文章编号:1001-2486(2011)06-0017-07

Cache 漏流功耗的自适应优化:动态容量调整*

张承义,郭 维,周宏伟
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:当集成电路制造工艺水平发展到超深亚微米阶段,漏流功耗所占的比例越来越大,成为微处理器功耗的重要来源。漏流功耗同电压、漏电流和晶体管数量等因素密切相关。Cache 是微处理器中面积较大的部件,对其漏流功耗进行优化是微处理器低功耗设计的首要任务。除了采取工艺上的改进措施外,cache 漏流功耗可以通过把握或改变 cache 的工作状态来进行体系结构级的自适应优化。提出了基于"逻辑路"的 cache 动态容量调整策略。模拟结果显示,在相联度较高的 cache 中,基于"逻辑路"的动态容量调整策略可以在几乎不影响性能的前提下,将 cache 的漏流功耗降低约 76.6%。

关键词:微处理器;高速缓冲存储器;漏流功耗;容量调整
中图分类号:TP302.7 **文献标识码**:A

Dynamic Resizing: Adaptive Optimization for Cache Leakage Power

ZHANG Cheng-yi, GUO Wei, ZHOU Hong-wei
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: The power leakage covers more and more of the consumption of power, especially when the production of highly integrated circuit has reached the level of very deep submicron, thus it becomes the main source of the power leakage of the microprocessor. Power leakage is closely related to voltage, leakage current and the amount of transistors. Cache is the sizable fraction of the total microprocessor, and its leakage power optimization must be firstly considered in low power microprocessor design. Besides process improvement, the leakage power of caches can be adaptively reduced by monitoring and controlling its operating states at architectural level. In light of this idea, a dynamic resizing policy based on cache replacement algorithm was proposed. The cache was dynamically resized on so-called logical way granularity according to cache operating states. Simulation results show that dynamic resizing policy can reduce cache leakage power by 76.6% without obviously performance drop, especially for high associative caches.

Key words: microprocessor; cache; power leakage; dynamic resizing

微处理器的功耗可以分为动态功耗和静态功耗(漏流功耗)两部分。在 CMOS 电路中,无论电路处于什么状态,漏电流总是存在。长期以来,动态功耗一直占据总功耗的绝大部分,漏流功耗往往可以忽略不计,但随着集成电路设计进入超深亚微米阶段,电源电压和阈值电压逐渐下降、短沟道效应以及隧道效应等多种原因导致漏电流成指数增长,漏流功耗在总功耗中所占的比例越来越大。例如在 FT1000 处理器的通用处理器核中,漏流功耗占到了一个处理器核总功耗的 30%。这种情况下,如果不对漏流功耗加以控制,将严重阻碍微处理器性能的进一步提升[1]。

漏流功耗与电压、漏电流和有效晶体管数量密切相关。电压和漏电流可以通过改进集成电路工艺的方式进行改进。有效晶体管数量是指能够产生漏电流的晶体管数量,可以结合电路和工艺技术控制某些晶体管动态地进入低漏流态或无漏流态,等效为降低了有效晶体管的数量,这视为体系结构级的漏流优化技术。Cache 是微处理器中面积最大的部件之一,同时也是漏流功耗最大的部件之一,特别是在高性能微处理器中,为了缓解存储墙问题,片上 cache 做得越来越大(大容量)、越来越宽(高相联度)、越来越深(多层次),因此研究针对片上 cache 的漏流功耗优化技术具有十分现实的意义。

降低漏流功耗可以从工艺级、电路级和体系

* 收稿日期:2011-06-20
基金项目:国家自然科学基金资助项目(60970036);国家 863 高技术资助项目(2009AA01Z124);国家"核高基"重大专项"超高性能 CPU 新型架构研究"资助项目(2011ZX01028-001-001)
作者简介:张承义(1978—),男,副研究员,博士。

