作仅占访存操作总数的 40%左右.

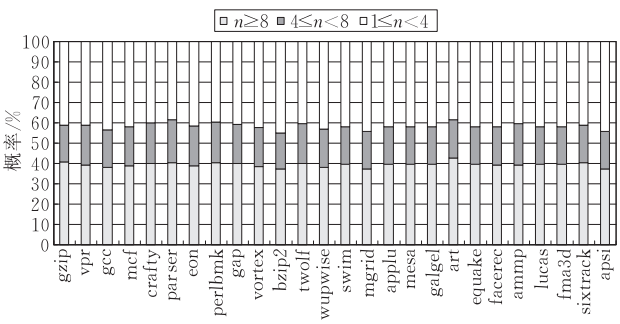

图 5 程序顺序上连续访问统一页面的概率

我们可以通过图 6 中的例子来说明上述现象,图 6 中的代码片段取自数组复制程序.

```
unsigned char array_src[4096] _attribute__((aligned(4096)));
unsigned char array_dst[4096] _attribute__((aligned(4096)));
…
for (int i=0; i<4096; i++) {
  array_dst[i]=array_src[i];
}
```

图 6 数组复制的代码片段

Unsigned char 数据类型的大小为 1Byte,源数组和目标数组都有 4096 个元素,大小都是 4KByte,在页面大小为 4KByte 的情况下,这两个数组都是页面边界对齐的(_attribute__((aligned(4096)))),所以两个数组分别独立占据两个不同的页面.在复制的过程中,从程序顺序上看,两个数组对应位置上的数据被交替访问,由于两个数组占据不同的页面,导致连续访问相同页面的访存操作不超过 1 个,也就是说在数组复制的过程中 n 始终为 1.即使可以由编译器进行循环展开优化,由于寄存器个数的限制,这种情况仍然存在.

然而,这种情况在乱序执行的超标量处理器中发生了变化.主流的超标量处理器广泛使用 LSQ,用于在 ROB 之外对访存操作进行管理和调度,每条进入乱序流水线的访存指令都在 LSQ 中占据一项.由于主流超标量处理器通常使用大指令窗口开发指令集并行性(ILP),LSQ 的容量也相应增大,可以同时维护多条访存操作.例如,Godson2E 的 LSQ 有 16 项[5],AMD Opteron 的 LSQ 有 36 项[11],IBM 的 Power4 的 LSQ 有 64 项[24].

相比在程序顺序上选取多个访问相同页面的连续访存操作,在 LSQ 中选择多个对应相同页面的访存操作就容易得多.在一个 4 发射、访存流水线的宽度为 2 的超标量处理器中,对于容量为 24 项的 LSQ,在其中有多于两个满足发射条件的访存操作时,从中随机选取两个访存操作,则这两个操作对应同一页面的概率如图 7 所示,这个概率对每个应用程序来说都超过 95%,平均概率达到 96.3%,本文的优化策略正是基于这一发现.

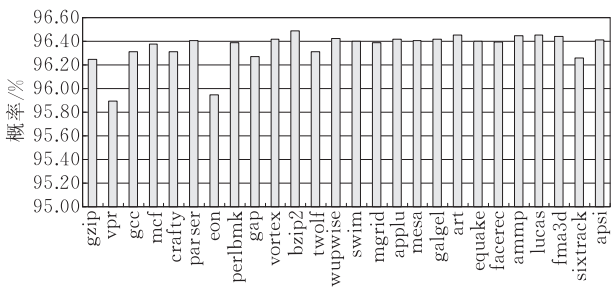

图 7 在 LSQ 中随机选择的两个访存操作对应相同页面的概率

实现多端口 TLB 对高带宽访存流水线是必要的,但是一味地增加访存流水线的宽度并不能使处理器的性能呈线性增长;采用乱序流水线的高性能的超标量处理器需要多端口 TLB,但如果端口数多于 2,并不能得到明显的性能提升,因此在接下来的讨论

中，我们将研究的重点放在双端口的 TLB 设计.

