

应用预取策略的行缓冲指令 Cache 设计

李伟立, 于立新

(北京微电子技术研究所, 北京 100076)

摘 要: 行缓冲是一种有效的低功耗方案, 但其极大地降低了处理器的运算性能. 设计并实现了使用预取策略的行缓冲 Cache, 使用一个缓冲行来预取存储在 L1 Cache 中的指令, 从而降低了行缓冲结构中由于容量缺失而造成的流水线停顿, 提升了处理器的运算性能. 以 Leon2 的 VHDL 模型为试验环境进行了验证, 带有预取策略的行缓冲结构较原来的结构平均提升了 12.4%.

关键词: 行缓冲; 预取; 性能

中图分类号: TP302.8

文献标识码: A

文章编号: 1000-7180(2011)01-0038-04

Line Buffer Instruction Cache with Prefetch Scheme

LI Wei-li, YU Li-xin

(Beijing Microelectronics Technology Institute, Beijing 100076, China)

Abstract: Line Buffer Cache architecture, which adds a line buffer between Level 1 cache and IU, has the advantage of low energy consumption. But the scheme decreases the performance dramatically. The paper introduces the prefetch scheme to the Line Buffer Cache. A buffer is used to prefetch the instructions that reside in the L1 Cache. The method reduces the miss rate and the stalls of pipeline due to the low capacity, hence improves the performance. The Leon2 VHDL model is used as the environment to conduct the experiments, and the proposed architecture improves the performance by 12.4% on average.

Key words: Line Buffer; prefetch; performance

1 引言

现代的处理器的, 无论是服务器、桌面应用或是嵌入式处理器中, Cache 都是一个不可或缺的组成部分. 尽管 Cache 的容量不大, 但是它却能极大地降低存储器与处理器之间的速度差异的影响, 提升处理器的性能. 但同时 Cache 消耗了大部分的处理器功耗, 由此带来了散热以及封装等问题. 在 Cache 的设计上, 要兼顾性能与功耗两个方面.

组相联结构的 Cache 中, 同组内的标志与数据都要被读出来进行比较, 但即使命中, 也仅有一组是有效的, 访问其他的组则会浪费能量^[1]. 有鉴于此, 一些低功耗的方案被提出来. 路预测组相联^[2]结构中, 使用标志位标记最近访问的行, 并在接下来的访问中只访问标记的行, 节省了能量. 但是当发生缺失

时, 则会浪费一个时钟周期. 两相 Cache 结构^[3]将标志的读取、比较与数据的读取分离开来, 先读标志并比较, 如果命中再读命中行的数据, 不命中则没有数据访问. 此时能节省功耗, 但是却极大地增加了单次访问 Cache 的延时, 影响处理器的时钟频率的提升. 行缓冲结构中^[4], 每次对 L1 Cache 的访问都读取整个的 Cache 行, 并对该行进行缓存. 取指时先访问缓存的行, 如果不命中才在接下来的时钟周期访问 L1 Cache. 如果接下来的访问是对同一个行的, 则能减少对 L1 Cache 的访问, 但缺失的时候会极大地降低性能. Filter Cache^[5]与行缓冲结构类似, 它使用了更多的行来缓存 L1 Cache 中的数据, 但同样对性能有较大的影响. 带有预测的行缓冲^[6]是一个高性能的方案. 此结构中使用预测器来预测下次的访问的目标是行缓冲, 还是 L1 Cache. 根据预测的结果, 选

择访问这两者之一. 如果预测的正确率较高, 此时即降低了功耗, 又不损失太多的性能. Hot-Spot 结构^[7]使用 L0 Cache 来存储访问频率很高的指令段, 而其他的指令则存储在 L1 Cache 中. 结构中实现了一个仲裁逻辑, 用来决定指令段的存储位置, 同时用预测逻辑来预测下次访问的位置. 此结构基本不降低处理器的性能, 同时降低了指令 Cache 的功耗.

在文献[6-7]中,都使用了预测的策略,用来降低功耗,提升性能。预取也是一种有效的提高性能的策略。本文将预取策略应用于行缓冲结构中,设计并实现了应用预取策略的行缓冲结构。

2 预取的行缓冲结构设计

2.1 行为级设计

行缓冲结构是一种有效的低功耗方案,然而其却极大地降低处理器的性能,而预取是一种提高存储器性能的有效方法.综合这两个技术,便形成了应用了预取策略的行缓冲指令 Cache 方案.预取结构是在行缓冲结构的基础上,预取当前缓冲行的下一行,亦即,在 L0 级应用预取的策略.在行缓冲结构中,有一个与 L1 Cache 的 1 行大小相同的存储器来缓存当行访问的行.而应用了预取策略后,为了防止预取的行替换掉当前的访问行,需要增加一个同样容量的存储器来存储预取的 Cache 行.因而在预取结构中,要实现两个缓冲行.同时还要实现一个执行历史表,用来记录指令的执行过程,并产生预取的地址.运行过程中,同时访问缓冲行与预取来的行,一旦取指请求命中于预取行中时,将预取的行输入到缓冲行中,同时预取新的指令.