结构级等多个层次来实现,在体系结构级进行cache的漏流功耗控制可以在较大的粒度进行,同时还可以综合考虑处理器运行的实时信息,且与其他级别的漏流优化措施正交,具有较好的经济性和灵活性。所有的体系结构级控制技术本质上都是利用在体系结构级所能看到的程序执行信息和处理器运行状态来控制电路进入低漏流的工作模式,从而降低漏流功耗,因此低漏流工艺和电路技术是体系结构级漏流控制的基础。目前已提出了多种低漏流电路技术,当应用于cache等存储电路时,可以分为状态丢失和状态保持的低漏流电路技术。门控电压技术(Gated-vdd)[2]是一种状态丢失的低漏流电路技术,通过一个睡眠晶体管将存储单元与电源或地断开,可以将单元的亚阈值漏电流降至最低,但是保存的内容丢失。目前有多种体系结构级控制策略以门控电压电路为基础,如cache衰退(decay)[3],被门控的存储单元在重新访问时需要进行数据重载,从下级cache重新取数(导致额外的动态功耗)。状态保持的低漏流存储电路如DVS(Dynamic Voltage Scaling)[4]和ABB多阈值电压CMOS电路[5]能够在降低漏流的同时保持存储单元内容,漏流优化能力较弱但能避免额外的数据重载导致的动态功耗。昏睡(drowsy) cache[6]就是利用DVS电路的体系结构级漏流控制策略,周宏伟[16]还研究了采用路预测技术对drowsy cache的路进行预先唤醒的机制。也有研究试图将两种低漏流电路技术结合使用,但这增加了工艺制备的难度,且需要更为复杂的控制策略[7]。此外,面向低功耗设计的一些动态cache分区策略[14]也为cache的漏流功耗优化提供了一些新的研究思路,但如何分区最为有效一直是研究人员竞相研究的难题。

本文提出了一种考虑cache工作状态的自适应动态cache容量调整策略,根据cache替换策略的记录信息,将cache分为多个"逻辑路"(非物理组织上的路),以"逻辑路"为粒度对cache容量进行调整。对于较高相联度的片上cache,能够取得较好的漏流功耗优化效果,而对性能的影响可以忽略不计。本文的研究对象为片上L1D cache,但所提算法可以扩展到片上的其他组相联cache。

1 L1D cache的访问特性

Cache是为了缓解"存储墙"问题而在存储器与寄存器之间增加的高速缓冲部件,L1D cache作为指令流水线的一部分,直接提供指令执行所需的数据,对处理器的访存性能具有重要影响。但研究表明,任意时刻L1D cache中平均有超过80%的数据块在被替换之前都不会被再次访问,即进入了"衰退期"[3]。我们针对64KB的4路组相联数据cache(块大小32字节)中数据块的访问情况进行了统计,如表1和表2所示[15]。数据cache中每个实例(即具有有效数据的cache块)的平均衰退期都在10k个时钟周期以上,最大的达到7M个时钟周期,这说明cache中数据块的生命周期的大部分时间都浪费在了衰退期内,处于一种"无为"状态,这为漏流功耗的优化提供了空间。而且从统计数据可以看出,平均访问间隔和平均衰退期具有明显的数量级的差别,通过历史信息可以很容易将它们区别开来。但不同的应用程序之间的访问间隔和衰退期不易通过简单的阈值来区别。

假定我们研究的微处理器包含片上二级cache,且二级cache对一级cache是包容的。这种假设下数据cache块被关断时仍在二级cache中有备份,失效时可以从二级cache中获得数据,因此相对于片外访问延迟来说开销较小。同时现代微处理器中非阻塞cache、前瞻执行、值预测以及多线程等技术的应用,少量的cache失效开销可以通过这些技术来隐藏。因此,为了更大力度地降低漏流功耗,我们在研究针对数据cache的漏流功耗控制技术时基于状态丢失的低漏流存储技术(如门控电压),即关断的cache块重新访问时需要进行数据重载。其本质是研究在何时、以何种粒度关断cache块最为合适。相应地,对于片上最后一级cache,由于失效时需要进行片外的存储访问,延迟较大,其漏流控制时则宜基于状态保持的低漏流存储电路,即关断的cache块重新访问时不需重载,但需要一个唤醒周期。

表1 数据cache中连续两次访问同一数据块的时间间隔(节拍)

Tab. 1 Intervals of access to the same data cache line (in cycles)

测试程序	最大访问间隔	平均访问间隔
applu	89912	436
art	184709	179
bzip2	2331548	61
equake	1188231	367
mcf	2241346	176
mgrid	82947	351
gcc	3850329	847
vortex	4275601	1002

表 2　数据 cache 数据块的衰退期(节拍)

Tab. 2　Cycles to decay of data cache line(in cycles)

测试程序	最大衰退期	平均衰退期
applu	538899	230878
art	1250356	17007
bzip2	3062485	2578881
equake	7773570	4658898
mcf	2369133	418408
mgrid	130047	63541
gcc	4525196	161631
vortex	4594759	188699