4.2.2　传统的双端口 TLB 设计

在介绍优化的虚实地址转换策略之前，本节首先介绍传统双端口 TLB 的结构和实现代价. 这种传统的 TLB 包括一个双端口的 CAM 和两个单端口的 RAM，如图 8 所示. 在这种结构中，每个时钟周期最多处理两个访存操作，这两个访存操作可以是任意的满足发射条件的 load 和 store 操作的组合. 这两个访存操作由位于 CAM 之前的 LSQ 中的选择逻辑选出，然后 CAM 被这两个访存操作并行访问，通过各自的虚拟地址查询 CAM，得到两个相应的索引，然后根据这两个索引分别对两个 RAM 进行访问，得到各自对应的物理地址. 这两个 RAM 的内容是完全一致的，在发生 TLB 缺失时，TLB 回填逻辑将新的虚实地址映射关系同时写入两个 RAM 中的相同位置，保证两个 RAM 的内容时刻保持一致，这实际上是一种通过多体复制来实现多端口 RAM 的策略. 由此可知，CAM 的读端口必须是两个，但写端口只需要一个.

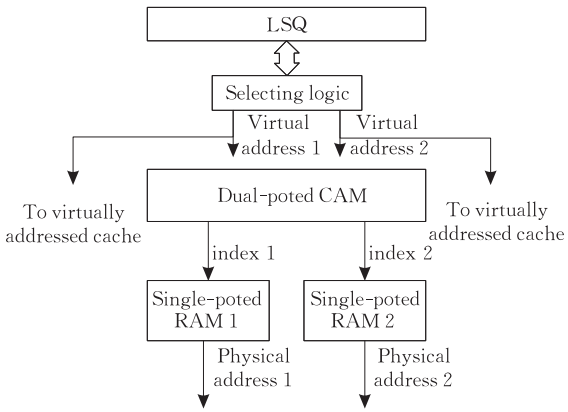

图 8　传统的双端口 TLB 设计

这种传统的双端口 TLB 设计实现简单，并且命中率较高，通过多个内容相同的单端口 TLB 来实现多端口 TLB 也不会导致 TLB 的体(bank)冲突[20]，但这种设计会带来功耗和面积的开销，这是该设计的主要缺陷. 传统双端口 TLB 的各部分面积分布如图 9 所示，在本章的后续各节中，主要研究通过结构优化减少 TLB 的面积和访存关键路径上的延迟.

4.2.3　利用页面局部性减少面积

通过前面对原始双端口 TLB 面积的分析可知，如果我们用一个由单端口 CAM 和一个单端口 RAM 组成的单端口的 TLB 来取代原始的双端口 TLB，TLB 的面积可以被大幅减少. 根据这一点可以对 LSQ 的发射选择逻辑进行修改，使之在满足发

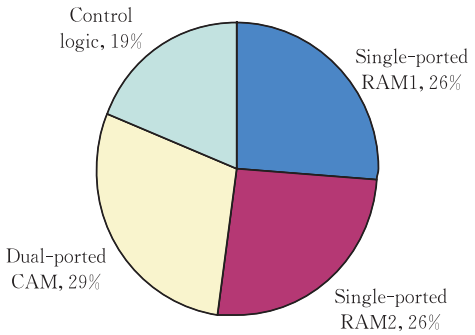

图 9　原始双端口 TLB 设计中的各部分面积比例

射条件的访存操作中选取两个对应相同页面的操作，使用这两个操作中任意一个的虚拟页面号对单端口 TLB 进行检索，得到的物理页面号被这两个访存操作共享，这就是伪多端口 TLB 的设计思路，其结构如图 10 所示，图 11 给出了改进后伪双端口 TLB 的各组成部分的面积分布.

