

多线程非阻塞指令 Cache 设计

胡孔阳^{1,2}, 陈 鹏^{1,2}, 桑红石^{1,2}

(1 华中科技大学 图像识别与人工智能研究所, 湖北 武汉 430074;

2 华中科技大学 多谱信息处理技术国家级重点实验室, 湖北 武汉 430074)

摘 要: 非阻塞 Cache 是指 Cache 在等待预取数据返回时, 还能继续提供指令和数据. 首先分析了多线程非阻塞 Cache 的处理器需求, 然后提出其时序要求和一种实现方案. 利用 SystemVerilog 对该方案进行 RTL 级建模和性能评估. 仿真结果表明, 该方案可以很好地应用于多线程、乱序执行处理器的指令引擎设计之中.

关键词: 多线程; 非阻塞; Cache; SystemVerilog; 仿真模型

中图分类号: TP338.6

文献标识码: A

文章编号: 1000-7180(2012)05-0143-05

Design of a Multithreading Non-blocking Cache

HU Kong-yang^{1,2}, CHEN Peng^{1,2}, SANG Hong-shi^{1,2}

(1 Institute for Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, Wuhan 430074, China; 2 National Key Laboratory of Science and Technology on Multi-spectral Information Processing, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: Non-blocking instruction Cache is one Cache that can continue to provide instruction and data, when waiting for the prefetch data. In this paper, first analyze the processors' demand for multithreading non-blocking cache, then put forward the timing request and the functional structure. SystemVerilog is employed to build up the simulation model of the proposed architecture and the performance evaluation. Evaluation results show that the architecture can be applied to the design of fetch engine in multithreading or out of order execution processors.

Key words: multithreading; non-blocking; cache; SystemVerilog; simulation mode

1 引言

Cache 在任何一种处理器系统中都起了相当重要的作用. Cache 中的 SRAM 读取速度一般要比主存储器(DRAM)快 3~4 倍. 根据时间局部性和空间局部性原理, Cache 可以极大地提高处理器的效率^[1-8].

为了提高处理器的性能, 很多 CPU 和 GPU 均支持多线程和超前执行机制, 在同一时间内尽可能处理多个线程的数据以及进行数据和指令的预取. 例如 SUN 公司 2006 年研发的多核、多线程的处理器 UltraSPARC T1. 该处理器可以有最多 8 个 CPU 核, 每个核最多可同时处理 4 个线程, 在最理

想的状态下可以同时执行 32 个线程. 对于传统的阻塞 Cache 而言, 当某个请求发生失效时, Cache 会因为等待外部总线的数据请求而停止接收处理器的后续请求, 导致处理器被迫暂停后续的访问存储操作, 这样会占用很多时钟周期, 导致性能严重下降. 而在非阻塞的 Cache 设计中, Cache 并不会因为某个失效而暂停, 这对于多核、乱序执行机制的处理器而言是十分必要的.

2 Cache 设计与优化策略综述

文献[9]中对非阻塞 Cache 提出了一种关键字优先的策略. 其主要思想为: 当出现一次 Cache 请求缺失时, 并不影响后续指令继续访问, 但是如果连续

收稿日期: 2011-09-02; 修回日期: 2011-10-28

基金项目: 国家自然科学基金(60736010)

两次均发生访问失效,流水线则会被阻塞.根据程序局部性原理,连续发生两次访问失效的概率很大,所以该方案对于非阻塞的要求解决得并不好.

文献[4]中提出了一种访问预取策略.增加一种分析访问失效的状态,并在失效时向主存发出预取请求,这样可以更好地利用程序的局部性原理,提高 Cache 命中概率.但是分析文章可以发现,该设计的预取请求和失效的总线数据请求相冲突,经常会导致失效的请求由于预取请求占用数据总线而推迟发送,在一定程度上阻塞了 Cache,降低了 Cache 的性能.

本文提出的非阻塞 Cache 结构对最多连续 8 次请求缺失情况不会暂停流水线,针对具体处理器要求还可配置对应参数,将最大缺失数进一步提高.同时相较于文献[4],提出一种优化的预取策略,对每次缺失引起的外部总线请求,均预返回相邻地址数据(见设计中的 Cache Line 作用),预取地址直接由缺失地址高位生成,因而不会造成总线请求数据的冲突.

