# 标志预访问和组选择历史相结合的低功耗指令 cache

张宇弘[1],王界兵[2],严晓浪[3],汪乐宇[1]
(1. 浙江大学数字技术及仪器研究所,浙江杭州 310027;2. 中天微系统有限公司,浙江杭州 310027;
3. 浙江大学 VLSI 研究所,浙江杭州 310027)

**摘 要**: 指令 cache 是处理器的主要耗能部件之一. 研究发现,在指令顺序执行的情况下,访问同一 cache 行只需要访问一次标志存储器,因此标志存储器存在大量空闲周期. 本方法利用标志存储器的空闲周期来预先访问地址连续的下一个 cache 行的标志,从而预先获得 cache 行命中和组选择信息,这样当真正取下一行的指令时,根据获得的该 cache 行的标志信息就无需访问没有被选中的数据存储器. 预先访问标志存储器的另一个优点是可以加入组预测算法来减少对标志存储器的访问. 为了减少短距离跳转时对 cache 的访问,环形历史缓冲区(CHB)保存了部分组选择结果来获得跳转目标地址的 cache 行信息. 该方法没有性能损失,而且具有硬件实现简单,硬件代价小等优点. 该方法已被应用于 250MHz 的 RISC 处理器中.

**关键词**: cache; 低功耗; CPU; 微体系结构
**中图分类号**: TN47 **文献标识码**: A **文章编号**: 0372-2112 (2004) 08-1286-04

# Pre-Visiting Tag and Keeping Way History to Reduce Power in Instruction Cache

ZHANG Yu-hong[1], WANG Jie-bing[2], YAN Xiao-lang[3], WANG Le-yu[1]
(1. *Digital Technology and Instrument Institute of Zhejiang University, Hangzhou, Zhejiang* 310027, *China*;
2. *C-sky Microsystem Company, Hangzhou, Zhejiang* 310027; 3. *VLSI Institute of Zhejiang University, Hangzhou, Zhejiang* 310027, *China*)

Abstract: Instruction cache consumes a large portion of power in processor. By using this method the number of accesses to both tag and data array of set-associative instruction cache can be significantly reduced. The method takes advantage of the fact that tag can be accessed once per line during sequential execution. And in the idle cycles of the tag memory, the next cache line can be pre-visited so that the cache-hit and way-select information can be obtained ahead of time. This information can be used to chip-deselect the missed data array later. Also way prediction mechanism can be utilized in tag pre-visiting scheme to further reduce the number of accesses to the tag array without performance penalty. A way-select circular history buffer (CHB) is maintained to record way-select information of a cache line. Short branches within the address range of the CHB will not require accessing to the tag array and missed data array. This method has been implemented in a 250 MHz high-performance low-power RISC processor.

Key words: cache; low power; CPU; microarchitecture

## 1 引言

功耗是处理器的重要指标,尤其表现在电池供电的便携式设备和应用中. 在现代处理器中,cache 不仅有效地解决了处理器和主存之间的速度匹配问题,而且由于减少了总线活动,对系统的功耗减少也是有益的. 但是由于访问频率高,cache 本身也是处理器的一大耗能部件,cache 的功耗占了处理器功耗的 30 %～60 %[1]. 由于指令 cache 有很高的访问频率,所以有效地降低指令 cache 的功耗对整个处理器的功耗有重要影响.

Cache 的低功耗问题一直是被广泛关注的问题. 通常用来降低 cache 功耗的方法有以下两种,一种是从存储器的结构出发,研发低功耗的存储器,比如在 ARM 上使用的基于 CAM 的 cache 结构,但是 CAM 的流片成本较高;另一种是设法减少对 cache 的访问次数. 本文从减少对 cache 中各存储器的访问频率这一基本思想出发,提出了全新的预访问标志技术和组选择环形历史缓冲区(CHB)技术,并将这两项技术相结合来最大限度地减少对 cache 中各存储器的访问次数.