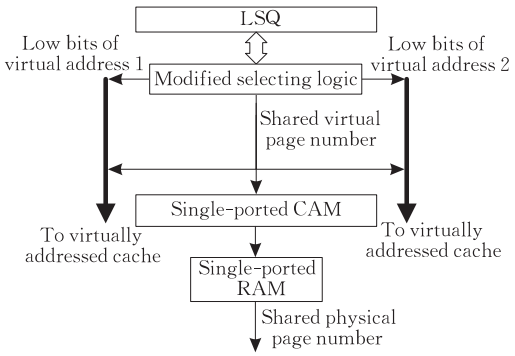

图 10　修改后的单端口 TLB 设计

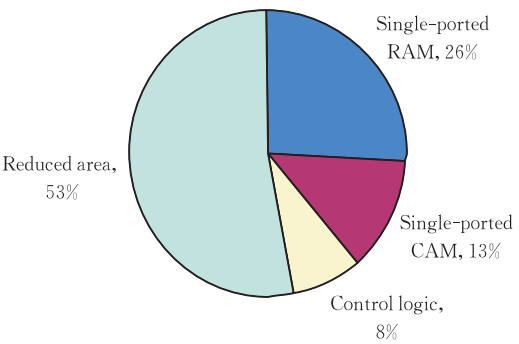

图 11　单端口 TLB 的各部分面积分布

LSQ 的选择逻辑从所有满足发射条件的访存操作中选择两个对应相同虚拟页面的操作，使用这一共同的虚拟页面号对 CAM 进行全相联检索，从中得到用于访问 RAM 的索引，然后使用这个索引从 RAM 中取出这两个访存操作共有的物理页面号. 优化后的 LSQ 发射选择逻辑在将虚拟页面号送到 CAM 的同时，两个访存操作的虚拟地址的页内偏移同时被送出，用于访问数据 Cache. 由于虚实页

面的起始都是在页边界对齐，虚拟地址的页内偏移和从 RAM 中取到的物理页面号拼接，得到各访存操作对应的物理地址.

通过上述方法，我们使用简单的单端口的 TLB 取代了原始的双端口 TLB，代价是增加了 LSQ 发射选择逻辑的复杂度，在原始的双端口 TLB 设计中，LSQ 的发射选择逻辑仅需要从所有满足发射条件的访存操作中选出任意两个，而新的 LSQ 发射选择逻辑中增加了对访存页面的限制，增加了访存关键路径的延迟. 改进后 TLB 的面积减少了 53%，如图 11 所示.

4.2.4 利用随机选择策略降低延迟

虽然改进后的设计减少了 TLB 的面积，但 LSQ 发射选择逻辑相比原始的设计变得更加复杂，LSQ 模块的延迟从 1.03ns 增加到 1.24ns，如表 3 所示. 导致 LSQ 模块延迟增加的原因是第 2 个访存操作的选择依赖于选出的第 1 个访存操作，因为第 2 个访存操作必须对应于第 1 个访存操作相同的虚拟页面号，图 12 给出了这种 LSQ 发射选择逻辑的详细结构，由于对两个访存操作的选择直接存在依赖，导致 LSQ 的延迟变长，而 LSQ 本来就是访存流水线的关键路径，这对处理器的主频有很大影响.

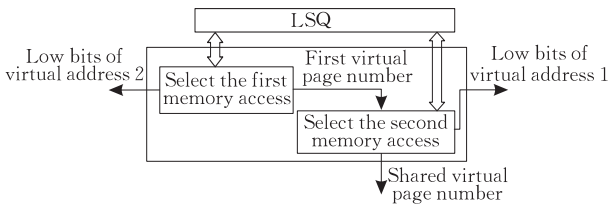

图 12 修改后的 LSQ 选择逻辑

表 3 各种 LSQ 选择逻辑的时延

LSQ 设计方案	延迟时间/ns
图 8 的设计	1.03
图 12 的设计	1.26
图 13 的优化设计	1.04

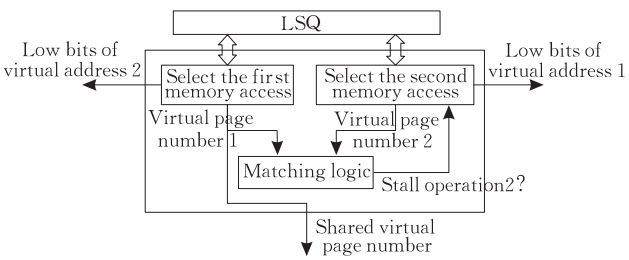

图 13 优化后的 LSQ 选择逻辑

为了降低 LSQ 发射选择逻辑的延迟，我们对访存流水线的结构进行了进一步优化，此次优化的依据是前面对程序的访存行为进行观察所得的结论，即在 LSQ 中随机选择两个满足发射条件的访存操作，这两个操作对应相同虚拟页面的概率很高，如图 7 所示. 因此，LSQ 选择逻辑可以随机从满足发射条件的访存操作中选出两个，这样选择第 1 个访存操作和选择第 2 个访存操作之间的依赖被削除，二者可以独立地被选择.