3 多线程非阻塞指令 Cache 的结构设计

本文设计的多线程非阻塞指令 Cache 旨在为多线程、乱序执行的并行处理器提供一种取指引擎.设计方案具有可配置,高性能,适合 ASIC 工艺的可综合性设计等特点,并且可以支持线程撤销请求,即当撤销请求 cancel 有效时,停止发送 cancel 信号有效前对应线程的数据.

3.1 设计指令 Cache 的基本工作原理

设计的指令 Cache 存储的是物理地址,需要将 CPU 输出请求的虚拟地址先由 TLB(Translation Lookaside Buffer)转化成物理地址.

直接映像高速缓存结构:每个物理地址(主存地址)对应于高速缓存中一个确定的地址.对应关系为:取物理地址中的低位作为 Cache 中的地址 index.由于每个 Cache 中的地址可能对应主存中的多个不同的地址,为了确定所请求的字是否在 Cache 中,将物理地址中的高位作为标记信息 TAG,这样一个物理地址中的 index 就对应一个标记 TAG 和一个数据块 data.

Cache 访问过程为:若一个物理地址中的 TAG 与经其 index 寻址 Cache 得到的 TAG 相同时,则命中,同时输出经该 index 寻址得到的 data.为了减少 Cache 中块的数量(index 的宽度),同时根据程序局部性原理进行数据预取,将原 index 中的低若干位

划分为 CL(Cache Line)位,此时一个物理地址中的 index 仍对应一个 TAG,但是却对应 CL 个数据,物理地址对应 data 的寻址变为 $\text{index} + \text{CL}$ (原 index).同时,由于一个物理地址对应的数据块并不止一个字节,所以将物理地址中的低位设为字节偏移 offset.

转换后的物理地址划分为:标记值 TAG,下标 index,Cache Line,和偏移 offset 四个部分.见图 1.

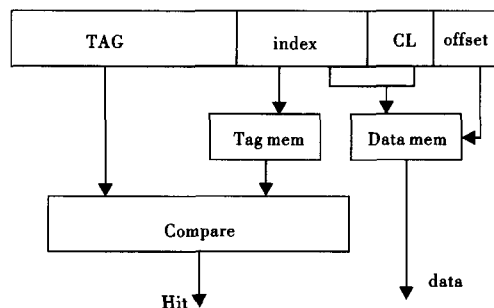


图 1 物理地址的划分

组相联:在一个组相联高速缓存中,每个块在替换时可以被放在固定数量(至少 2 个)的位置上;每个块有 N 个位置可放的组相联高速缓存称为 N 路组相联高速缓存.一个 N 路组相联高速缓存由许多组构成,每个组中有 N 个块.每个物理地址由下标 index 对应到高速缓存中的唯一的组,替换时可以放在此组中任何一块的位置上.检测物理地址是否命中时,将物理地址由 index 映像到对应组,然后检索该组中所有块看 TAG 是否匹配.

例如:采用四路组相联映像方式时,将 Cache 分成 u 组,每组中有 4 个块,Cache 总块数为 $4u$.通过物理地址访问 Cache 时,index 选择的一组的全部标记(4 个)同时被送往比较器与主存地址给出的标记进行比较.如果命中,则按块内地址访问组内命中块,否则 Cache 命中失败,将物理地址发送到总线,访问主存.如图 2 所示.

Cache 组相联的替换策略:采用 LRU(Least Recently Used),最近最少使用算法.

最近最少使用算法:被装入或被替换的块,其对应的计数器清为“0”,同组中其他所有块所属的计数器都加“1”.命中的块,其对应的计数器清为“0”.同组中其它所有计数器中,凡是计数器的值小于命中块所属计数器原来值的,都加“1”,其他计数器不变,设计中要增加判断逻辑防止计数器溢出.需要替换时,在同组的所有计数器中选择计数值最大(一般为全 1)的计数器,它所对应的块就是要被替换的块.