## 2 组关联 cache 的基本结构

组关联 cache 内部分成 $n$ 个组,每个组包括一个数据存储器和一个标志存储器. 传统的组关联 cache 并行地访问每一组的数据存储器和标志存储器,并进行标志检查. 根据标志检查的结果选择其中一个组的数据存储器的输出. 这种访问

收稿日期:2003-06-09;修回日期:2004-04-16
基金项目:国家 863 重大科技攻关项目 32 位 C-CORE 嵌入式 CPU 开发(No. 2002AA1Z1050)

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

方式被称为 cache 的基本访问方式. 图 1 显示了 cache 的基本结构. 标志检查的结果包括要取的指令是否在 cache 中(cache 命中信息)以及判断指令在 cache 的哪一组中(组选择信息). 在估算 cache 的成本时,标志存储器的成本容易被忽略.

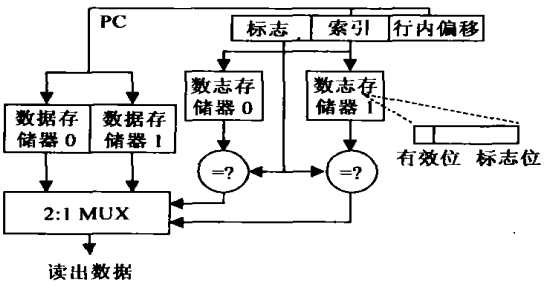



图 1　基本二路组关联 cache 结构

现有的一种减少 cache 访问次数的方法被称为分步 cache[4]. 分步 cache 试图通过先访问标志存储器,再访问数据存储器的方法来减少对数据存储器的访问. 但是这种方法不是在同一时钟周期上访问标志存储器和数据存储器. 这意味着在处理器中需要增加新的流水线节拍,很可能造成性能损失. 文献[5]对要访问的 cache 行所在的数据存储器进行预测,只要预测成功,就可以避免访问不需要的数据存储器,但是如果预测失败,性能和功耗都会遭受损失.

文献[6]通过软件设置控制寄存器,来禁止一个或多个 cache 组,从而在一些对性能要求不高的应用中减少 cache 功耗. 但是在同一个应用中一般不能随意改变设置,而且该方法对软件编程有相当高的要求.

另一种方法是利用 Cache 行缓冲区(cache line buffer)技术. 该方法每次读出一个 cache 行的内容存到行缓冲区中,继续访问同一 cache 行的内容时,就不访问 cache,而从行缓冲区中直接取值. 但是在程序流程跳转时,从 cache 中读出的一部分指令是没有用的,造成功耗损失. 另外,行缓冲区本身也浪费了面积和功耗. 在性能上,数据通路上需要增加一个多路开关(mux)来选择从 cache 还是行缓冲区中取指令,这可能是关键路径. 本文提出的方法能够在不损失性能的前提下,用很少的逻辑来实现大量减少对标志存储器和数据存储器的访问.

## 3　标志预访问技术

### 3.1　基本标志预访问

对于某个指令而言,它只存储在一个数据存储器中,所以在 cache 的基本访问方式中,对其它数据存储器的访问都不是必需的. 如果能够只访问某指令所在的数据存储器,就象直接映射 cache(direct map cache)一样,就可以节省大量功耗. 对指令 cache 而言,由于 80 %左右的程序代码都是顺序执行,可以利用这个特点对行内和相邻行之间的 cache 访问进行优化.

文献[3]提出了改进同一 cache 行内访问的方法. 一个 cache 行一般包括 4 到 16 个指令,或者更多,所有同一 cache 行内的指令的标志信息是相同的. 如果当前访问地址与前次访问地址在同一个 cache 行内,那么当前指令可以共享前一条指令的标志信息. 如图 2 所示的二路组关联 cache 中,每一 cache 行有四条指令,地址连续的两条指令 $i$ 和 $j$ 在同一 cache 行内,指令 $j$ 的 cache 命中信息和组选择信息与指令 $i$ 完全一致. 如果把指令 $i$ 的 cache 命中信息和组选择的信息保存在 cache 命中寄存器和组选择寄存器中,当读取指令 $j$ 时,只需要访问组选择寄存器所指向的那个组的数据存储器就可以了. 也就是说读取指令 $j$ 时,不需要访问标志存储器和其它组的数据存储器,这就可以大大节省 cache 的功耗.

