·体系结构与软件技术· 文章编号:1000-3428(2015)07-0075-07 文献标识码:A 中图分类号:TP302

嵌入式微处理器中的低功耗Cache技术研究

胡 瑞,马 鹏,章建雄

(中国电子集团公司第三十二研究所,上海,200233)

摘 要: 高速缓冲存储器(Cache)作为微处理器的重要组成部分,在芯片面积和功耗上都占比过高。针对Cache功耗问题,基于分段访问Cache技术和路预测Cache技术,提出一种低功耗组相联Cache的预访问策略。在Cache中增加一个缓冲寄存器(Buffer),用以存储最近Cache命中后被访问的标签和数据子阵列信息。在开始进行标签访问之前,选中该Buffer,并将所访问的Cache标签和Buffer标签进行匹配,根据匹配结果选择采用路预测访问或分段访问方式。通过MiBench基准测试程序并使用SimpleScalar和Sim-Panalyzer进行实验,结果表明,与传统组相联Cache技术相比,该策略能降低25.15%的能量延迟积。

关键词: 低功耗;高速缓冲存储器;多路组相联;路预测;分阶段;预访问

中文引用格式: 胡 瑞,马 鹏,章建雄.嵌入式微处理器中的低功耗Cache技术研究[J].计算机工程,2015,41(7):75-81.

英文引用格式: Hu Rui, Ma Peng, Zhang Jianxiong. Research on Low Power Consumption Cache Technology in Embedded Microprocessor[J]. Computer Engineering,2015,41(7):75-81.

Research on Low Power Consumption Cache Technology in Embedded Microprocessor

HU Rui, MA Peng, ZHANG Jianxiong

(The 32nd Research Institute of China Electronics Technology Group Corporation, Shanghai 200233, China)

【Abstract】 Cache, as an important part of the microprocessor, accounts for an undueproportion of the chip area and power consumption, which is a problem needs to be solved, especially the intensiveenergy cost. Under the background and based on two kinds of micro structural-level low-power optimization techniques for set-associative Cache——phased Cache, as well as the way-predicting one, this paper proposes a low-power set-associative pre-access Cache strategy. In this strategy, Buffer is added in Cache to store the hit tag and data subarray-information from hit Cache. It selects the buffer before accessing the tag, and then matches the tag from two parts——accessed Cache and Buffer. Referring to the matching result, way-predicting or phased Cache can be chosen to access. Experimental results through MiBench benchmarks, Simple-Scalar and Sim-Panalyzer show that Energy-delay Product (EDP) can be reduced by 25.15% in this strategy.

【Key words】 low power consumption; Cache; multi-way set-associative; way-predicting; phased; pre-access

DOI: 10.3969/j.issn.1000-3428.2015.07.014

1 概述

嵌入式处理器是嵌入式系统的核心部件,控制并辅助系统的运行[1]。高性能和低功耗是嵌入式处理器设计的2个重要维度,两者相互制约。目前嵌入式微处理器设计面临的最大挑战是给定电源限制后如何最大化微处理器的性能。近几十年来,处理器工作频率的高速增长与主存储器速度的缓慢增长使得"存储墙"问题日益严重。经统计,处理器的运行速度年均增长约60%,而存储器年均增长仅7%,严重滞后于处理器的增长速度[2]。

高速缓冲存储器(Cache)尽管极大地缓解了处理器与内存速度的差距,但引入的问题是在芯片面积和功耗上都占比过高,Cache的功耗甚至决定着整个嵌入式系统的功耗表现。因此,Cache的低功耗研究一直以来都是处理器微体系结构领域的热点问题,国内外相关研究学者也针对Cache的低功耗设计开展了大量研究工作。

作者简介: 胡 瑞(1991-),男,硕士研究生,主研方向:计算机体系结构,嵌入式系统;马 鹏,高级工程师;章建雄,研究员。

收稿日期: 2014-08-07 **修回日期:** 2014-09-07 **E-mail:** hurui1991@hotmail.com

万方数据

本文研究主存储器中数据块和 Cache 数据块结构间的相关联模式以及多路组相联 Cache 结构，分析分段访问 Cache 和路预测 Cache 技术及其他低功耗技术，在此基础上提出一种组相联 Cache 的预访问策略，并分别从理论和实验两方面给予分析与证明。

2 低功耗 Cache 技术研究

2.1 Cache 基本组织形式及性能分析

Cache 是按块进行管理的，一般来说，主存容量远大于 Cache 的容量，因此，当把一个块从主存调入 Cache 时，需要按照映射规则解决如何放置的问题[3]。主存储器中的数据块与 Cache 数据块结构存在 3 种相关联的映射，包括全相联模式（fully-associative）、直接映射模式（direct-mapped）和组相联模式（set-associative）。

