

一种基于 ESVW 技术的新型载入存储队列设计

马汝亮, 谢 憬, 毛志刚

(上海交通大学 微电子学院, 上海 200240)

摘 要: 在超标量处理器设计中, SVW 技术通过降低重新执行的载入指令数目提高系统性能. 本文提出一种基于 ESVW 技术的新型载入存储队列设计方案, 使得 SVW 结构中的 SSBF 不仅记录 SSN 值, 同时记录存储地址和数值, 重新执行的载入指令就可以从中获取数据而不必访问缓存, 进一步降低指令的重新执行率. 实验结果显示, 最优情况下, 超过 99% 的载入指令免于二次访问缓存, 系统性能提高约 2%.

关键词: 超标量处理器; LSQ; SVW; SSBF

中图分类号: TP302.2

文献标识码: A

文章编号: 1000-7180(2013)07-0020-04

A Novel Design of Load Store Queue with Enhanced Store Vulnerability Window

MA Ru-liang, XIE Jing, MAO Zhi-gang

(College of Microelectronics, Shanghai Jiaotong University, Shanghai 2000240, China)

Abstract: In superscalar process design, Store Vulnerability Window technique improves system performance by reduce load re-execute rate. In this paper, we present a novel design of load store queue with Enhanced SVW. We observe that if store sequence Bloom filter (SSBF) unit in SVW is modified to keep not only store sequence number (SSN) but also complete memory addresses and data values, trailing loads can get data from SSBF rather than cache, therefore data cache access rate will be further reduced. Experiments show that, in the best case, over 99% re-execute loads are kept from memory re-access, and system performance is improved by 2%.

Key words: superscalar processor; LSQ; SVW; SSBF

1 引言

载入存储队列 (Load Store Queue, LSQ) 是传统超标量处理器中的设计方案, 具有三重功能: ①避免指令相关性冲突; ②数据传递; ③精确中断. 为实现上述功能, 传统设计采取队列关联搜索的方式. 虽然有助于提高性能, 但弊端是关联搜索的不可扩展性. 如果队列深度增加, 地址匹配产生的延时将不可接受.

为解决这一难题, 不少设计方案被相继提出. G. Chrysos 和 J. Emer 提出存储指令集 (Store Set) 的概念^[1]. 载入和存储指令的相关性被检测出之后, 信息被记录在表格中. 指令再次执行时, 通过

访问表格提前确认相关性信息, 优化指令调度. Lei Jin 和 Sangyeun Cho 等人提出的宏数据载入 (Macro Data Load)^[2]可看作是预测技术的演变, 该技术通过增强载入数据重用达到减少缓存访问的设计效果. Amit Gandhi 等人提出可扩展的载入存储队列^[3], 分别给存储载入队列增加次级缓存. 载入队列的次级缓存是组相连缓存, 存储队列的次级缓存则是 FIFO 结构. Gharachorloo 等人以载入指令重新执行的方法代替关联搜索^[4], 而 Cain 和 Lipasti 等人对该技术进行优化^[5], 降低了被重新执行的指令数目. SVW 技术^[6]作为更有效的筛选技术, 使载入指令的重新执行率显著降低.

本文将介绍一种基于存储脆弱窗 (Store Vul-

收稿日期: 2012-10-23; 修回日期: 2012-11-14

基金项目: 国家“八六三”计划项目 (2009AA011705)

nerability Window, SVW) 的新型 LSQ 设计,称之为 ESVW(Enhanced SVW). ESVW 用载入指令重新执行的方法代替传统的地址匹配,通过改进 SVW 技术,进一步降低重新执行的载入指令数量.最后通过在 SimpleScalar 平台上对设计方案加载标准 Benchmark 测试程序,本文测试并分析了 ESVW 技术对免于二次访问缓存的指令比例和设计的优化效果.

2 增强型 SVW 设计方案

SVW 是在以载入指令重新执行为基础的优化技术.它通过给存储指令序列编号和 Bloom 筛选,使得约 85% 的载入指令免于重新执行^[6].

2.1 SVW 基本原理

在载入指令重新执行的设计方案中,如果载入指令所读的存储器地址在一定时间内没有被写入,该载入指令就没有必要重新执行. SVW 技术正是利用这一点,赋予每个顺序调度的存储指令一个递增的编号,称为存储序列编号(Store Sequence Number, SSN),利用 SSN 标记出可能跟载入指令有相关性的存储指令的范围.