2.2 结构级设计

应用了预取策略的处理器结构如图 1 所示。在 IU(Integer Unit)与 L1 Cache 之间,使用了两个行缓冲结构,分别为 PLB(Prefetch Line Buffer)及 NLB(Normal Line Buffer)。它们都是用来缓存由 L1 Cache 取来的一个 Cache 行,其中 NLB 主要用的正常的访问,而 PLB 则主要用于缓存预取的行。这两个行缓冲结构都包含有标志、有效和数据三个字段,另外 PLB 还包含一个预取地址字段。其中,有效字段与数据字段的位宽与 L1 Cache 中的一致,而标志字段的倍数要比 L1 Cache 多出几位(与 Cache 容量,相联度以及行的大小相关)。预取地址字段则是用来缓存由预取地址生成逻辑产生的预取地址。

由于程序运行是表现出时间及空间的局域性,使用执行历史表来记录程序执行的信息,并以此产

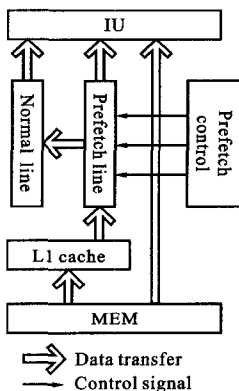


图 1 预取结构框图

生预取地址. 执行历史表结构如图 2 所示(两路组相联结构). 执行历史表的结构与 L1 Cache 相同, 每个 Cache 行对应一个执行历史表项, 并使用相同的地址, 同时被访问. 表项中, 记录本表项所对应的 Cache 行的下一个执行的首地址, 也就是本 Cache 行的预取地址. 预取地址在当前行被存储到 L1 Cache 时产生, 在替换出 Cache 后也同样被替换. 这一地址或者是当前行的下一行的地址, 或者是当前行内分支指令的目标地址, 如图 3 所示.

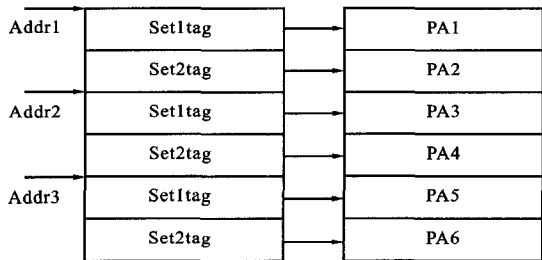


图 2 执行历史表框图

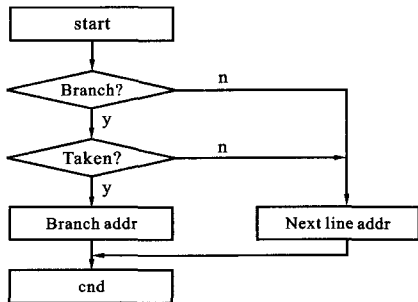


图 3 预取地址生成方法

执行历史表与 Cache 一同被访问,每次预取启动时,执行历史表中的地址都被输入到 PLB 的预取地址字段中,在预取行被访问时,以此地址作为指令的预取地址,进行预取。

2.3 状态机设计

状态机使用了 4 个状态来实现, 分别为命中态、预取态、过渡态以及访存态, 功能如表 1 所示. 状态

转换图如图 4 所示。

表 1 状态功能描述

状态	功能描述
命中态	访问 NLB 及 PLB
预取态	预取 L1 Cache 中指令
过渡态	设置信号, 以返回到命中态
访存态	访问主存储器

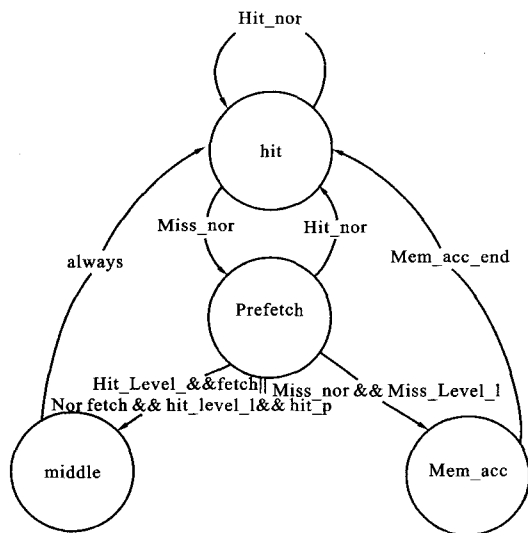


图 4 预取结构状态转换图

命中态是默认的状态, 在命中态中, NLB 与 PLB 同时被访问, 并且 IU 的取指请求命中于 NLB 或 PLB 中. 如果请求命中于 NLB, 则直接返回请求的指令, 不改变状态; 如果请求命中于 PLB 中, 则由 PLB 返回请求的数据, 同时, 将 PLB 的内容传输至 NLB, 并利用 PLB 的预取地址启动预取, 进入预取态; 如果 PLB 和 NLB 均缺失, 以请求指令的地址访问 L1 Cache, 进入预取态进行处理。