图 1（a）为全相联映射示意图，在该映射模式下，主存中的任意一块数据都可以放到 Cache 中任意一行上；图 1（b）为直接映射示意图。直接映射将主存储器中的每块数据映射到固定的一个 Cache 行中，即主存中的任意一个数据块跟 Cache 的某一行是一一对应的；组相联映射[4]示意图如图 1（c）所示，它将 Cache 分成大小相同的组，每个组内所包含的每一块成为一路。主存中的任意一块可以放到唯一的一个组中的任意一块。组相联映射的优势在于兼得直接映射和全相联映射两种设计的优点，即结构相对简单而且 Cache 命中率高。目前 Cache 的设计基本都是采用组相联映射方式。

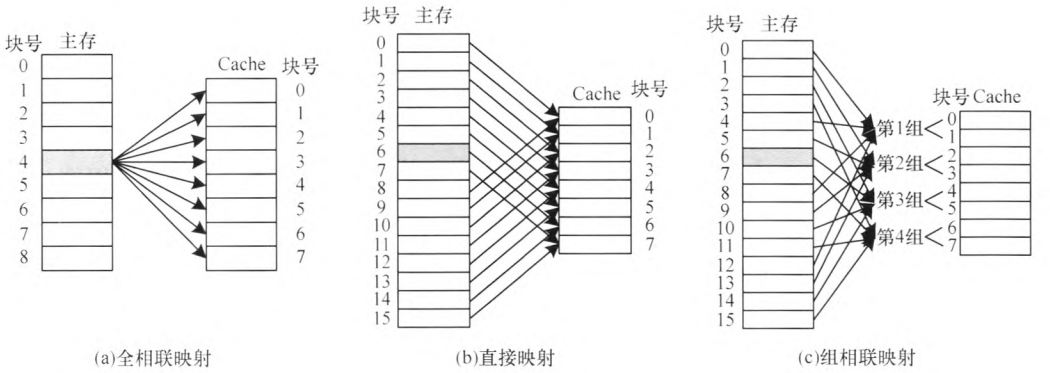

图 1 Cache 组织方式

按照组中路的多少分为 2 路组相联、4 路组相联等，多路组相联的结构如图 2 所示。

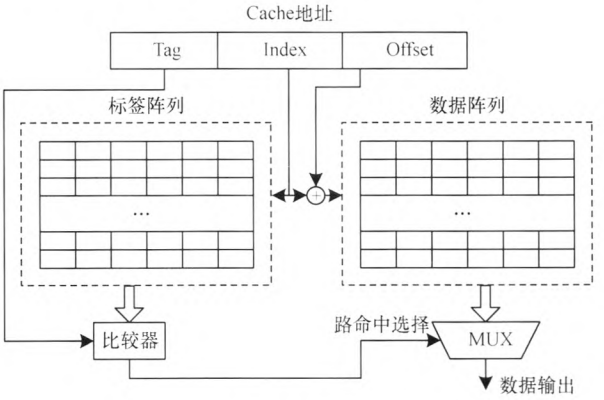

图 2 多路组相联 Cache 结构

在传统的组相联 Cache 访问过程中，当前被访问的 Cache 组中的所有 Cache 块的标签阵列和数据阵列同时被访问，如图 3 所示。然后再根据标签比较的结果从所有读出的数据阵列中选择命中的数据子阵列。无论命中还是失效，所有路都是活动的，而且对 Cache 的访问可以在一个周期内完成。

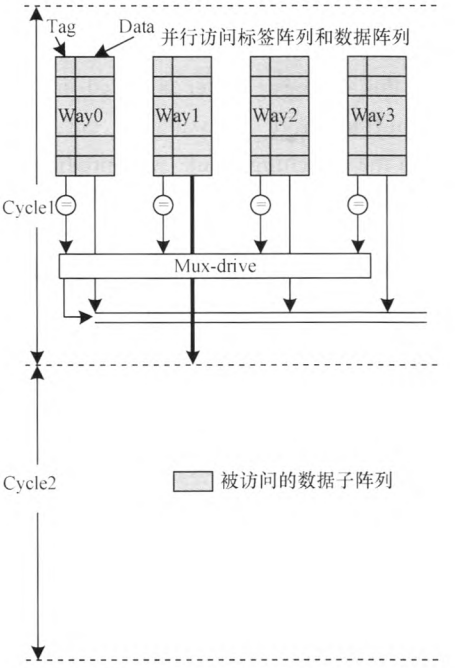

图 3 传统 4 路组相联 Cache

访问一个组相联 Cache 的整个能量消耗可以近似表示为：

万方数据

E_{Decode}：驱动地址总线并译码 Memory 地址的能量消耗；

E_{Memory}：访问标签子阵列和数据之阵列，主要是驱动字线和位线，刺激敏感放大器等消耗的能量；

$E_{I/O}$：当置换发生时驱动外部 I/O 引脚的能量消耗。

因为 E_{Decode} 与 E_{Memory} 相比非常小，而 $E_{I/O}$ 在高命中率的 Cache 中影响很小，所以 E_{Cache} 可以用式(1)简单表示：

$$E_{Cache} = E_{Memory} \tag{1}$$

$$N_{Tag} \times E_{Tag} + D_{Data} \times E_{Data} \tag{2}$$

其中，N_{Tag}，N_{Data} 分别表示当 Cache 进行访问时，访问 Tag 子阵列和 Data 子阵列的数量；E_{Tag}，E_{Data} 分别表示访问一个 Tag 子阵列和 Data 子阵列的能量消耗。