2　组相联 cache 的替换算法

出于性能的考虑,目前多数微处理器的数据 cache 都是组相联结构。相联度越高,cache 的失效率越低,特别是冲突失效率越低,因此当前的 cache 相联度也呈现出越来越大的趋势。

替换算法对组相联 cache 的性能具有较大的影响,好的替换算法可以将处理器需要的数据尽可能地留在 cache 内,而将已经处于衰退期的数据替换。目前经常使用的替换算法有:随机选取(random)、先进先出(FIFO,First In First Out)、最近最少使用(LFU,Least-Frequently-Used)和最近未使用(LRU,Least-Recently-Used)。其中 LRU 是目前处理器设计中最普遍使用的替换算法。

在高性能微处理器中,出于减少冲突失效的考虑,一般都采用较高相联度 cache($n \geqslant 4$)。虽然 LRU 对于高相联度 cache 仍然可以持续降低失效率,但效果已经不甚明显,因为一次 cache 访问命中 LRU 排序链中靠后的数据块的机会已经微乎其微。我们进行了模拟并统计了 2 路、4 路和 8 路组相联 L1 数据 cache 命中每组 LRU 链后半区的概率,如图 1[15]。可以看出,高相联度 cache 中大部分应用(具有颠簸式访问模式的应用除外)命中后半区比例极小(1% 左右,不超过 3%),即每组后半区中的数据块有大于 97% 的可

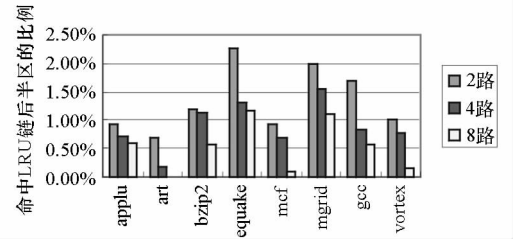

图 1　命中 LRU 排序链后半区的比例

Fig. 1　The proportion of hit in the low half of LRU stack

能已经进入了衰退期。

3　自适应 cache 动态容量调整

组相联 cache 中的替换信息包含了部分暗示,即下一个将被替换的数据块可以作为 cache 漏流功耗优化的候选。因此可以直接利用组相联 cache 的替换算法信息,控制即将被替换的数据块提前进入低漏流状态,对于 LRU 替换算法而言,即控制把长时间未访问的数据块尽早进入低漏流状态。传统的基于计数器溢出的漏流控制策略[3,6]中,数据块根据计数器信息控制数据块关断或睡眠,而其实该数据块在计数器溢出之前的很长时间内已经没有被访问,替换算法的信息早已使我们获悉了这一点。在这段时间内数据块一直处于高漏流但无访问状态(衰退期),这是能量的浪费,具有较大的优化空间。

对于 LRU 替换算法,在本质上,LRU 排序链和计数器在记录的信息上存在部分重复。假定 LRU(n)表示最近使用的数据块,LRU(1)表示最早使用的数据块,则 LRU(n)计数器的值一定最小,最先溢出的计数器一定是 LRU(1)的计数器,而任一时刻 LRU(1)的计数器也肯定是该组所有计数器中最大的一个。如果不考虑硬件实现的代价,负责漏流控制的计数器完全可以行使 LRU 的功能,但这种全 cache 的 LRU 对容量较大的 cache 而言不切实际,将导致访问和失效开销的大幅增加。因此,可以利用 LRU 状态作为辅助决策的信息,不但利用计数器溢出确定数据块进入节能模式,还可以根据数据块在 LRU 排序链中的位置提前使其进入节能模式。

在我们早期研究中已经提出了基于 LRU 信息控制 cache 漏流功耗的 LRU-assist 算法[8,17-18]。基本的 LRU-assist 算法思想是:在 n 路组相联 cache 中,根据 LRU 排序的信息,总是将每组最近未使用的 w 路置为低漏流状态。基本的 LRU-assist 算法本质上等同于 $n-w$ 路组相联 cache,实用意义不大。结合第一节中分析的各种不同应用程序,其平均衰退周期也各有不相同的结论,各组的 w 可以独立设置,同时应该是可以动态变化的,即自适应 LRU-assist 算法。完全自适应 LRU-assist 算法中 w 是随着 cache 的工作状态动态改变的,而且每组之间也各不相同。

我们通过实验统计(如图 1),相联度较高的 cache 中($n \geqslant 4$),最近未使用的数据块和次最近未使用的数据块其访问概率都很低,在前者进入衰退期后,后者也很大概率上进入了衰退期。因