虽然在 LSQ 中随机选出的两个满足发射条件的访存操作对应相同页面的概率很大，但二者对应不同页面的可能性依然存在. 为了保持设计的完备性，在 LSQ 中两个独立的发射选择模块之后，增加了页面号匹配逻辑. 这个匹配逻辑对比两个被选出的访存操作对应的虚拟页面号，如果相同，则共同的虚拟页面号被送到单端口 TLB，检索相应的物理页面号；如果不同，则取消第 2 个访存操作，只处理第 1 个访存操作，这个被取消的访存操作仍然在 LSQ，可以被 LSQ 的发射选择逻辑重新选取. 在 LSQ 中满足可发射条件的访存操作不多情况下，如果只有一个访存操作被发射选择逻辑选出，则页面号匹配逻辑只需要将第 1 个访存操作送出，相当于不进行任何处理. LSQ 发射选择逻辑进一步改进后的结构如图 13 所示.

从图 13 中可以看出，在匹配逻辑对两个访存操作的虚拟页面号进行对比的同时，第 1 个访存操作的虚拟页面号就被作为两个访存操作共有的虚拟页面号送到输出总线上，所以 LSQ 有效输出的延迟仅仅依赖第 1 个访存操作的选取，这与原始的 LSQ 相似，同原始设计相比并不会增加 LSQ 发射选择逻辑的延迟，如表 3 所示.

通过上述分析，最终的优化设计比起原始的设计大大减少了 TLB 的面积，并且 LSQ 发射选择逻辑的延迟并没有明显改变. 这种设计的思路源自我们前面对程序访存行为的分析，即这样的改进不会明显影响处理器的性能，接下来我们讨论这种设计对处理器性能的影响.

5 结果分析

改进前后的 Godsonx 访存流水线结构如图 14、图 15 所示.

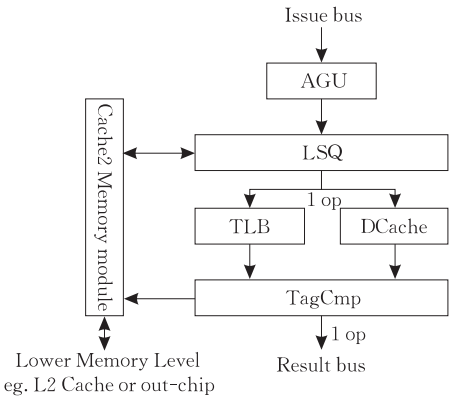

图 14　改进前的 Godsonx 访存流水线

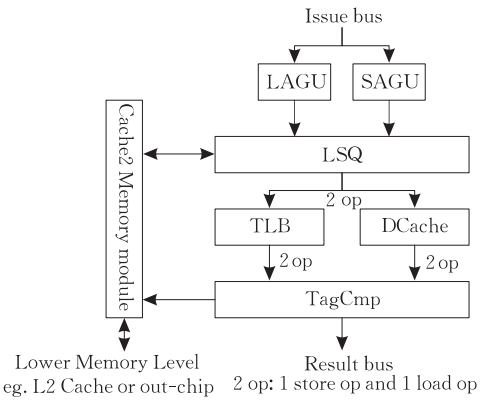

图 15　改进后的 Godsonx 访存流水线

5.1　相对单通路访存流水线的面积改变

改进设计后的面积变化主要体现在还有因访存通路变宽引起的新增加的模块及控制逻辑，还有因访存通路变宽引入的逻辑. 因为采用了简化的双端口 Cache 设计，仅有 Tag SRAM 因为采用双端口设计而增加了面积，占据 Cache 绝大部分面积的 Data SRAM 却仍然是单端口的，所以整个 DCache 的面积变化不大. 改进设计前后 Godson2 处理器的面积对比如图 16 所示，可以看出增加访存通路的带宽后，各模块的面积分布并没有明显的变化(图 16).