但是如果地址连续的两条指令不在同一 cache 行内(在相邻行内),如图 2 中的 $j$ 和 $k$ 指令. $k$ 可能和 $i$、$j$ 在同一个数据存储器,也有可能在另一个数据存储器,如图中 $k$ 的位置,所以指令 $k$ 无法共享指令 $j$ 的标志信息. 按照传统的实现方法,读取指令 $k$ 时就必须象 cache 的基本访问方式一样,同时访问所有的标志存储器和数据存储器,在所有的数据存储器的输出中选择结果. 事实上,相邻行的访问可以借用 cache 行内访问的思想来减少对数据存储器的访问. 如前面所述,在顺序执行的前提下,在一个 cache 行内只需要访问一次标志存储器,所以在大部分时间内,标志存储器处于空闲状态. 因此在读取当前行的指令时,就有可能和有时间来访问标志存储器,以获得下一 cache 行的标志信息. 那么当读取下一行的指令时,利用提前获得的标志信息就可以只访问目标指令所在的数据存储器,同时禁止所有其它数据存储器的访问. 这种提前访问下一 cache 行的标志存储器的思想可以称为标志预访问技术. 它结合了文献[3]的方法和分步 cache 的思想,节省了功耗,而且不会引起性能损失.

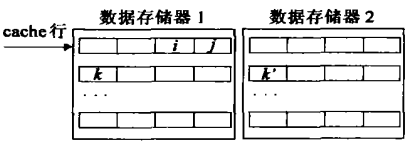



图 2　地址连续的指令在 cache 中的位置

### 3.2　带预测的标志预访问

对标志存储器的预访问使无预测代价的组预测成为可能,从而减少对标志存储器的访问. 在预访问标志存储器时,只访问所预测的那组的标志存储器. 如果访问的结果显示下一行果然在预测的组中,就不必访问其它组的标志存储器. 否则,继续访问其它组的标志存储器.

这种组选择的预测即使失败也不会带来功耗上的损失. 极端的情况是:下一行不在 cache 内部,即 cache 不命中,这时,所有组的标志存储器都会被顺序地遍历一次. 也就是说,预测的最差情况也只是相当于没有组选择预测.

这种组选择的预测也不会带来性能上的损失. 以二路组关联 cache 为例来介绍一种预测策略:如果一个 cache 行包括 4 条指令,在顺序执行时,可以在取当前行的第三个指令的同时,预测并访问下一行的一个组的标志存储器,如果不匹配,则在访问当前行的最后一个指令时,访问另一组的标志存储器. 这样在读取下一 cache 行的指令开始前,下一行的标志信息就已经存在了. 利用这个标志信息就可以用来选择数据存储器,或者在标志存储器都不匹配时,直接进入 cache 替换状态.

预测算法的准确率对所能节省的功耗有很大影响. 对

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

二路组关联 cache ,简单地从信号的概率上而言 ,随机预测的准确率为 50 %. 但是根据对程序行为的观察可得 ,顺序的程序块一般在同一 cache 组中 ,利用本行的组选择结果来预测下一行的组的准确率要远高于 50 %.

## 4 环形历史缓冲区

上述的对 cache 功耗的节省仅仅局限于程序顺序执行的时候. 如果程序控制发生转移 ,就没有时间来预访问标志存储器 ,所以只能用 cache 的基本访问方式来访问 cache ,即同时访问所有的标志存储器和数据存储器. 环形历史缓冲区 (CHB) 被利用来记录以前访问 cache 的组信息 ,从而在发生相对跳转时重用以前访问的信息.