此我们可以在前者关断时,预测性地关断后者来提高 cache 关断的比例。

每组路掩码 w 变化的状态机如图 2 所示,工作流程描述如下:

cache 所有组的 w 赋初始值为 n,即所有 cache 块均关断,类似于 cache 的冷启动过程;

访问某一块发生失效时,激活该数据块,并从下一级存储层次取数,并将该组(设为 i)的 $w_i = w_i - 1$,直到 $w_i = 0$;

当某一组(设为 i)的 $LRU_i(k)$ 由于计数器溢出被关断时,预测 $LRU_i(k+1)$ 的计数器也即将溢出,因此可以提前控制其关断,更新 $w_i = k + 1$。

按照上述流程,w_i 就根据 cache 的运行状态在 0 到 n 之间来回摆动,体现了一种性能和功耗的平衡。

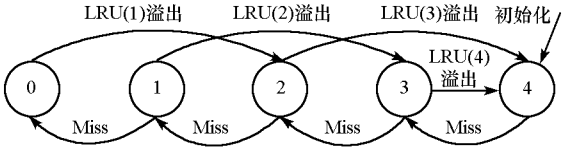

图 2　4 路组相联 cache 中 w_i 变化的状态机

Fig. 2　w_i FSM of 4-way associated cache

为 cache 的每个组都配置一套 w 状态机(称为路掩码),硬件开销十分可观($n \cdot set$ 位),开销包括面积和相应的功耗。为了减少这些开销,可以仅设置一个共享的路掩码 w,即 cache 各组的关断比例相同,但所关断的路号在物理上不同。

同样根据图 1 的实验数据,我们可以得到,某一组的最近未使用数据块进入衰退期后,其他组的最近未使用块也较大概率上进入了衰退期,或者即将进入衰退期。因此,我们可以在某一组的最近未使用数据块关断时,预测性地关断所有组的最近未使用块。

上述算法需要略作改动,状态机如图 3 所示,工作流程如下:

cache 所有组的 w 赋初始值为 n,即所有 cache 块均关断,类似于 cache 的冷启动过程;

任意一组发生 cache 失效时,置整个 cache 的共享路掩码 $w = w - 1$,直到 $w = 0$;

如果某组中有任一数据块的计数器溢出,则预测其他各组的 $LRU(w+1)$ 也将溢出,置 $w = w + 1$,直到 $w = n - 1$。(w 不能为 n,$w = n$ 意味着整个 cache 被旁路,此时会严重影响性能)

利用状态丢失的低漏流存储电路控制 cache 的漏流功耗相当于对 cache 实施动态容量调整策略。通过共享的路掩码 w 在 $0 \sim n-1$ 之间的变化,cache 可以以容量的 $1/n$ 为单位进行动态重构,以最适合当前程序工作集的 cache 容量工作。

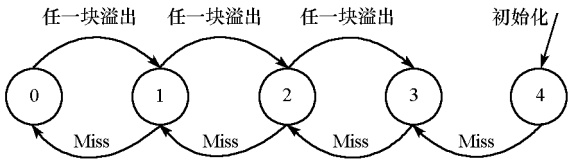

图 3　4 路组相联 cache 共享路掩码 w 变化的状态机

Fig. 3　w FSM of 4-way associated cache with shared way mask

国际上已对 cache 动态容量调整策略进行过较多研究。AMC[9]、cache 衰退技术[3]以及自适应 LRU – assist 算法等是以 cache“块”为粒度进行容量调整,DRI 是以 cache“组”为粒度进行容量调整[2],Selective cache way 则是以“路”为粒度进行容量调整[10]。我们所提出的基于共享路掩码的 LRU-assist 策略可以看作是以“逻辑路”为粒度进行容量调整[15]。逻辑路的概念如图 4 所示。