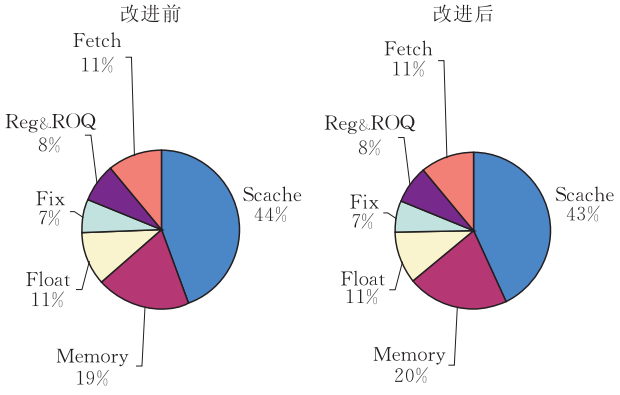

图 16　改进前后各模块的面积分布

在发射宽度为 2 的访存流水线中，TLB 仍然采用单端口设计，仅在 LSQ 中增加了两个访存操作的地址匹配逻辑，并没有显著增加设计的复杂度和面积.

由于采用双端口 Tag RAM 以及增加带宽引入的控制逻辑，使 Cache 部分的面积比原始的单端口设计增加了 6%. 相对于发射宽度为 1 的流水线，在增加访存带宽之后，整个处理器面积只有 1.7% 的增加.

5.2　相对单通路访存流水线的性能提升

按照前面的分析，在 LSQ 中相互独立地选取一条 load 和一条 store 可能会导致体冲突，实验表明，这种情况很少出现. 体冲突的频率如图 17 所示，平均频率为每 10000 拍 4.5 次.

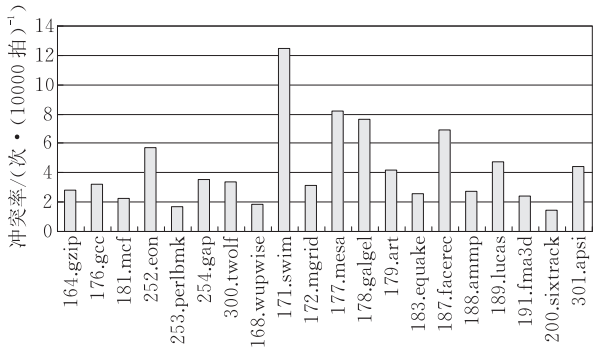

图 17　选择一条 load 和一条 store 时的体冲突频率

同传统高带宽访存流水线相比，虽然我们简化了 LSQ 中的选择逻辑，处理器的整体性能并没有受到显著影响. 图 18 给出了 SPEC CPU2000 在采用简化双端口 Cache 与采用传统基于多体设计的双端口 Cache 时的 IPC 对比，我们可以看到 IPC 的变化并不明显. 大部分程序的 IPC 降低不足 0.5%，平均下降 0.3%.

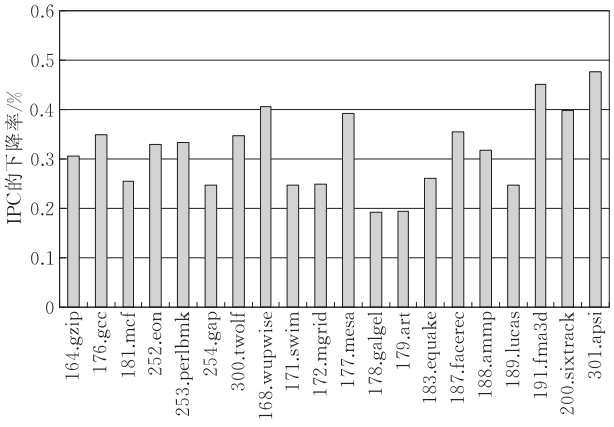

图 18　简化前后设计后的 IPC 变化

改进访存通路后的性能表现如图 19 所示,同增加带宽前相比,所有程序的性能均有明显提高,IPC 平均提高了 8.6%,其中最高的 galgel 提高了 11.7%,最少的 fma3d 提高了 6.2%. 可见在 GodsonX 的原始设计中,每拍只能处理一条访存操作的较窄访存通路是限制其性能的关键因素,必须提高访存通路的带宽,平衡整个处理器的设计.