根据跳转指令获取目标地址的方式 ,跳转指令可以分为两类 :绝对跳转和相对跳转. 绝对跳转的目标地址来自于寄存器或内存存储单元. 而相对跳转的目标地址是通过计算当前地址偏移量得到的. 对程序行为的分析结果显示 ,相对跳转指令占了程序的 20 %左右 ,而绝对跳转指令则较少. 在顺序执行的情况下 ,每次访问标志存储器的组选择信息都被顺序存入 CHB :如果 cache 命中 ,存入 CHB 的是当前的组选择信息 ;如果 cache 不命中 ,替换 cache 行 ,则把被替换的组号存入 CHB. 所以 CHB 保留了顺序的几个 cache 行的组选择信息 ,每一项都表示一个 cache 行的组选择信息.

CHB 有一个有效范围 ,该范围是指记录在 CHB 中的顺序的 cache 行所组成的程序块. 如果发生绝对跳转或者相对跳转的目标地址在 CHB 的有效范围以外 ,就清空 CHB ,这是为了保证只要是在 CHB 中的程序块一定是 cache 命中的 ;如果发生相对跳转 ,而且目标地址在 CHB 的有效范围内 ,被称为 CHB 命中 ,那么就可以重用 cache 行内的组选择信息. 这时 ,无需访问索引存储器就可以选中指令所在的数据存储器. 索引存储器和数据存储器的访问次数都被有效地减少了.

如图 3 所示 ,连续的程序段 *A* , *B* , *C* 分别属于三个 cache 行 ,cache 行 *A* , *B* , *C* 的组选择结果已经存入 CHB 中. 如果指令 *i* 是 *B* 中的一个相对跳转指令 ,它有可能在指令流水线的解码节拍或者执行节拍上反馈回取指节拍. 如果是向后跳转的指令 ,而且目标地址落在 *A* 或 *B* 范围内 ,就可以重用 CHB 内的组选择信息和默认的 cache 命中信息. 同理 ,如果是向前跳转指令 ,而且目标地址落在 *B* 或 *C* 内 ,也可以重用 CHB 的信息.

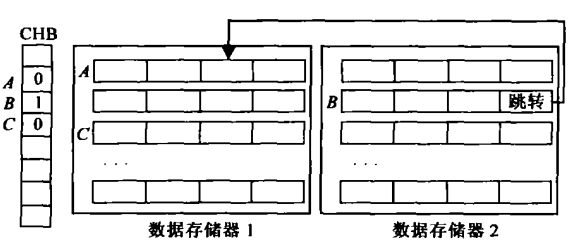



图 3 CHB 工作原理

CHB 通过三个指针来管理 :头指针 ,尾指针和当前指针. 头指针和尾指针界定 CHB 的有效范围 ,当前指针指向当前访问的 cache 行. 如果 CHB 包含 8 个项 ,最多可以记录 8 个顺序的 cache 行 ;对于二路组关联 cache ,每项中只需一位寄存器.

## 5 设计实现

这种集标志预访问 ,组选择预测和组选择历史 (CHB) 的方法已经实现在基于 MCORE 指令集的 CK520 处理器上. CK520 是一种 32 位数据 ,16 位指令的嵌入式处理器 ,采用了二路组关联的 cache ,每个 cache 行包括 8 个 16 位指令. 由于指令的宽度是数据宽度的一半 ,每一组的数据存储器可以分为两个 ,一个存储高 16 位的指令 ,另一个存储低 16 位的指令. 如图 4 所示 ,共有 4 块数据存储器 ,数据存储器 0 ,1 属于组 0 ,数据存储器 2 ,3 属于组 1. 32 位地址线的倒数第二位 (*A*1) 的奇偶属性可以用来选择数据存储器. 这样即使是在基本模式下访问 ,一次也只需访问四个中的两个数据存储器. 该方法可以称作奇偶地址选择法.