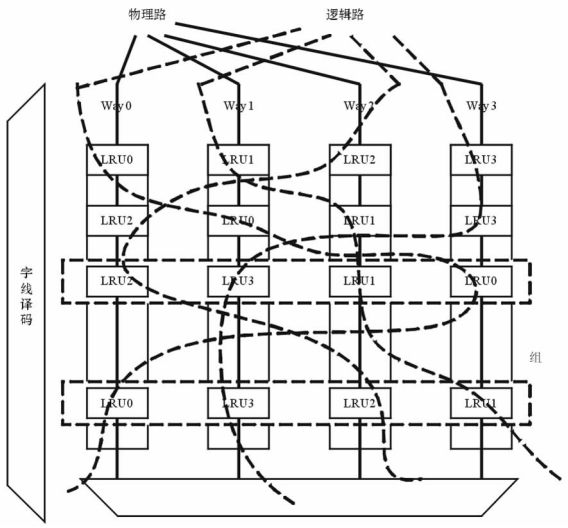

图 4　组、物理路与逻辑路

Fig. 4　Set, physical way and logical way

基于逻辑路的容量调整相对于“物理路”和“组”的优势是逻辑路包含了 cache 块访问的历史信息,避免了对性能产生过于严重的影响。共享路掩码的 LRU-assist 策略不仅仅是逻辑路可重构,当 $w = n - 1$ 时,即位于 LRU 排序链前端的 $n - 1$ 个逻辑路都已被关断,关断比例达到 $(n-1)/n$,此时其余各 cache 块的计数器开始以“块”为单位控制其关断,因此既能保证性能,又可以获得类似“块”重构的最大漏流控制效果。

当前的示例中采用的是基于 LRU 替换算法辅助的逻辑路动态容量调整策略。实际上,任何替换算法(Replacement Algorithm, RA)都可以用来辅助进行基于逻辑路的 cache 动态容量调整,例如 LRFU 算法、RRIP 算法等。后续工作中我们将对基于 LRFU[19] 和 RRIP[20] 替换算法的 cache

逻辑路动态容量调整策略进行研究。

当前 cache 的物理实现中,多以物理路为单位进行 cache 体的组织。采用基于逻辑路的容量调整策略时,可以将逻辑路与物理路的实现统一起来,即物理上的路来组织逻辑上的路。每次在 cache 访问时,根据替换算法要重新对所有路之间的顺序进行排序,因此数据需在多个路之间进行交换,重新放置,以维持逻辑路与物理路上的统一。这种实现方式虽然能够简化容量调整的硬件实现,但是其适应范围有限。比如,由于每次访问需要交换的数据太多,对基于 LRU 算法的逻辑路容量调整并不适合,但适用于基于伪 LRU 算法的逻辑路动态容量调整。

4　实验环境

选用 Simplescalar3.0[11] 模拟器开展性能评价工作,为了更真实地反映实际情况,处理器模型配置成类 Alpha21264 的机器,其中 L1D cache 大小为 64KB,4 路组相联,块大小为 32 字节,本文主要以 L1D cache 为例进行功耗优化的评价,主要原因是 L1D cache 对性能的影响很大,如果 L1D cache 采用基于“逻辑路”的动态容量调整策略,既能降低漏流功耗又能避免较大的性能损失,则其他级 cache 采用该策略也会带来较高的能效。功耗模型选用普林斯顿大学的 Wattch 动态功耗模型[12] 以及弗吉尼亚大学开发的 HotLeakage 漏电流功耗估算模型[13],工艺参数以 65nm 工艺为标准,电压 0.9V,主频 5.6GHz,环境温度 80℃,N 管和 P 管的阈值电压分别为 0.19V 和 0.21V。

在 SPEC CPU2000 中选择了 10 个测试程序运行,分别是 ammp,art,lucas,swim,mcf,wupwise,bzip2, gcc, twolf 和 vortex。测试程序用 Compaq Alpha 编译器在 peak 配置下编译成二进制代码。

5　实验结果

基于逻辑路的自适应动态容量调整策略增加了部分存储资源和相应的控制逻辑,包括路掩码存储部件和状态转换逻辑,将带来额外的动态功耗和漏流功耗。同时,如果被关断 cache 块被再次访问,则需要从下一级存储层次(二级 cache)重新取数,导致额外的动态功耗。这些额外导致的功耗会抵消掉部分动态调整策略所节省的漏流功耗,因此在计算漏流功耗优化能力时需要减去这些额外功耗和开销,得到净节省的漏流功耗。采用上述实验环境,我们针对计数器阈值和 cache 相联度两个指标对基于逻辑路的动态容量调整策略的效果进行了实验分析。

5.1　计数器阈值的设置

首先分析了不同的计数器阈值设置对基于“逻辑路”的动态容量调整策略(图中以 resize 表示)优化效果的影响。实验中配置 L1D cache 容量为 64KB、4 路组相联,块大小 32B。图 5 为动态容量调整在不同计数器阈值设置下的性能。可以看出,计数器阈值设的越大,对性能的影响越小。这是显然的,因为较小的计数器阈值设置带来的误关断数目较多,从而导致 cache 的命中率下降,失效增多,更多的访问需要涉及二级 cache。同 cache 衰退技术[3] 类似,基于逻辑路的动态容量调整在某些测试程序上也可以意外获得性能的改善(如图中的 art 和 mgrid 程序),其原因同样也是由于 cache 在关断数据块时会将被修改过的数据块(“脏块”)主动写回下一级存储层次,这种强制写回的操作是在后台运行的,隐藏了正常访问失效时 cache 写回的处理时间,降低了失效开销,因此有的测试程序 IPC 会升高。同时也说明不断改进容量调整的预测算法,提高预测关断的准确率,