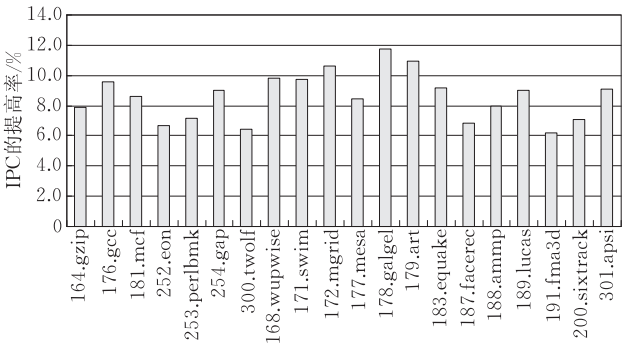

图 19 改进后 IPC 的提高

6 总结和未来工作

本文的工作立足于分析程序固有的访存特性,总结程序访存行为的规律性,并根据这些规律性给出高带宽访存流水线的低复杂度、低延迟、低功耗解决方案,降低了 load-to-use 延迟,提高了处理器的整体性能,从一定程度上缓解影响处理器性能的访存瓶颈. 本文的工作被用于指导 Godsonx 处理器的访存设计,在处理器整体面积增加 1.7%的情况下,将访存流水线的带宽提高了一倍,处理器的整体性能平均提高了 8.6%. 本文的后续工作将继续探索通用处理器的高带宽访存流水线的设计和优化空间.

参 考 文 献

[1] Saulsbury A, Pong F, Nowatzyk A. Missing the memory wall: The case for processor/memory integration//Proceedings of the 23rd Annual International Symposium on Computer Architecture(ISCA'96). Philadelphia, PA, USA, 1996: 90-101

[2] Wulf Wm A, McKee Sally A. Hitting the memory wall: Implications of the obvious. ACM SIGARCH Computer Architecture News, 1995, 23(1): 20-24

[3] Yeager Kenneth C. The MIPS R10000 superscalar microprocessor. IEEE Micro, 1996, 16(2): 28-41

[4] Compaq Corp. Alpha 21264 Microprocessor Hardware Reference Manual. 1999

[5] Hu Wei-Wu, Zhang Fu-Xin, Li Zu-Song. Microarchitecture of the Godson-2 processor. Journal of Computer Science and Technology, 2005, 20(2):243-249

[6] Doweck Jack. Intel smart memory access: Minimizing latency on Intel core microarchitecture. Intel Technology Magazine, 2006

[7] Agarwal Amit, Roy Kaushik, Vijaykumar T N. Exploring high bandwidth pipelined Cache architecture for scaled technology//Proceedings of the Design, Automation and Test in Europe Conference and Exhibition-2003 (DATE'03). Munich, Germany, 2003: 778-783

[8] Rivers Jude A, Tyson Gary S et al. On high-bandwidth data Cache design for multi-issue processors//Proceedings of the 30th International Symposium on Microarchitecture (MICRO-30). Research Triangle Park, North Carolina, USA, 1997: 46-56

[9] Kessler R E. The Alpha 21264 microprocessor. IEEE Micro, 1999, 19(2): 24-36

[10] Edmondson John H et al. Internal organization of the Alpha 21164: A 300-MHz&-l-bit quad-issue CMOS RISC microprocessor. Digital Technical Journal, Special 10th Anniversary Issue, 1995, 7(1): 119-135

[11] Keltcher Chetana N, McGrath Kevin J et al. The AMD opteron processor for multiprocessor servers. IEEE Micro, 2003, 23(2): 66-76