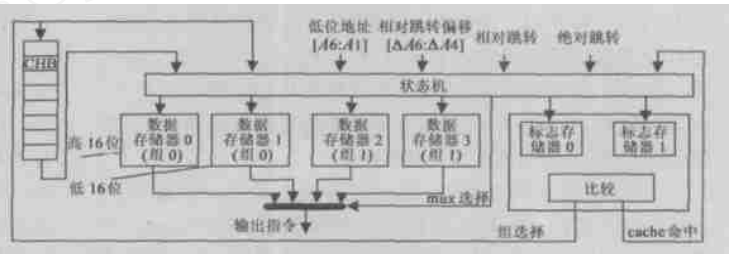
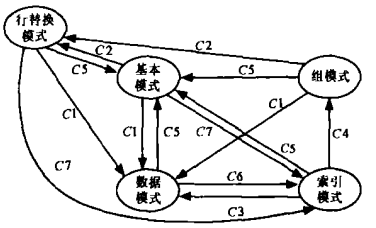



图 4 标志预访问结合 CHB

在 CK520 中采用的提前访问策略是在访问当前行的最后一个指令时 ,预测并访问下一行的一个标志存储器. 针对这种访问策略 ,在控制通路中 ,设置了 5 种访问模式 :基本模式、组选择模式、数据模式、索引模式和行替换模式. 基本模式与传统方法一样 ,即同时访问所有组的索引标志存储器和数据存储器 ,以索引标志存储器的结果来选择数据存储器的输出. 组选择模式只访问一个组的索引标志存储器和数据存储器. 数据模式只访问一个组的数据存储器. 索引模式是在访问当前行的一个组的数据存储器的同时 ,根据预测访问下一行的一个组的索引标志存储器. 行替换模式在 cache 不命中时 ,从主存取数据来替换 cache 行. 控制通路状态机如图 5 所示 ,在状态机中共有五个状态 ,对应五种访问模式.

可以看出 ,如果一行有 8 个指令 ,在顺序执行的情况下 ,第一个指令在预测成功时工作在数据模式 ,在预测失败时工作在组选择模式下 ;取接着的 6 个指令时 ,工作在最省电的数据模式下 ;第 8 个指令工作在索引模式下. 所




*C*1 :cache 命中 & 在本行内 ; *C*2 : 不命中 ; *C*3 组预测命中 ; *C*4 : 组预测不命中 ; *C*5 :绝对位移 | 相对位移离开有效区 ; *C*6 : 下一行不在 CHB 中时 ,进行组选择预测 ; *C*7 : cache 命中 & 对下一行作出预测

图 5 指令 cache 的控制状态机

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

以在顺序执行的情况下,每个 cache 行指令越多,就有越多的指令可以通过最省电的数据模式读取,也就可能越省电.但是更大的 cache 行可能在行替换时花费更多的功耗,在 cache 大小一定时,过大的 cache 行可能引起更多的行替换次数.

## 6　功耗分析

在 CK520 中采用了 8K 的指令 cache.上述几种算法都用 RTL 实现并被综合成门电路进行仿真,采用的是 TSMC0.18 工艺库.表 1 显示了不同 cache 访问算法的指令 cache 模块在运行基准程序时的功耗.

表 1　不同 cache 访问算法的结果(单位:W)

(*a*) 基本 cache 访问算法

| | 原始算法 | 奇偶地址选择法 |
|---|---|---|
| dhry | 0.1023 | 0.0700 |
| gcc | 0.0991 | 0.0678 |
| compress | 0.0987 | 0.0634 |
| tex | 0.1034 | 0.0707 |

(*b*) 优化的 cache 访问算法

| | 减少标志比较法(文献[3]) | 基本预访问和 CHB 结合算法 | 带预测预访问和 CHB 结合算法 |
|---|---|---|---|
| dhry | 0.0404 | 0.0234 | 0.0222 |
| gcc | 0.0396 | 0.0244 | 0.0190 |
| compress | 0.0408 | 0.0256 | 0.0240 |
| tex | 0.0424 | 0.0248 | 0.0231 |

图 6 是各算法的功耗比较.图中结果的显示以原始算法的结果为 1,其它算法的结果都按比例投影到原始算法上.从图 6 中可以看出,运用了带预测的预访问和组历史的访问算法耗费的功耗是原始算法 20 %左右,是减少标志比较法的 50 %左右.