预取态中, IU 请求的指令由 NLB 响应, 同时访问 L1 Cache. 如果命中于 NLB 中, 则不论预取命中与否, 都将返回到命中状态; 如果 NLB 缺失, 那么根据进入预取态时的信

号(表明当前访问 L1 Cache 是取操作还是预取操作)以及当前预取的结果, 进入过渡态或者是访存态. 过渡态内的逻辑转换如图 5 所示。

首先判断此状态中进行的是预取, 还是真正的取指. 如果是取指, 接下来判断取指请求是否命中于 L1 Cache 中, 根据判断的结果, 进入过渡态或者是访存态. 如果是预取, 依据 NLB 与 PLB 的命中信息, 返回命中态, 或者进入过渡态及访存态。

过渡态是预取态与命中态之间的过渡性状态,

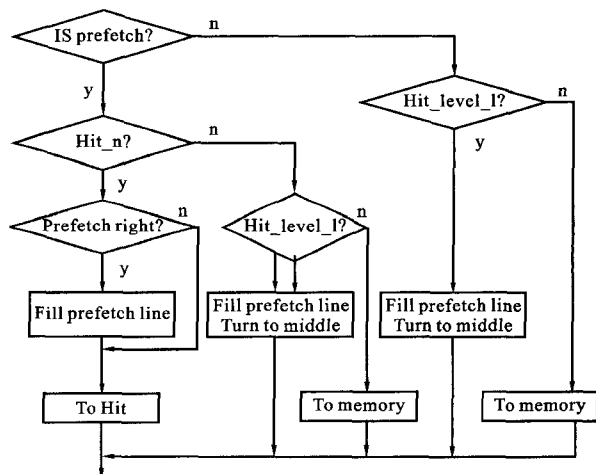


图 5 预取状态中流程

在此状态中清除由命中态至预取态的控制信号. 进入此状态前, 一定发生了由于指令缺失而造成的流水线停顿, 所以在此状态中, 用正确的指令填充流水线译码级, 并重启流水线。

访存态是在 NLB、PLB 缺失, 并且 L1 Cache 也缺失的时候, 访问主存储器, 以获得所需的指令。

3 测试环境与结果

本文中, 使用开源的 Leon2 VHDL 模型作为基本的仿真环境. Leon 2 VHDL 模型基于 GPL 许可协议, 实现了符合 SPARC V8 标准的 32 位的处理器. Leon2 模型是高度可配置的, 使用 5 级流水线, 并在片上集成了单独的指令和数据 Cache. 实验过程中, 指令 Cache 的容量为 32 kB, 4 路组相联结构, 行大小为 32 字节; 数据 Cache 的容量为 16 kB, 2 路组相联结构, 行大小为 16 字节. 指令和数据 Cache 都采用 LRU 替换算法. L1 Cache 每个时钟周期完成一次取指请求, 而主存储器则每两个周期完成一次存储器访问。

在以上的配置下, 修改指令 Cache 的文件, 实行缓冲结构, Filter 结构, 以及应用了预取策略的结构, 并对这几个结构进行性能比较. 行缓冲结构中, 使用寄存器实现了容量为 1 个 L1 Cache 行大小的缓冲器, 其每个时钟周期完成 1 次访问请求; Filter 结构中, 使用寄存器实现了容量为 4 个 Cache 行的 L0 级 Cache; 预取结构中, 用寄存器实现了 NLB 以及 PLB, 且都可可在一个时钟周期内被访问。

这几种结构都是可综合的, 并在 SMIC 0.18 μm 工艺下进行了综合. 测试程序使用的是 Leon2 模型中自带的测试程序, 以及 Dhrystone 2.1 版本的程序. 本文比较了各结构的 CPI (Circles Per In-

struction)、功耗和芯片使用的面积. 为了提高可读性,所有的 CPI 都对 Leon2 的 CPI 进行了归一化.