因此，传统四路组相联 Cache(4SACache)平均一次访问的总能量消耗($E_{4SACache}$)和平均 Cache 访问时间($T_{4SACache}$)表示如下：

$$E_{4SACache} = 4E_{Tag} + 4E_{Data} \tag{3}$$

$$T_{4SACache} = 1 \tag{4}$$

目前针对 Cache 的微体系结构级动态功耗优化技术，最经典的是分段访问 Cache(Phased Cache)和路预测 Cache(Way-Predicting Cache)2 种技术。

这两种方法都是面向多路组相联 Cache 架构，通过预测减少每次 Cache 访问过程中的 Cache 被访问的标签子阵列和数据子阵列的数目，降低 Cache 的动态访问功耗。下文将详细介绍这两种架构的 Cache。

2.2　分段访问 Cache

为了避免激活不必要的数据路，文献[5]提出了一种分段访问组相联 Cache(Phased Set-Associative Cache，PSA Cache)。

分段访问 Cache 访问过程如图 4 所示。分段访问 4 路组组相联 Cache 的原理如下：Cache 中的标签阵列和数据阵列分别在 2 个不同的流水线中被访问；整个访问过程需要 2 个周期：第 1 周期，被访问组中的标签位并行检测，这一段没有数据访问发生；第 2 周期，获得路命中信息，如图 4 中的 Way1，则该路执行数据访问，减少了剩下路的数据访问能耗。显然，PSACache 的低能耗是以访问时间的增加为代价的，虽然分段访问需要经过 2 个周期才能获得所需数据，但是现代处理器中使用的流水线重叠技术可以一定程度上隐藏所增加的访问时间。

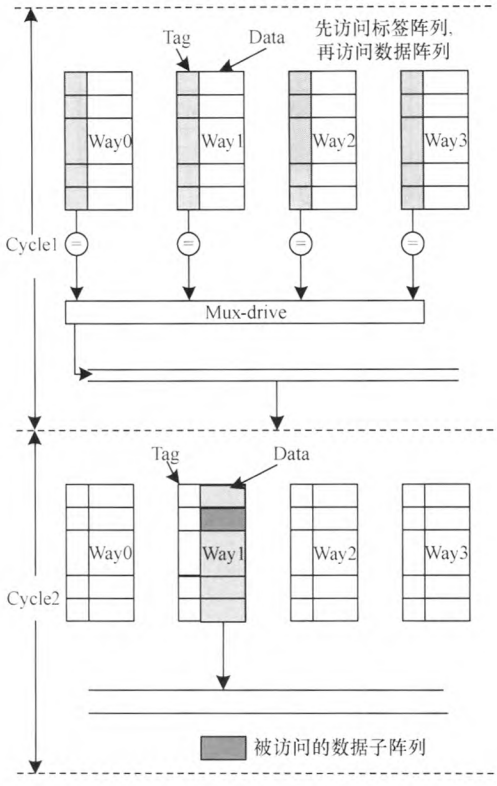

图 4　分段访问 4 路组相联 Cache 过程

因此，分段访问四路组相联 Cache(P4SACache)平均一次访问总的能量消耗($E_{P4SACache}$)和访问时间($T_{P4SACache}$)表示如下：

$$E_{P4SACache} = 4E_{Tag} + CHR \times E_{Data} \tag{5}$$

$$T_{P4SACache} = 1 + CHR \times 1 \tag{6}$$

其中，CHR 表示 Cache 命中率。

2.3　路预测 Cache

在 PSACache 的基础上，文献[6]提出了基于组相联 Cache 的路预测算法。路预测 Cache 的结构如图 5 所示。路预测 Cache(Way-predicting Set-associative Cache，WPSA Cache)使用了一个 MRU(Most Recently Used)算法，根据程序访问的局部性原理，MRU 记录每一个 Cache 组的路命中信息，作为对该组再次被访问时的路预测信息，并刷新 MRU 表。每次 Cache 在进行正常的访问之前根据路预测信息，推断的选择一路，如图 5(a)所示先访问预测的路。如果预测正确，称为预测命中(Predict Hit)，Cache 访问将成功实现，此时路预测 Cache 仅仅消耗预测路的活动能量而且整个访问过程可以在一个周期内完成；如果预测失效(Predict Miss)，则进行如图 5(b)所示并行访问剩下的路，这时需要 2 个连续的周期，增加了 Cache 访问时间；因为所有剩下的路的访问将跟传统组相联 Cache 的规则一样，这时预测路 Cache 将不能减少能量消耗。