[12] Lee Chih-Chieh, Uhlig Richard A, Mudge Trevor N. A case study of a hardware-managed TLB in a multi-tasking environment. University of Michigan, Ann Arbor, MI, USA: Technical Report, CSE-TR-341-97 1997

[13] Chen J Bradley, Borg Anita, Jouppi Norman P. A simulation based study of TLB performance//Proceedings of the 19th International Symposium on Computer Architecture (ISCA-19). Gold Coast, Australia, 1992: 114-1

[14] Cekleov Michel, Dubois Michel. Virtual-address caches, Part I: Problems and solutions in uniprocessors. IEEE Micro, 1997, 17(5): 64-71

[15] Hennessy John L, Patterson David A. Computer Architecture: A Quantitative Approach. 3rd Edition. Amsterdam, Netherlands: Elsevier Science Pte Ltd, 2003

[16] Fan Dong-Rui, Tang Zhi-Min et al. An energy efficient TLB design methodology//Proceedings of the International Symposium on Low Power Electronics and Design-2005(ISLPED'05). San Diego, California, 2005: 351-356

[17] Min Jung-Hi, Lee Jung-Hoon et al. A Selectively Accessing TLB for High Performance and Lower Power Consumption//Proceedings of the APASIC'02. Taipei, China, 2002: 45-48

[18] Lee Jung-Hoon, Park Gi-Ho et al. A selective filter-bank TLB system//Proceedings of the ISLPED'03. Seoul, Korea, 2003: 312-317

[19] Heinrich Joe. MIPS R4000 Microprocessor User's Manual. Mountain View, CA ,USA: MIPS Technologies, Inc. , 1994

[20] Austin Todd M, Sohi Gurindar S. High-bandwidth address translation for multiple-issue processors//Proceedings of the 23rd Annual International Symposium on Computer Architecture (ISCA-23). Philadelphia, USA, 1996: 158-167

[21] Intel Corp. IA-32 Intel Architecture Software Developer's Manual, Vol. 1: Basic Architecture. Silicon Valley, USA: Intel Corp., 2003

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

[23] Wilson Kenneth M, Olukotun Kunle, Rosenblum Mendel. Increasing Cache port efficiency for dynamic superscalar microprocessors//Proceedings of the 23rd International Symposium on Computer Architecture (ISCA-23). Philadelphia, USA, 1996: 147-157

[24] Tendler J M, Dodson J S et al. POWER4 system microarchitecture. IBM Journal of Research and Development, 2002, 46(1): 5-25

ZHANG Hao, Ph. D., associate researcher. His research interests include design, analysis, implementation, and benchmarking of processor architectures; switching and routing of on chip networks; and high throughput memory system.

LIN Wei, born in 1979, Ph. D. candidate. His research focus on the architecture using massively parallel approach and the implementation of processor.

ZHOU Yong-Bin, born in 1981, Ph. D. candidate. His research interests include on-chip networks, on-chip memory system and parallel algorithm optimization on many-core architecture.

YE Xiao-Chun, born in 1981, Ph. D. candidate. His research interests include high performance computer architecture and algorithm parallelization & optimization.

FAN Dong-Rui, born in 1979, Ph. D., researcher. His research interests include high performance computing technique and advanced processor architectures.

Background

There is a near-exponential increase in processor speed and memory capacity. However, memory latencies have not improved as dramatically, and access times are increasingly limiting system performance. Low load-to-use latency is a key to approach high memory performance, and many researchers have devoted much time to increase cache hit ratio, but the pain is always much greater than the gain. Increasing the bandwidth of memory pipeline always works, but high bandwidth needs more ports for cache. Although a multi-ported cache design provides an excellent bandwidth at each cycle, its latency, area and power dissipation increase sharply as the number of ports increases. For memory access, there should be some tradeoff between performance and complexity/latency, which is more close to the performance side.

This work is under the support of the National Natural Science Foundation of China (grant No. 60736012), and the National Basic Research Program (973 Program) of China (grant No. 2005CB321600).