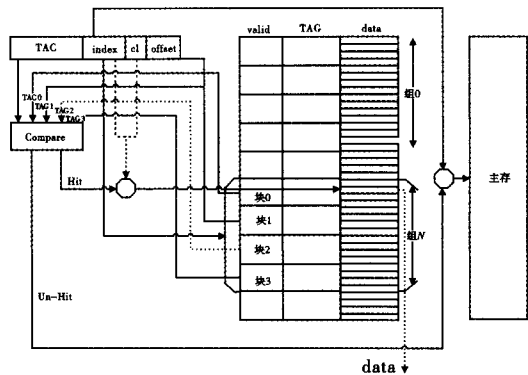


图 2 四路组相联示意图

3.2 设计指令 Cache 的时序要求

图 3 中 rdy(ready)为 Cache 的就绪信号,如果 rdy 无效则停止接收 CPU 的请求;req(request)为 CPU 的请求使能信号;PC 和 tid(thread identity)为 CPU 请求的虚拟地址和线程号;en(enable)为 Cache 返回数据的使能信号;data 和 tid 为 Cache 返回的数据和对应 tid 号。

对数据返回顺序的要求:同一线程间返回的数据需要和输入请求顺序相一致,不同线程间可以不按输入顺序返回。

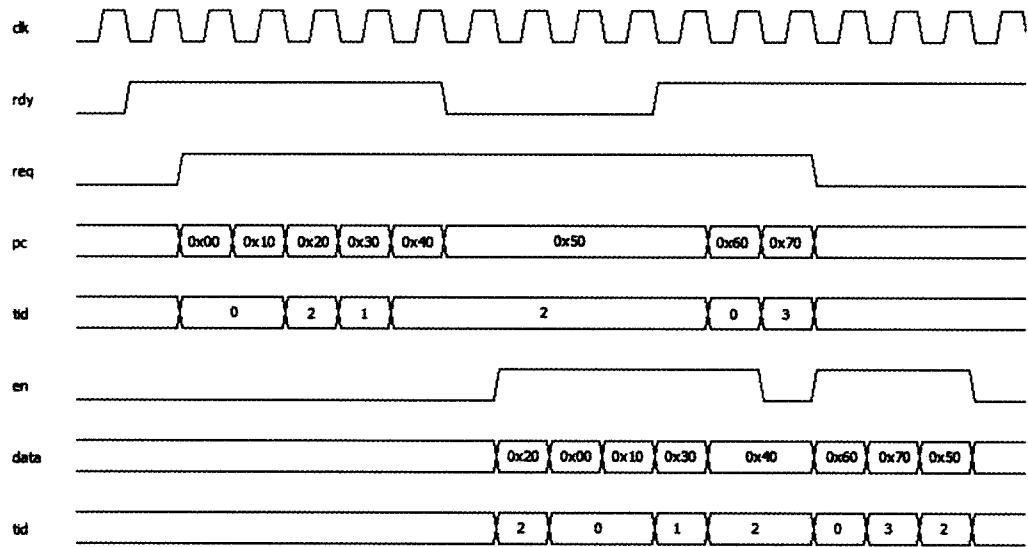


图 3 指令 Chche 与 CPU 接口时序

3.3 多线程、非阻塞的结构设计

为了实现非阻塞和多线程返回顺序的要求,在设计的指令 Cache 中使用 Cache 缓冲表和线程请求顺序表两种硬件结构来实现。

Cache 缓冲表中记录了 CPU 请求的虚拟地址、线程号及经 TLB 转换后的物理地址和返回数据,同时保存每个请求的状态信息和 cancel 信号有效时的保护位。线程请求顺序表中记录每个线程输入请求在 Cache 缓冲表中的位置,返回数据时查询该表,先请求先返回。设计中的大量组合逻辑围绕上述两种表结构进行,对请求地址的转换、标记命中的检测、外部总线的数据请求及数据的输出均有对应的电路模块独立工作,搜寻缓冲表中对应状态的表项并行处理,以达到非阻塞的设计目的。

同时,当虚拟地址向物理地址转换时,为了避免

多次请求 TLB,在 Cache 中生成一个简单的 TLB 缓冲结构,转换优先访问 TLB 缓冲表,未命中则发起 TLB 请求。设计的指令 Cache 结构见图 4。