万方数据

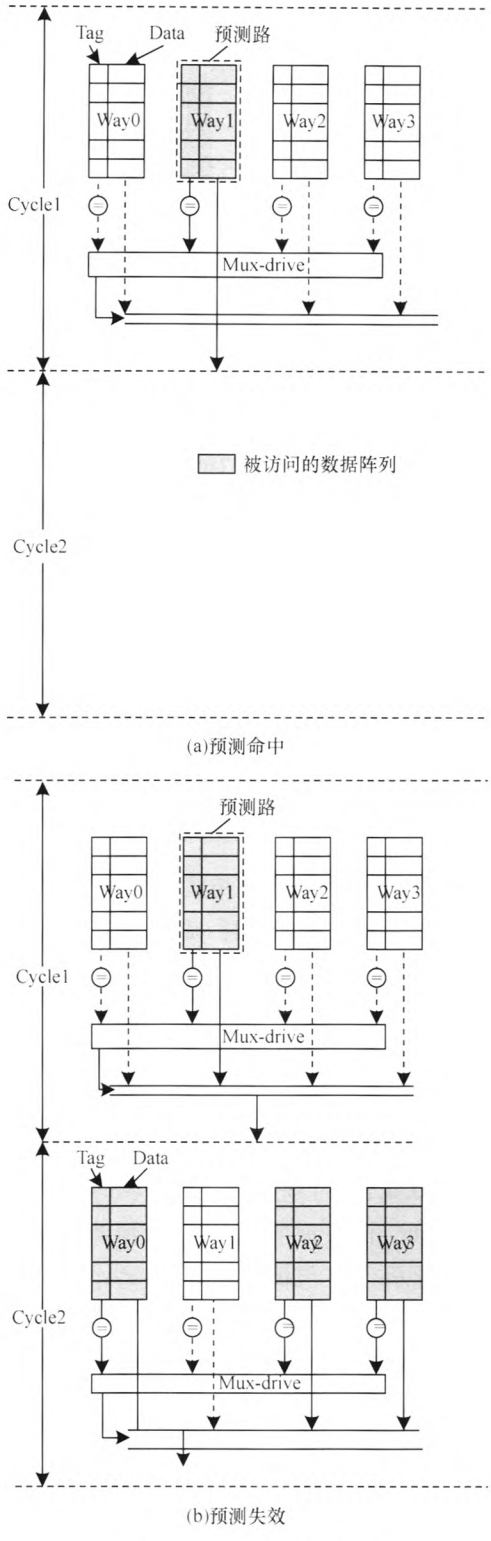

图 5 路预测 Cache 结构

实验结果表明:基于 MRU 的路预测 Cache 的预测正确率高于 90%(在指令 Cache 和数据 Cache 中分别是 96% 和 86%);与传统的组相联 Cache 相比,路预测 Cache 节省了 60% ~70% 的功耗,但有 10% 的性能损失[6]。

所以,路预测四路组相联 Cache(WP4SACache)平均一次访问总的能量消耗($E_{WP4SACache}$)和访问时间($T_{WP4SACache}$)表示如下:

$$E_{WP4SACache} = (E_{Tag} + E_{Data}) + (1 - PHR) \times (3E_{Tag} + 3E_{Data}) + (1 - PHR) \quad (7)$$

$$T_{WP4SACache} = 1 + (1 - PHR) \times 1 \quad (8)$$

其中,PHR 是预测命中率。

2.4 其他低功耗技术

文献[7]针对低功耗 Cache 设计,提出一种模式预测方案,并给出了 4 种访问模式的预测器:通过预测器判断当前指令是否适合采用路预测技术,若适合则采用;否则,采用 Phased Cache 方式访问。文献[8]提出一种针对指令 Cache 基于减少标签检验的低功耗优化技术:通过使用编译器标出长度低于指令 Cache 大小的环线并添加一些简单的逻辑电路去控制对标签阵列的访问,这能够减少不必要的标签检验并节省指令 Cache 的能量消耗。文献[9]设计了一种基于预缓冲机制的指令 Cache:通过预缓冲控制布局的预测,使处理器需要的指令尽可能在缓冲区命中,从而避免访问指令 Cache 所造成的功耗。文献[10]提出一种低功耗动态可重构 Cache 算法:该算法根据指令时间数监测程序段的变化,确定容量调整,在程序段内,状态机根据平均访问时间对 Cache 的访问进行预测,然后根据预判的结果确定当前程序段的 Cache 结构。文献[11]为低功耗嵌入式处理器设计一种基于行为的可配置 Cache;文献[12]使用有效位预判决为并行架构设计一种低功耗路预测 Cache。