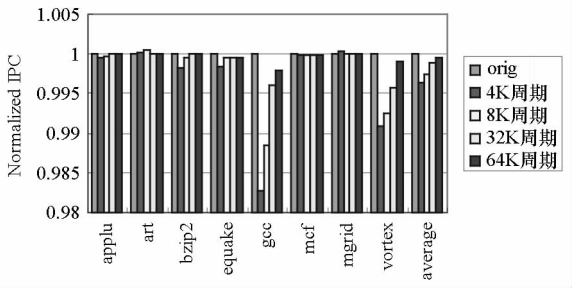

图 5　不同阈值动态容量调整对性能的影响

Fig. 5　Performance impact of dynamic resizing with different threshold

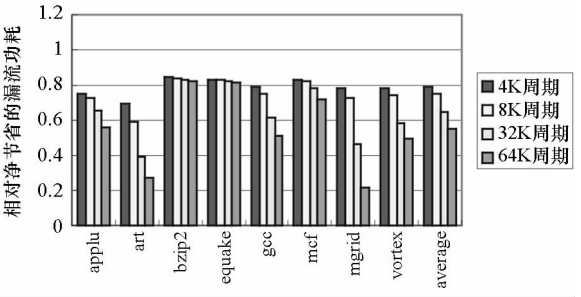

图 6　不同阈值动态容量调整的漏流优化效果

Fig. 6　Leakage optimization of dynamic resizing with different threshold

不但能够降低漏流功耗,而且还可以提高性能。图 6 为不同计数器阈值配置下动态容量调整算法相对净节省的漏流功耗。可见,随着计数器阈值的增大,漏流的优化效果越弱,所取得的漏流功耗的净节省也越小。而采用综合评估指标[15],同时

考虑性能和功耗的折中效果,计数器阈值为 8k 时钟周期的动态容量调整可以获得最佳的能量效率,如图 7 所示。

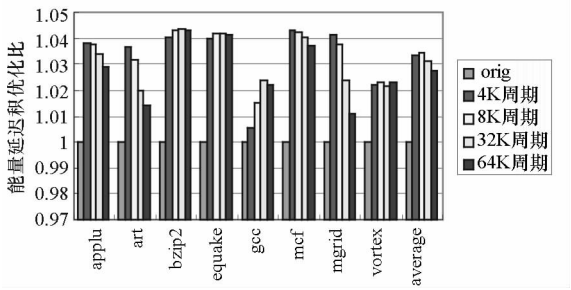

图 7　不同阈值动态容量调整的能量效率

Fig. 7　Energy efficiency of dynamic resizing with different threshold

5.2　cache 相联度

由于基于逻辑路的 cache 动态容量调整算法主要应用于组相联 cache,因此我们考察了算法针对不同相联度的 cache 应用时的效果。在实验中,数据 cache 容量固定为 64KB,块大小固定为 32 字节,相联度分别设置为 2 路、4 路和 8 路。计数器阈值设为 8k。

图 8 和图 9 为不同相联度情况下动态容量调整算法对性能、漏流功耗的影响。可见,基于“逻辑路”的动态容量调整策略在不同相联度的 cache 中均表现出良好的优化效果,而从能量效率的角度看[15],8 路组相联 cache 中漏流优化效果与其他相当,而性能损失较小,如图 10 所示。可见,动态容量调整策略由于其控制粒度较大,比较适合用在相联度较大的 cache 中。