3.4 Cache 状态机设计

Cache 状态机如图 5 所示。

st_idle:空闲态。Cache Table 缓冲表对应项为空,可以接收新的请求,记录新的请求后转到 st_wtlb 态。

st_wtlb:等待查询 tlb 缓冲表状态。如果 tlb 缓冲表命中,则将其中的物理地址写回至 Cache 缓冲表中对应的表项,并将状态转至 st_trdy;未命中,则状态转至 st_tmiss。

st_tmiss:tlb 缓冲表 miss。发起 tlb 请求并等待 tlb 返回物理地址,地址转换完成后,转到 st_trdy。

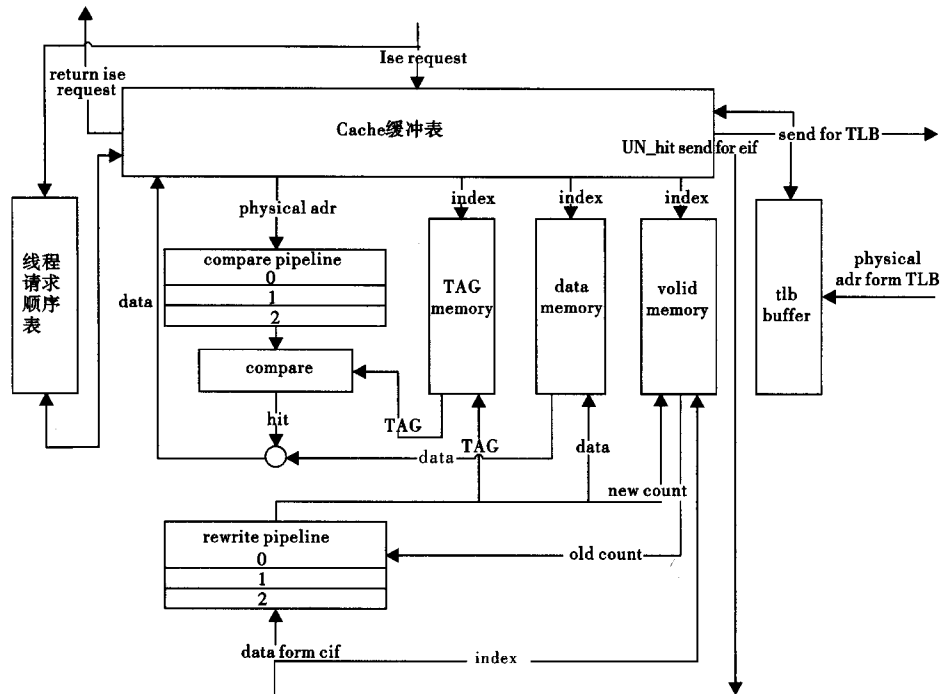


图4 指令Cache模块结构图

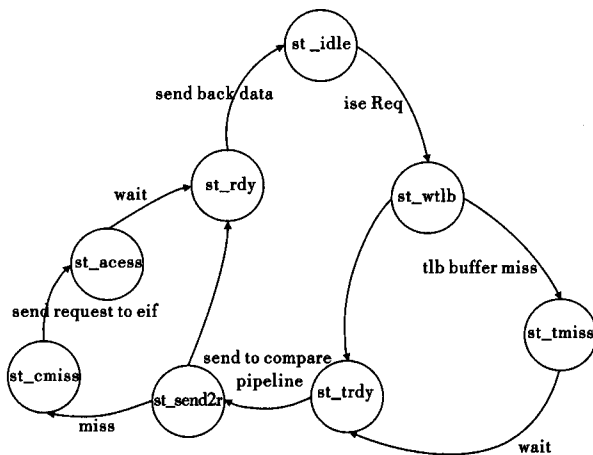


图5 指令Cache状态机

st_trdy: 物理地址转换完成, 等待检测 TAG 是否命中. 遍历 Cache 缓冲表选择处于 st_trdy 状态的表项, 送出其物理地址中的 index, 并将 TAG 送入延迟线, 等待与 TAG memory 中的对应项进行比较.

st_send2r: 送入比较延迟线进行命中检测. 如果 TAG 相同, 则命中, 将读出的 data 写回缓冲表, 将状态变为 st_rdy; 如果没有命中, 则将状态改为 st_cmiss.

st_cmiss: TAG 未命中. 遍历缓冲表选择处于 st_cmiss 的状态, 向 eif (external interface) 发送外部数据请求, 并将状态置为 st_access.

st_access: 已发送外部数据请求, 等待数据返回. eif 返回数据后, 状态转至 st_rdy.

st_rdy: 数据准备就绪, 查询线程请求顺序表, 相同 tid 按输入顺序返回, 数据返回至 ise (instruction stream engine) 后, 重新转到 st_idle, 准备接收新的请求.