图 6 中显示,应用预取策略的结构在大部分测试程序中运行的要快于其他的结构,表现为较小的 CPI. 尤其在 Dhrystone 中,预取结构相对于行缓冲结构要快 15.2%,相对 Filter 结构要快 7.1%. 对平均性能来讲,预取结构相对行缓冲结构,性能提升 12.4%,而对 Filter 结构,性能提升 7.3%. 但对 Cache 与 Watchpoint 等程序,性能却不升反降,这是由于在这两个测试程序中,指令 Cache 被禁止,此时预取策略不起作用,因而影响了性能.

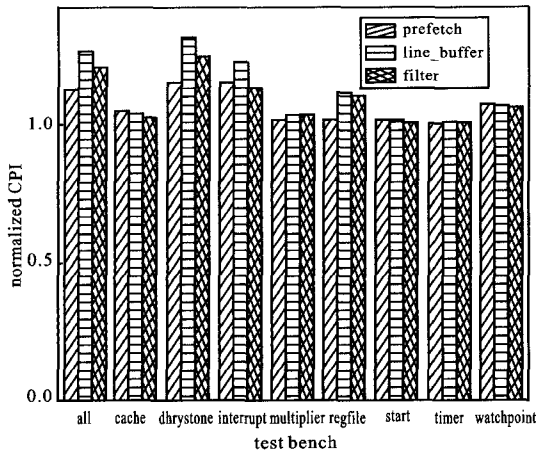


图 6 归一化的 CPI 比较图

图 7 所示为功耗与芯片面积的比较. 面积是综合后得到,功耗是由综合后的网表仿真得出. 所提结构比行缓冲结构多消耗了 3.0%的面积与 4.9%的功耗;相对于 Filter 结构,多使用了 4.6%的面积,但是节省了 3%的功耗.

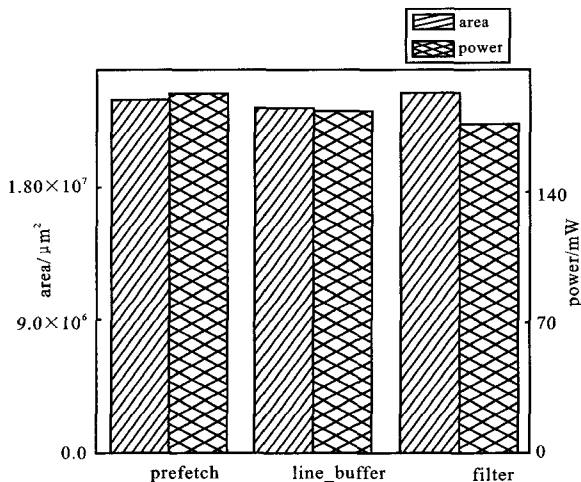


图 7 面积与功耗比较图

4 结束语

文中将预取策略应用于行缓冲结构中,以较小的功耗与面积代价,取得了较大的性能提升. 接下来的工作中,可进一步优化预取地址的生成方法,提高预取的准备率,进一步提升系统的性能.

参考文献:

- [1] Albonesi. Selective cache ways: on-demand cache resource allocation[C]// 32nd ACM/IEEE Symposium on Microarchitecture. Flaifa: ACM/IEEE, 1999: 248-259.
- [2] Inoue K, Ishihara T, Murakami K. Way-predicting set-associative cache for high performance and low energy consumption[C]// ACM/IEEE international Symposium on Low Power Electronics and Design. New Orleans: ACM/IEEE, 1999:273-275.
- [3] Hasegawa A, Kawasaki I, Yamada K, et al. High Code Density, Low Power[J]. IEEE Micro, 1995,15(6):11-19.
- [4] Ghose K, Kamble M. Reducing Power in Superscalar Processor Caches using Subbanking Multiple Line Buffers and Bit-Line Segmentation[C]// Symposium on Low Power Electronics and Design. San Diego: ACM, 1999:70-75.
- [5] Vivekanan K, Srikanthan T. Custom instruction filter cache synthesis for low-power embedded systems [C]// 16th IEEE Workshop on Rapid System Prototyping. Chania: IEEE, 2005:151-157.
- [6] Ali K, Aloelaze M, Datta S. Energy Efficient I-Cache using Multiple Line Buffers with Prediction[J]. IET Comput Digit Tech, 2008(2):355-362.
- [7] Yang Chia Lin, Lee Chien Hao. Hotspot Cache: Joint Temporal and Spatial Locality Exploitation for I-Cache energy Reduction[C]// Symposias on Low Power Electronics and Design ISPLD' 04. Newport Beach: ACM, 2004:114-119.

作者简介:

李伟立 男,(1983-),硕士研究生. 研究方向为微处理器设计与 SoC 系统集成.

于立新 男,(1966-),研究员,硕士研究生导师. 研究方向为微处理器设计与 SoC 系统集成.