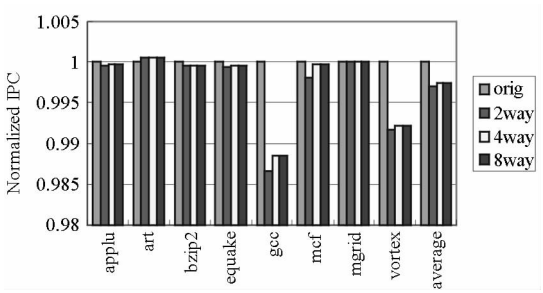

图 8　不同相联度容量调整的性能

Fig. 8　Performance of dynamic resizing with different association

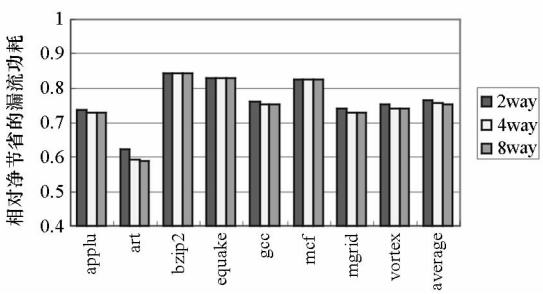

图 9　不同相联度容量调整的漏流优化

Fig. 9　Leakage optimization of dynamic resizing with different association

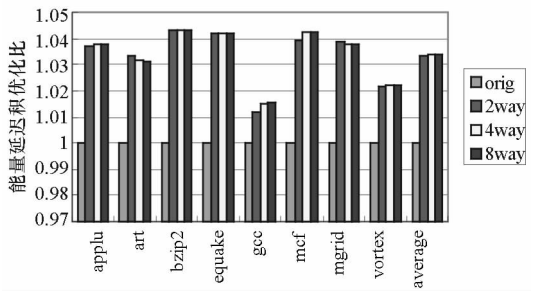

图 10　不同相联度容量调整的能量效率

Fig. 10　Energy efficiency of dynamic resizing with different association

5.3　与其他控制算法的比较

为了验证基于“逻辑路”cache 容量调整算法的优越性,我们将其同传统的两级计数器 cache 衰退技术进行了比较。评测过程中,数据 cache 容量为 64KB、4 路组相联,块大小 32 字节。我们在模拟器上同时实现了基于“逻辑路”的 cache 容量调整算法和两级计数器 cache 衰退技术,其衰退间隔均为最优的 8k 个时钟周期,通过 SPEC 2000 测试程序的模拟,统计并比较了两种漏流控制算法的 IPC 和净节省的漏流功耗。

图 11 为使用漏流优化算法对性能造成的影响,可见两种算法的平均性能都有损失(平均不超过 0.5%,gcc 最大,约为 1%),都是由于误关断造成的,两者对性能的影响相当。图 12 为算法的漏流功耗优化效果,基于逻辑路的动态容量调整技术可以获得比简单的两级计数器策略好的漏流功耗优化能力(cache 衰退为 75.5%,动态容量调整为 76.6%)。综合考虑性能和功耗[15],动态容量调整技术的能量效率显然要优于传统两级计数器的 cache 衰退技术,能更好地在性能与功耗之间取得折中。