3 组相联 Cache 的预访问策略

3.1 访问策略介绍

本文基于以上 2 种组相联 Cache 的低功耗优化技术的优势提出一种组相联 Cache 的预访问策略(Pre-access Set-associative Cache,PASA Cache)。该策略相对于分段访问 Cache 和路预测 Cache,不但能节省功耗还能减少访问时间,但是会增加很小的硬件开销。具体的访问流程如图 6 所示。

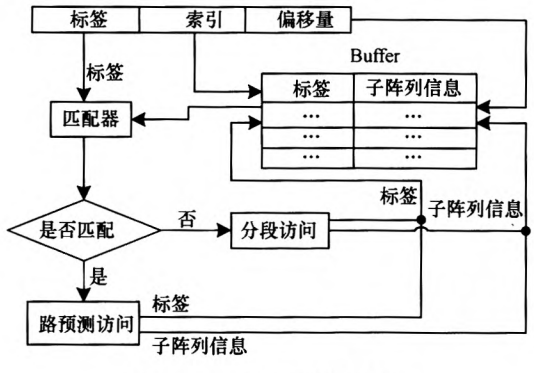

图 6 Cache 预访问流程

通过借鉴路预测 Cache 访问方式,该策略在 Cache 中增加一个缓冲寄存器(Buffer),用来存储最

万方数据

近 Cache 命中后被访问的标签和 Way 信息。最近访问的 *Tag* 和被访问的子阵列信息存储在这个 Buffer 中。在一个 4 路组相联 Cache 中,在开始进行标签访问之前,选中这个 Buffer,并将所访问的 Cache 标签和 Buffer 中的标签进行比对匹配,若匹配结果正确,则说明所需要的数据在匹配标签对应的 Way 中,那么可以进行路预测访问,如图 7 所示;若匹配不正确,则根据程序的局部性原理,将要访问的数据很有可能不是最近访问过的,如果这时继续使用路预测方式进行访问,最后的 Cache 访问很可能是 Miss 的,那么可以采用分段访问方式,虽然需要近 2 个周期,但是保证了 Cache 命中率,如图 8 所示。

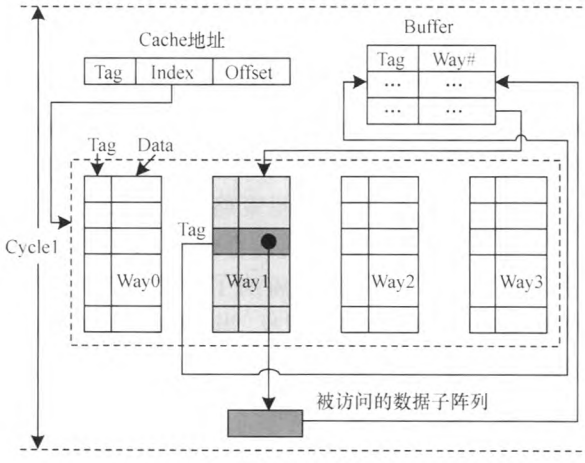

图 7　路预测访问

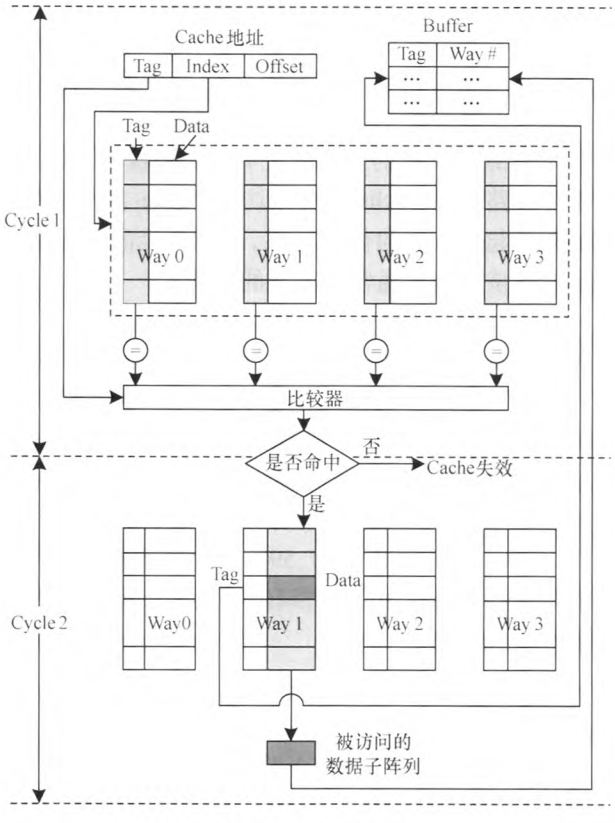

图 8　分段访问

而且无论采用哪种方式进行访问,访问结束后,都需要把这次访问的标签和 Way 信息储存到 Buffer 中并按照 MRU 置换算法剔除最长时间未使用的标签和 Way 信息。