载入指令发射时,最近提交的存储指令的 SSN 被记录在载入队列中,如果一个 SSN 值更大的存储指令先于该载入指令执行,那么指令的相关性就有被违反的可能.

SVW 技术使流水线级数增加一级,新流水级在提交级之前,称为 SVW. 存储指令在提交前,根据地址低位访问 SSBF 单元(Store Sequence Bloom Filter),记录下 SSN 值. 被标记的载入指令在提交前也要访问 SSBF,如果 SSBF 记录的 SSN 大于载

入指令的 SSN 值,则载入指令必须重新执行,消除指令相关性被违反的可能性.

2.2 SSBF 优化

经 SVW 筛选后仍有部分载入指令需要再次访问缓存.若出现重新执行的载入指令与其它指令竞争缓存的情况,系统性能会因此下降.优化 SSBF 可以缓解这一问题.

SSBF 以地址低位做索引,若经过 SSN 值比较,载入指令重新执行,这种情况下无法确认载入指令与 SSBF 记录的存储指令访问的是否为同一个存储地址.但是根据局部性原理,两者访问同一地址的可能性非常高.基于上述考虑,本论文提出 ESVW 技术,优化 SSBF 使其具有缓存的部分功能,不仅保存 SSN 值,同时保存存储地址和数据,那么载入指令在重新执行时,根据地址匹配的结果,可以从 SSBF 中而非缓存中获取数据,缓存访问率将进一步降低.

图 1(a)显示的是 SVW 实现基本结构图.存储队列包含地址域,数据域,SSN 域和地址匹配电路.存储指令在提交前根据地址低位访问 SSBF,SSN 值更新 SSBF 条目.由于 SSBF 访问和缓存写入是同时发生,因此 SSBF 更新不增加流水级.

载入队列包含地址域,数据域,SSN 域和重新执行位. SSN 域记录的是指令发射时最近提交的存储指令的 SSN. 如果载入指令发射时存在地址不明且指令顺序更靠前的存储指令,则重新执行位设为 1,否则为 0. 在提交时,根据重新执行位以及 SSN 比较结果,判断是否重新执行.

图 1(b)显示的是优化 SVW 之后的结构图. SSBF 中加入地址标签域和数据域,标签域辅助判断地址是否匹配,数据域在必要时为载入指令提供数据.

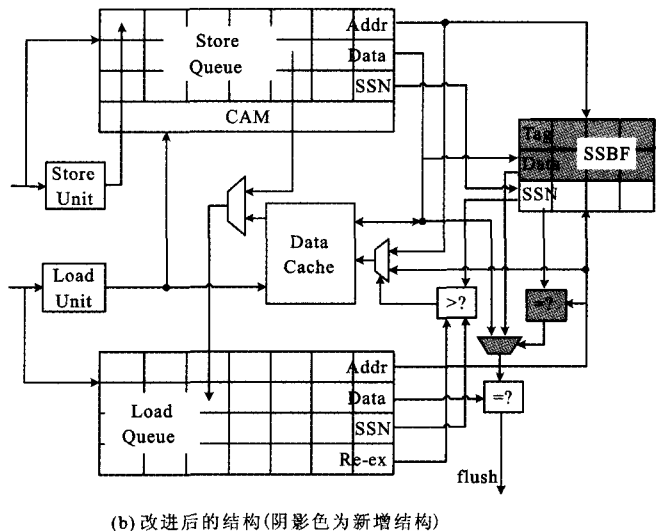
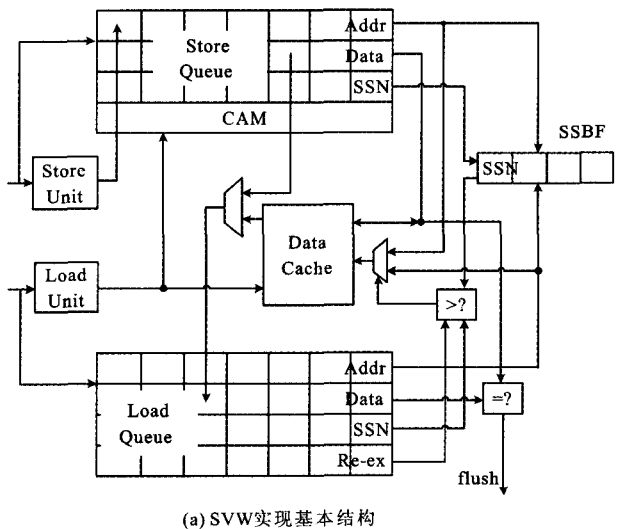