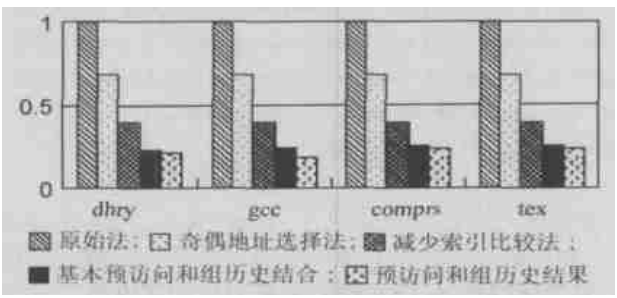



图 6　基准程序在各种访问算法下的功耗比较

综合结果显示,带预测标志预访问和 CHB 结合算法比原始算法多出约 300 门的单元逻辑.

## 7　结论

本文提出的节省 cache 功耗的方法充分利用了指令顺序执行和相对跳转的特性来优化访问指令 cache 的算法.与以前提出的组预测算法完全不同,由于实现该方法所增加的逻辑都位于控制通路,对数据通路没有影响,所以该方法在时序上没有性能损失,而且与取指逻辑相结合的标志预访问思想还可能提高系统取指效率.该方法实现简单,硬件代价小,节省功耗显著,已应用于 250MHz 的 CK520 处理器(采用 TSMC0.18 工艺),该处理器已在 TSMC 流片成功,在 250MHz 下全速运行时,dhrystone number 为 175.

## 参考文献:

[1] R Gonzalez, M Horowitz. Energy dissipation in general purpose microprocessors. IEEE Journal of Solid State Circuits[J]. 1996, 31(9): 1277 - 1284.

[2] E Witchel, S Larsen, C Ananian, K Asanovi'c. Direct addressed caches for reduced power consumption[A]. Proc. of the 34th Int. Symp. On Microarchitecture[C]. Austin, Texas, 2001. 124 - 134.

[3] R Panwar, D Rennels. Reducing the frequency of tag compares for low power i-cache design[A]. Proc of the 1995 Int Symp on Low Power Electronics and Design[C]. New York, 1995. 57 - 62.

[4] J Montanaro et al. A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor[J]. In IEEE ISSCC, 1996, 31(11): 1703 - 1714.

[5] K Inoue, T Ishihara, K Murakami. Way-predicting set - associative cache for high performance and low energy consumption[A]. In Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)[C]. San diego, CA, USA, 1999. 273 - 275.

[6] D H Albonesi. Selective cache ways: On - demand cache resource allocation[A]. Proceedings of the 32nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 32)[C]. Haifa, ISRAEL. 1999. 248 - 259.

[7] B Batson, T N Vijaykumar. Reactive associative caches[A]. Proceedings of the 2001 International Conference on Parallel Architectures and Compiliation[C]. Barcelona, Spain, 2001. 49 - 60.

[8] J Kin, M Gupta, W H Mangione-Smith. The filter cache: An energy efficient memory structure[A]. In Proceedings of the 30th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 30)[C]. Research Triangle Park, North Carolina. 1997. 184 - 193.

[9] S Manne, A Klauser, D Grunwald. Pipline gating: Speculation control for energy reduction[A]. Proceedings of the 25th Annual International Symposium on Computer Architecture[C]. Barcelona, Spain, 1998. 132 - 141.

[10] M Gowan, L Biro, D Jackson. Power considerations in the design of the alpha 21264 microprocessor[A]. In 35th Design Automation Conference[C]. New York. 1998. 726 - 731.

## 作者简介:

**张宇弘**　男,1976 年 9 月生,浙江宁波人,浙江大学生仪学院博士研究生,主要从事嵌入式处理器设计、SOC 设计和低功耗设计方法研究.

**王界兵**　男,1967 年 10 月生,湖北武汉人,美国内华达大学物理学博士,美国斯坦福大学电子工程硕士,主要从事高性能低功耗嵌入式 CPU、网络处理器的设计研究.

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net