3.2　性能分析

由以上分析得知:传统路预测访问策略在开始访问前读取 MRU 信息可能会使 Cache 访问时间增长,但是同分段访问一样,可以通过在更早的流水线级计算组索引地址来隐藏访问时间。与分段访问 Cache 相比,最大的优势就是在预测正确的情况下减少了一个周期的访问时间,因为消除了一个路选择的延迟。当然,它的缺点就是对性能的影响很大程度上取决于路预测命中率的大小,因为在预测失效的情况下不仅增加了访问时间而且访问能耗与传统组相联 Cache 相比也没有减少。而且传统的路预测访问在局部性不好的程序中会起到增加性能开销的反作用。

本文通过使用 Buffer 来降低这种失效开销,因为在进行整个 Cache 访问之前预先通过对两类标签的匹配比较,只有在有匹配后才进行路预测访问,将对预测命中率的依赖降到最低,预测失效产生的功耗损失也降到最低。而且如果标签不匹配,通过进行分段访问,它消耗的能量要比路预测在预测失效情况下消耗的能量低,因为分段访问在第 2 个周期仅仅需要访问一路数据阵列,而路预测访问在第 2 个周期则要访问除预测路以外剩下的所有路的数据阵列。所以通过预测性的选择一种方式去进行访问将比单独的采用分阶段访问或路预测访问对 Cache 的整体性能提高更有效。

由于指令 Cache 中数据块的局部性非常明显,复用性强,刚刚被访问的数据块可能马上又会被访问,因此指令 Cache 访问的可预测性比数据 Cache 和二级 Cache 都要强。所以理论上该访问策略对指令 Cache 的性能提升要比对数据 Cache 的性能提升高。

3.3　功耗和访问时间分析

如图 5 ~ 图 8 所示,预访问四路组相联 Cache (PA4SACache)平均一次访问总的能量消耗(E_{Cache})和访问时间(T_{Cache})表示如下:

$$E_{\text{PA4SACache}} = (E_{\text{Tag}} + E_{\text{Data}}) \times THR + (1 - THR) \times (4E_{\text{Tag}} + CHR \times E_{\text{Data}}) + E_{\text{Buffer}} \quad (9)$$

$$T_{\text{PA4SACache}} = (1 + (1 - PHR) \times 1) \times THR + (1 - THR) \times (1 + CHR \times 1) + T_{\text{Buffer}} \quad (10)$$

其中,THR 是访问开始时 Cache 中标签和 Buffer 中标签匹配成功率;E_{Buffer} 是读取 Buffer 中标签消耗的能量与访问结束后将信息存储进 Buffer 中消耗的能量之和;T_{Buffer} 是指 Cache 访问前 Cache 中标签和

万方数据

Buffer 中标签匹配所需时间,这里都取平均值。

4 实验及结果分析

由于传统的 SimpleScalar 不支持对功耗的分析,本文采用扩展的 Sim-Panalyzer,并从 MiBench 测试集中选择 10 个基准测试程序进行实验。

为了评估该访问策略的性能,对于以上提到的 4 种采用不同访问策略的 Cache,本文选取 Cache 块、Cache 行块和相联度分别为 16 KB、32 Byte、和 4 Byte。文献[6]中已证明:$E_{Tag}=0.078E_{Data}$,而且分析可知,E_{Buffer}和 T_{Buffer}相对于 $E_{PA4SACache}$和 $T_{PA4SACache}$小得多,可忽略不计。选择传统四路组相联 Cache 作为基准,路预测组相联 Cache 和分段访问组相联 Cache 作为参照。进行实验的 Cache 模型在容量、相联度以及其他外部环境上均保持一致,因此具有一定的可比性。为了便于比较各 Cache 访问模型的访存功耗和延时,这里采用归一化的方法,将传统四路组相联 Cache 作为归一化标准,表中的数值都是相对值,无单位;其中,路预组相联 Cache 的归一化值参考文献[13]。最后以平均能量延迟积(Energy-Delay Product,EDP)作为性能评价指标。各访问模型的访问功耗和访问时间以及 EDP 值分别如表 1 ~ 表 3 所示。

表 1 平均访问能耗

测试程序	路预测组相联 Cache	分段访问组相联 Cache	预访问组相联 Cache
Susan	0.678 9	0.512 7	0.597 4
Bitcount	0.749 3	0.474 6	0.640 2
Dijkstra	0.748 8	0.455 3	0.632 6
Patricia	0.616 2	0.461 6	0.535 9
Sha	0.683 6	0.501 5	0.585 4
Djpeg	0.740 5	0.478 4	0.599 3
Cjpeg	0.663 9	0.505 2	0.587 2
Lame	0.698 2	0.518 4	0.602 9
Mad	0.715 9	0.488 2	0.568 0
Fft	0.647 2	0.493 9	0.547 9