图 1 SVW 改进方案图

优化前后存储和载入指令的执行过程略有变化. 存储指令提交时, 需要将数据写入 SSBF 中, 这一过程与缓存写入同时发生, 因此不增加流水级.

载入指令在提交前, 如果重新执行位为 1, 那么要同时进行 SSN 值比较和地址匹配, 如果 SSN 比较结果显示需要重新执行, 而地址不匹配, 载入指令再次访问数据缓存, 系统性能未得到优化. 若此时地址也匹配, 载入指令可从 SSBF 中得到数据. 由于省去缓存的二次访问, 系统性能因此提高.

上述优化方案的设计代价是更多的硬件开销和更高的设计复杂度. SSBF 中增加地址标签域和数据域, 大小增加到原来的 3 到 4 倍, 判断是否二次访问缓存的电路规模也增大一倍左右.

3 实验结果和分析

为评估上文提出的新型载入存储队列的设计效果, 文章借助 SimpleScalar 仿真工具, 以 SPEC2000 整形测试包中的部分程序作为测试程序, 主要测量 SSBF 优化后, 免于二次访问缓存的载入指令百分比.

3.1 测试程序

SPEC2000 整形测试包中的一个子集作为实验的测试程序. 测试程序经 SimpleScalar 交叉编译工具编译, 无优化选项. 所有测试程序运行时以参考负载作为输入. Mcf 程序由于负载过大只采样了前

500M 条指令运行后的结果, 其余程序的仿真结果均是运行完毕后得出的.

3.2 主要结构设计定义

模拟出的超标量处理器宽度为 4, 顺序发射, 乱序执行, 主要设计细节如表 1 所示.

表 1 主要结构设计定义

模块	定义
SSBF	深度 128, SSN 值位宽 13, 地址标签位宽 25, 数据 32 位, 大小约 1.1KB
LSQ	深度 16
L1 数据缓存	16KB, 4 路组相连, 访问延时为 1 时钟周期
L1 指令缓存	16KB, 直接映射, 访问延时为 1 时钟周期
二级缓存	256KB, 4 路组相连, 访问延时为 6 时钟周期

3.3 结果和分析

图 2(a)显示的是 SSBF 的命中率. 从表中可以看出, 优化效果与程序类型和所加负载有关. 与一般程序相比, 数据密集型程序具有更显著的局部性特征, 因此像 gzip 和 parser 这样的数据密集型程序的优化效果更明显, 命中率接近 100%.

程序负载也会对实验结果产生影响. 负载是图像, 程序或源代码时, gzip 程序的命中率接近 100%, 一旦负载变为随机输入, 命中率急剧下降至不足 70%. 规则负载由于数据局部性更高, 因此优化效果更明显.