4 指令Cache的仿真实验、综合

基于 SystemVerilog 业界制定了一些描述模块层次结构及模块间数据交互的模式, 其中应用比较广的是 Mentor Graphic 和 Cadence 公司推出的 OVM 库, 利用这套库可以很方便的建立模块并完成模块间的数据交互. 我们在 QuestaSim 下利用 OVM 库建立相关模块并完成模块间数据交互, 随机生成仿真数据和 cancel 信号. 为了避免人工校对数据繁重的工作量和容易失误的情况, 在 test bench 中增加对返回数据的自动校对匹配. 如图 6 所示.

设计的指令 Cache 其组相联数, 标记值、下标值、Cache Line 长度、数据宽度、片上存储器相关参数均可配置. 在验证环境下, 通过修改这些参数值, 可以很方便的发现参数对性能的影响.

通过仿真结果看出, 本文设计的多线程非阻塞指令 Cache, 在请求最短返回周期和数据预取机制上具有优势, 对于指令最大阻塞长度, 可以通过增加

Cache 缓冲表的深度进一步增加,但是由于 Cache 缓冲表占用了较多的寄存器资源,需要在面积和功耗上做出取舍。

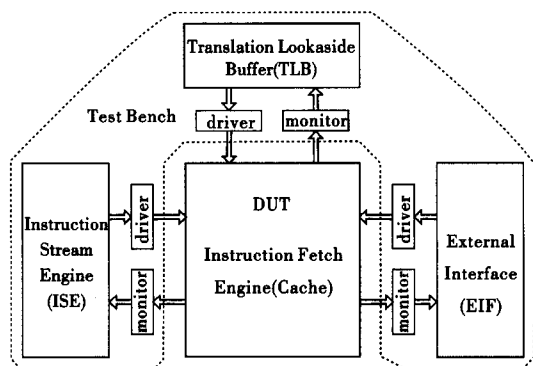


图 6 测试模块互联

5 结束语

通过验证环境的测试及功能覆盖率检测,保证了设计方案的功能正确性,但是并不能很好的反映出程序中指令和数据相关性的特点。进一步的工作除完成数字后端,进一步优化降低设计功耗外,需要将 Cache 应用于本实验室开发的 EPIP 处理器之中,利用处理器程序进行仿真,比较加入 Cache 前后处理性能的提升。

参考文献:

- [1] Huang Yongqin, Yuan Aidong, Li Jun, et al. A Novel Directory-Based Non-busy, Non-blocking Cache Coherence//IEEE Computer Science Technology and Applications. Wuxi: IEEE, 2010: 21-24.
- [2] Brown J A, Porter L, Tullsen D M. Fast thread mi-

gration via cache working set prediction[J]. IEEE High Performance Computer Architecture, 2011, 31(2): 31-34.

- [3] Teng-Feng Yang, Chung-Hsiang Lin, Chia-Lin Yang. Cache-aware task scheduling on multi-core architecture [J]. VLSI Design Automation and Test, 2010.
- [4] Dhakad P, Katariya A, Arya A, Performance Verification for Cache Memory of Multicore Processor[J] IEEE Computational Intelligence and Communication Networks, February 2011,12(5): 43-45.
- [5] 何荣森,何希顺,张跃. 从 ARM 体系看嵌入式处理器的发展 [J]. 微电子学与计算机, 2002,19(5): 3-4.
- [6] 马卓,刘宗林,吴虎成. X86 结构中微程序 ROM 的实现技术浅析 [J]. 微电子学与计算机, 2007, 24(4): 5-7.
- [7] 陈鹏,袁雅婧,桑红石,等. 一种可扩展的并行处理器模型设计及性能评估 [J]. 航空兵器, 2010: 5-6.
- [8] 胡伟武,张福新,李祖松. 龙芯 2 号处理器设计和性能分析 [J]. 计算机研究与发展, 2006,43(6): 8-9.
- [9] Pease R L, Sternberg A L, Boulghassoul Y, et al. Comparison of SETs in bipolar linear circuits generated with an ion microbeam, laser light, and circuit simulation [J]IEEE Trans, 2003.

作者简介:



胡孔阳 男,(1986—),硕士.研究方向为数字 IC 设计。