表 2 平均访问时间

测试程序	路预测组相联 Cache	分段访问组相联 Cache	预访问组相联 Cache
Susan	1.137	1.824	1.227 6
Bitcount	1.134	1.869	1.255 9
Dijkstra	1.144	1.871	1.306 3
Patricia	1.149	1.848	1.249 7
Sha	1.138	1.892	1.262 6
Djpeg	1.154	1.901	1.314 5
Cjpeg	1.157	1.850	1.305 9
Lame	1.128	1.864	1.245 0
Mad	1.144	1.893	1.295 7
Fft	1.146	1.797	1.224 4

表 3 平均能量延迟积

测试程序	路预测组相联 Cache	分段访问组相联 Cache	预访问组相联 Cache
Susan	0.771 9	0.935 1	0.733 4
Bitcount	0.849 7	0.887 0	0.804 0
Dijkstra	0.856 6	0.851 9	0.826 4
Patricia	0.708 0	0.853 0	0.669 7
Sha	0.777 9	0.948 8	0.739 1
Djpeg	0.854 5	0.909 4	0.787 8
Cjpeg	0.768 1	0.934 6	0.766 8
Lame	0.787 6	0.966 3	0.750 6
Mad	0.819 0	0.924 2	0.736 0
Fft	0.741 7	0.887 5	0.670 8

分析式(3)~式(10)可知:在同样的 Cache 命中率(*CHR*)和预测命中率(*PHR*)情况下,当访问开始时 Cache 中标签和 Buffer 中标签匹配成功率 $THR=0$(即所有的标签匹配均不成功,此时每次都只进行分段模式访问)时,采用预访问策略的 Cache 平均访存能耗等于分段访问 Cache 的访存能耗,但是访存时间与路预测 Cache 相等,这时解决了分段访问 Cache 会增加的访存时间的问题;相反,当 $THR=100\%$(即所有的标签匹配均成功,此时每次都只进行路预测访问)时,此时采用预访问策略的 Cache 平均访问能耗与传统 Cache 相比节省了 75%,小于路预测组相联 Cache 的访存能耗(因为在指令 Cache 和数据 Cache 中分别是 96% 和 86%),但是访存时间与路预测 Cache 相当,远小于分段访问 Cache,这时解决了传统路预测访问策略在整个预测失效后产生的功耗和延迟开销且太过依赖于预测命中率的问题。总的来说,当采用预访问策略的 Cache 工作在2 种极端情况下,对于传统路预测 Cache 和分段访问 Cache 各自存在的问题都能起到一定的优化效果,下面实验的结果分析也能验证该策略在通常情况下的表现。

实验结果显示:对选取的基准测试程序而言,采用预访问策略的组相联 Cache 与传统组相联 Cache 相比,能节省 41.03% 的能耗但是增加了 24.63% 的访问时间,总体性能指标 EDP 值提高了 25.15%;而且相对于路预测组相联 Cache,访问能耗降低了14.99%,EDP 提高了 5.67%;相对于分段访问,访问时间降低了 31.82%,EDP 提高了 17.59%。这说明在综合考虑性能和功耗的情况下,预访问组相联 Cache 是三者中优化程度最高的。

5 结束语

本文利用当前低功耗 Cache 研究中的 2 种经典的设计技术:分阶段访问 Cache 技术和路预测 Cache 技术,提出一种基于预测的采用预访问策略的低功

万方数据

耗 Cache 设计策略。该策略不仅结合了传统的路预测 Cache 和分段访问 Cache 的优势,而且在一定程度上能解决其各自存在的最大问题。实验结果也证明相对于以上 2 种组相联 Cache,本文提出的预访问策略在 EDP 上均有提高,而与传统组相联 Cache 相比,虽然增加了 24.63% 的访问时间,但是能节省 41.03% 的能耗,总体性能指标 EDP 值提高了 25.15%。

由于实验的局限性,并不能对每一类程序都进行实验测试,具体应用到实际环境中的效果如何是下一步研究的方向。通过实验数据能看出:本文访问策略虽然能降低功耗,但是增加的访问时间在某些要求严格的处理器中是不允许的,通过采用流水线重叠方式隐藏增加的访问时间也是今后研究工作的重点之一。而且,因为本文提出的访问策略需要在 Cache 中增加一个 Buffer,在一个 16 KB 四路组相联 Cache 中,Buffer 需要 4 KB 的大小,并且标签匹配器也增加了一定的硬件开销,所以如何降低这些硬件开销也是下一步研究的内容。此外,本文只是对 Cache 的访问方式进行研究,低功耗 Cache 还有很多研究方向,因此,同时也将对其他主要设计方法进行分析,并结合各方优点进行改进。

参考文献

[1] 张承义.超深亚微米微处理器漏流功耗的体系结构级优化技术研究[D].长沙:国防科学技术大学,2006.