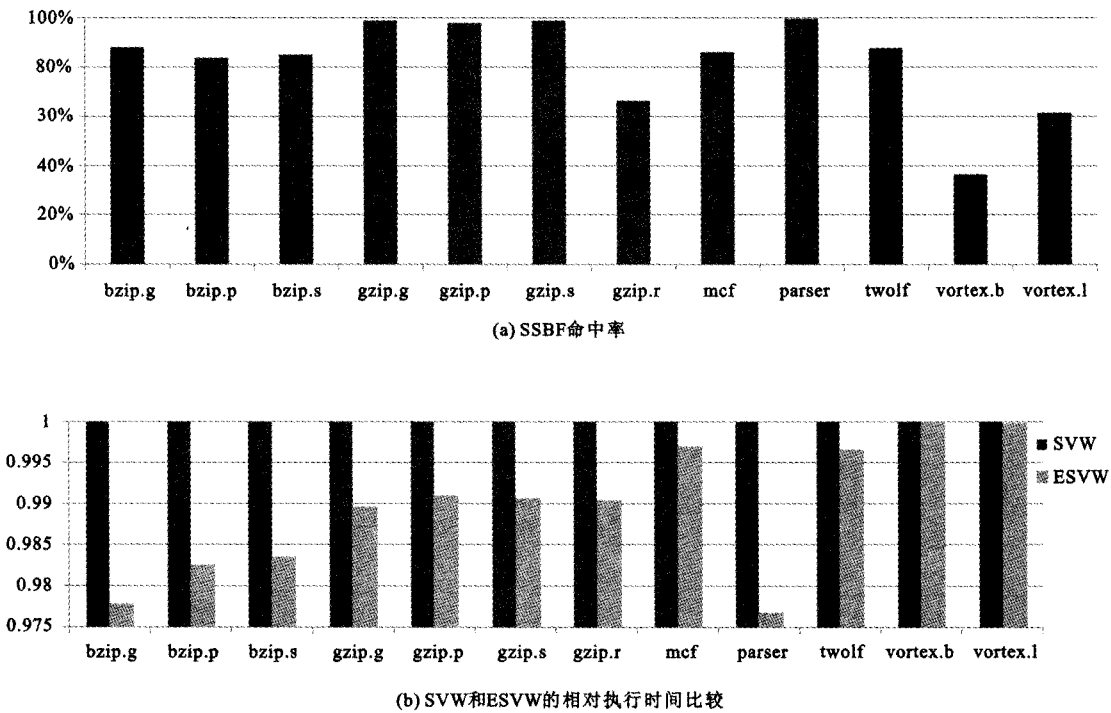


图 2 改进后的 ESWV 效果图

Vortex 的命中率明显低于其它测试程序,这与程序类型和负载属性均无关,而是重新执行的指令数目过少造成的.重新执行的载入指令只占总数的2%左右,数目以百计,与SSBF的深度相当,因此造成SSBF的强制性缺失.仿真结果显示,程序lendi-an(vortex.1)提交的指令数约是bendian(vortex.b)的2倍,前者的SSBF命中率同样约为后者的2倍.由此可以推断,随着vortex程序中重新执行的载入指令数目的增加,命中率也会随之上升.

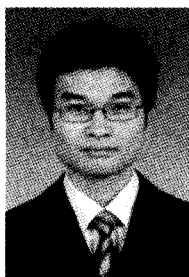
4 结束语

本文提出一种基于SVW技术的新型载入存储队列设计.本文提出改进原SVW技术中的SSBF单元使其不仅保存SSN值,而且保存数据值,在地址匹配的情况下代替缓存为载入指令提供数据.实验结果显示,具有规则输入的数据密集型程序性能优化最为显著,在最优情况下,超过99%的载入指令免于重新访问缓存,系统性能提高约2%.

参考文献:

- [1] Chrysos G, Emer J. Memory dependence prediction using store sets[C]// InISCA-25, Spain: Barcelona, 1998:142-153.
- [2] Lei Jin, Sangyeun Cho. Macro data load: an efficient mechanism for enhancing loaded data reuse[J]. Computers, IEEE Transactions on, 2011, 60(4): 526-537.
- [3] Gandhi A, Akkary H, Rajwar R, et al. Scalable load and store processing in latency tolerant processors[C]// Proc. 32nd International Symposium on Computer Architecture USA: Madison, Wisconsin, 2005: 446-457.
- [4] Gharachorloo K, Gupta A, Hennessy J. Two techniques to enhance the performance of memory consistency models[C]// Proc. of the International Conference on Parallel Processing. USA: Austin, 1991: 355-364.
- [5] Cain H, Lipasti M. Memory ordering: A value based definition[C]// Proc. 31st International Symposium on Computer Architecture, Germany: Munich, 2004: 90-101.
- [6] Roth A. Store vulnerability window (SVW): re-execution filtering for enhanced load optimization[C]// Proc. 32nd International Symposium on Computer Architecture. USA: Madison, Wisconsin, 2005: 458-468.

作者简介:



马汝亮 男,(1986—),硕士研究生.研究方向为大规模集成电路设计.

谢 憬 男,(1981—),助理研究员.研究方向为大规模集成电路设计.

毛志刚 男,(1962—),博士生导师,教授.研究方向为大规模集成电路设计.