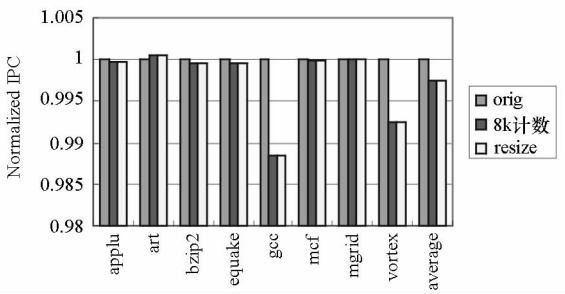

图 11　动态容量调整算法的性能损失

Fig. 11　Performance loss of logic way based DR

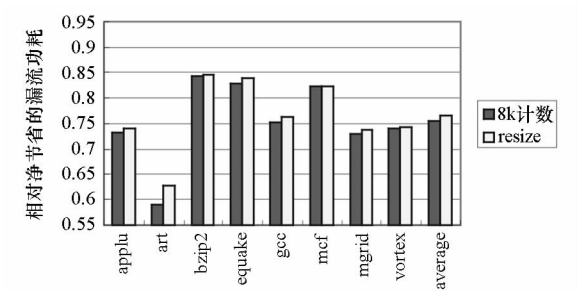

图 12　动态容量调整算法的漏流功耗优化效果
Fig. 12　Leakage optimization of logic way based DR

6　结 论

本文在基于计数器溢出的漏电流功耗控制策略的基础上提出了一种基于“逻辑路”的 cache 动态容量调整策略,利用传统处理器中固有的替换策略信息进行 cache 的漏流功耗优化。讨论了算法原理和有效性以及实现开销,运行 SPEC2000 测试程序的模拟结果表明,该算法具有较好的能量效率。由于利用 cache 中现有替换算法逻辑,相对于原仅使用计数器的 cache 关断策略没有增加新的复杂逻辑和存储部件,对于共享路掩码的 LRU-assist 算法而言,仅增加了一个 $n+1$ 状态数(n 为 cache 组相联路数)的状态机,硬件开销小,且动态容量调整的控制逻辑不在访存的关键路径上,对处理器性能几乎不造成任何影响,平均可将 cache 漏流功耗净节省 76.6%。虽然本文的数据均是基于 L1D cache 和基于状态丢失的低漏流电路技术来讨论的,但从其本质分析上可以知道,基于“逻辑路”的 cache 动态容量调整策略可以适用于任何组相联 cache,也适用于状态保留的低漏流电路技术(如 drowsy),充分体现了体系结构级漏流功耗优化技术与底层低漏流电路实现技术的正交性。

参 考 文 献:

[1] Borkar S, et al. Design Challenges of Technology Scaling[J]. IEEE Micro, 1999, 19(4): 23 - 29.

[2] Powell M D, Yang S H, et al. Gated-vdd: A Circuit Technique to Reduce Leakage in Deep-submicron Cache Memories [C]//Proceedings of the 2000 International Symposium on Low Power Electronics and Design, 2000: 90 - 95.

[3] Kaxiras S, Hu Z, Martonosi M. Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power[C]// Proceedings of the 28th International Symposium on Computer Architecture, 2001:240 - 251.

[4] Pering T, Burd T, Brodersen R. The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms s[C]// Proceedings of the 1998 International Symposium on Low Power Electronics and Design, 1998:76 - 81.

[5] Nii K, Makino H, et al. A low Power SRAM Using Auto Backgate Controlled MT-CMOS[C]//Proceedings of the 1998 International Symposium on Low Power Electronics and Design, 1998:293 - 298.

[6] Flautner K, Kim N S, et al. Drowsy Caches: Simple Techniques for Reducing Leakage Power [C]//Proceedings of the 29th Annual International Symposium on Computer Architecture, 2002:147 - 157.

[7] Li Y, Parikh D, et al. State Preserving vs. Non State Preserving Leakage Control in Caches [C]//Proceedings of the Design Automation and Test in Europe Conference. 2004:22 - 27.

[8] 张承义,张民选, 邢座程,等. LRU-assist:一种高效的 cache 漏流功耗控制算法[J]. 电子学报,2006, 34(09):1626 - 1630.

[9] Zhou H, Toburen M, Rotenberg E, et al. Adaptive Mode Control: A Static-power-efficient Cache Design [C]// Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques, 2001:61 - 70.

[10] Albonesi D H. Selective Cache Ways: On-demand Cache Resource Allocation [C]//Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture, 1999: 248 - 259.

[11] Burger D, Austin T. The Simple Scalar Tool Set. version 2.0. [J] ACM SIGARCH Computer Architecture News, 1997, 25(3):13 - 25.

[12] Brooks D, Tiwari V, et al. Wattch: A Framework for Architectural-level Power Analysis and Optimization [C]// Proceedings of the 27th Annual International Symposium on Computer Architecture, 2000:83 - 94.

[13] Zhang Y, Parikh D, et al. Hotleakage: An Architectural, Temperature-aware Model of Subthreshold and Gate Leakage[R]. CS-2003-05, University of Virginia, 2003.

[14] Kotera I, Abe K, et al. Power-Aware Dynamic Cache Partitioning for CMPs [J]. Transactions on High-Performance Embedded Architectures and Compilers Ⅲ, Lecture Notes in Computer Science, 2011, 6590(2011):135 - 153.

[15] 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:国防科技大学, 2006.

[16] 周宏伟. 微处理器中 Cache 漏流功耗的体系结构级优化技术研究[D]. 长沙:国防科技大学,2007.

[17] 张承义,张民选,邢座程. 组相联 cache 中漏流功耗优化技术研究[J]. 小型微型计算机系统,2007, 28(02):372 - 375.

[18] 张承义,张民选. 片内二级 cache 的静态功耗优化技术研究[J]. 计算机工程与科学,2007, 29(03):77 - 79, 90.

[19] Lee D, Choi J, Kim J, et al. LRFU: A Spectrum of Polices that Subsumes that the Least Recently Used and Least Frequently Used Policies [J]. IEEE Trans. Computers, 2001,50(12):1352 - 1361.

[20] Jaleel A, Theobald K B., Steely S Jr., et al. High Performance Cache Replacement Using Re-Reference Interval Prediction (RRIP)[C]//ISCA'10, France, 2010:60 - 71.