[2] Henessy J L,Patterson D A.计算机系统结构:量化研究方法[M].3 版.郑纬民,译.北京:电子工业出版社,2004.

[3] 彭蔓蔓.体系结构级低能耗 Cache 和动态电压缩放技术研究[D].长沙:湖南大学,2007.

[4] 姜难难.嵌入式处理器中 Cache 的研究与设计[D].哈尔滨:哈尔滨工业大学,2009.

[5] Hasegawa A,Kawasaki I,Yamada K,et al. SH3:High Code Density,Low Power[J].IEEE Micro,1995,15(6):11-19.

[6] Inoue K, Ishihara T, Murakami K. Way-predicting Set-associative Cache for High Performance and Low Energy Consumption[C]//Proceedings of ISLPED'99. San Diego, USA:IEEE Press,1999:273-275.

[7] Zhu Zhichun,Zhang Xiaodong. Access-mode Predictions for Low-power Cache Design[J]. IEEE Micro,2002,22(2):58-71.

[8] Li Quanquan, Bao Lidan, Zhang Tiejun, et al. Low Power Optimization of Instruction Cache Based on Tag Check Reduction [C]//Proceedings of ICSICT'12. Washington D.C.,USA:IEEE Press,2012:1-3.

[9] 王　冶,张盛兵,王党辉.基于预缓冲机制的低功耗指令 Cache[J].计算机工程,2012,38(1):268-272.

[10] 任小西,刘　清.一种低功耗动态可重构 Cache 算法的研究[J].计算机应用研究,2013,30(2):414-416.

[11] Ye Jiongyao,Jin Jiannan,Watanabe T. A Behavior-based Reconfigurable Cache for the Low-power Embedded Processor[C]//Proceedings of ASICON'11. Washington D.C., USA:IEEE Press,2011:1-5.

[12] Chen Hsin-Chuan,Chiang Jen-Shiun. Low-power Way-predicting Cache Using Valid-bit Pre-decision for Parallel Architectures[C]//Proceedings of Conference on Advanced Information Networking and Applications. Washington D.C.,USA:IEEE Press,2005:203-206.

[13] 彭　方.路预测与可重构 Cache 的自适应低能耗算法研究[D].长沙:湖南大学,2008.

编辑　金胡考

(上接第 74 页)

参考文献

[1] Smith J E,Nair R. 虚拟机系统与进程的通用平台[M].安　虹,译.北京:机械工业出版社,2009.

[2] Jones R,Hosking A,Moss E. The Garbage Collection Handbook: The Art of Automatic Memory Management[M].[S.1.]:Chapman & Hall/CRC,2011.

[3] Xu Xiaoliang,Shen Jiang. Research on Stack-allocation Based JVM Garbage Collection[J]. Advanced Computer Theory and Engineering,2010,(2):346-349.

[4] Liu S,Tang J,Wang L,et al. Packer: Parallel Garbage Collection Based on Virtual Spaces[J]. Computers, 2012,61(11):1611-1623.

[5] Libic P,Bulej L,Horky V,et al. On the Limits of Modeling Generational Garbage Collector Performance [C]// Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering. New York,USA: ACM Press,2014:15-26.

[6] Zhao X,Nakamura S,Nakagawa T. Optimal Tenuring and Major Collection Times for a Generational Garbage Collector [J]. Asia-Pacific Journal of Operational Research,2012,29(3):50-59.

[7] 郑　磊,张欢欢,李建华.基于区域标记的动态分代垃圾收集算法[J].华东理工大学学报:自然科学版,2011,37(2):218-223.

[8] 吴志军,何加铭,曾兴斌.基于嵌入式 Java 虚拟机的垃圾收集优化算法[J].计算机工程,2012,38(7):46-48.

[9] Singer J,Brown G,Lujan M,et al. Towards Intelligent Analysis Techniques for Object Pretenuring [C]// Proceedings of the 5th International Symposium on Principles and Practice of Programming in Java. New York,USA:ACM Press,2007:203-208.

[10] Li Jingmei, Zhang Bo. Garbage Collection Algorithm Based on Embedded KVM[J]. Multimedia Technology, 2011,11(1):3535-3538.

[11] IBM. Jikes RVM User Guide[EB/OL]. (2013-10-11). http://wwwjikesrvm.org.

[12] Hu G,Chai Z,Zhao W,et al. Towards Garbage Collection Mechanism for RTSJ-oriented Embedded Java Processor[C]//Proceedings of Computer and Information Technology. Washington D.C.,USA:IEEE Press,2010: 1312-1316.

编辑　索书志

万方数据