LRU-Assist 一种高效的 Cache漏流功耗控制算法

张承义, 张民选, 邢座程, 王永文
(国防科技大学计算机学院 610室, 湖南长沙 410073)

摘 要: 随着集成电路制造工艺进入超深亚微米阶段, 漏电流功耗在微处理器总功耗中所占的比例越来越大, 在开发新的低漏流工艺和电路技术之外, 如何在体系结构级控制和优化漏流功耗成为业界研究的热点. Cache在微处理器中面积最大, 是进行漏流控制的首要部件. LRU是组相联 Cache最常用的替换算法, 而研究发现, 访存操作命中 LRU后半区的概率很低. LRU-Assist算法以 Drowsy Cache, Cache Decay等控制策略为基础, 在保证处理器性能不受影响的前提下, 利用既有的 LRU信息把 Cache的关闭率平均提高了 15%, 大大降低了漏电流功耗.

关键词: 微处理器; cache功耗; 漏电流; LRU-assist
中图分类号: TP302.7 文献标识码: A 文章编号: 0372-2112 (2006) 09-1626-05

LRU-Assist An Efficient Algorithm for Cache Leakage Power Controlling

ZHANG Cheng-yi, ZHANG Min-xuan, XING Zuo-cheng, WANG Yong-wen
(Lab 610, School of Computer, National University of Defense Technology, Changsha, Hunan 410073, China)

Abstract The leakage power issue is challenging high-performance microprocessor design, especially as feature size shrinks. Not only are low leakage technologies and circuits well researched, but also architectural control methods are studied hotly. Caches represent a sizable fraction of the total power consumption, so they need to be managed firstly. LRU is the most popular replacement algorithm used in set associative caches, but researches show that the latter blocks in LRU list are rarely accessed again. LRU-assist algorithm proposed in this paper exploits existing LRU information to expand the low leak portion in cache in addition to the time-based drowsy and decay mechanism. Simulation results show that the cache off ratio can be increased by 15% and leakage power is greatly saved with negligible performance overhead.

Key words microprocessor; cache power; leakage; LRU-assist

1 引言

长期以来, 性能一直是高性能微处理器所追求的首要目标. 可是随着性能的提高, 微处理器的功耗问题也越来越严重, 对微处理器设计提出了严峻的挑战. 微处理器的功耗可以分为动态功耗和静态功耗两部分, 动态功耗主要是指翻转功耗, 由电压、频率、有效电容决定[1], 而静态功耗是指由漏电流导致的功耗, 在 CMOS电路中, 无论电路处于什么状态, 漏电流总是存在, 因此漏电流受电压、工艺参数以及温度的控制[2]. 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深亚微米、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 (SCE, Short Channel Effect)、DIBL (Drain-Induce Barrier Lowering)、GIDL (Gate-Induced Drain Leakage)以及隧道效应等多种原因导致漏电流成指数增长, 漏电流所导致的静态功耗在总功耗中所占的比例越来越大, 甚至超过了动态功耗. Borkar估算每代微处理器产品的亚阈值漏电流会增加 7.5倍[3], 半导体协会则预测栅极漏电流会以 500倍的速度增长[4], 在接下来几代工艺中, 静态功耗将超过 50% (≤65nm), 如果不加以控制, 将严重阻碍微处理器性能的进一步提升. 因此, 越来越多的公司和大学开始着手漏电流功耗控制与优化技术的研究, 以期在工艺级、电路级、体系结构级等多个层次对静态功耗进行控制.

Cache是微处理器中功耗最大的部件之一. 在 Alpha21264高性能微处理器中, cache的功耗占全部功耗

收稿日期: 2005-07-28; 修回日期: 2006-01-20
基金项目: 国家自然科学基金 (No. 60376018, 60273069, 90207011)

的 15% [5], 而 Itanium2处理器片上集成了三级 cache, 片上 cache总容量达到了 3.3MB[6]. 为了持续的提升性能、缩减处理器带宽与存储器带宽之间的差距, 片上 cache的容量越来越大, 相联度也越来越高, 因此 cache的漏电流控制成为低功耗微处理器设计中首先要解决的问题. 门控电压 (Gated-Vdd)[7]和动态电压调整 (DVS)[8]是目前比较有效的两种控制漏电流的电路技术, cache decay[9]和 drowsy cache[10]就是分别利用这两种电路技术的微体系结构设计方案.

Cache decay和 drowsy cache的原理都基于以下事实: cache中的绝大部分 cache块在其最后一次访问到被替换出该级 cache之前有较长的一段时间处于无用状态, 这是一种功耗的浪费. Cache decay利用睡眠晶体管技术将这些存储单元的电源线或地线切断, 使其漏电流功耗几乎为零, 保存的数据也随之丢失; 再次访问该单元时, 睡眠晶体管导通, 重新从下一级 cache中读入数据, 误关闭的开销较大. Drowsy cache则将可能不再使用或短期内不再使用的存储单元置于一个低电压、低漏流状态, 虽然漏电流仍存在, 但数据得以保持, 再次访问时耗时 1到 2拍重新调整到高电压即可, 开销较小. 因此权衡性能损失和功耗的降低, 在下一级 cache访问延迟较大时 (如访问片外 cache)宜选择 drowsy cache策略, 而下一级 cache访问延迟不大时 (如访问片上 cache)可以选择使用 cache decay策略[11].

确定无用块一般都使用计数器溢出策略, 为每个 cache块设置一个计数器, 在每次访问该块时, 计数器复位, 然后每拍增 1, 如果计数溢出时仍未被再次访问, 则该块可以 decay或进入 drowsy状态, 溢出信号就作为低漏流控制信号. 为了减少计数器翻转带来的额外动态功耗, 可以使用两级计数器, 每个 cache块设置一个较小的局部计数器, 整个 cache共享一个全局计数器[9], 全局计数器每拍翻转, 只有当全局计数器溢出时局部计数器才翻转. 计数器的阈值需要进行仔细的设计, 阈值太大, 可关闭 (或睡眠)的 cache块太少, 漏流功耗降低不明显, 甚至不能抵消计数器动态功耗; 阈值过小, 误关闭率变大, 处理器性能损失太严重, 最终 EDP反而变大. 因此, 如何保证在性能损失不大的情况下最大限度的关闭或睡眠更多的 cache块成为 cache漏电流功耗优化新的研究目标. Y. Meng等人综合利用 cache decay和 drowsy cache技术, 为 cache的漏流功耗优化给出了一个上限, 并建议采用预取策略接近这个值[12]. 有人提出了自适应策略改变计数器的宽度, 根据应用程序执行的情况动态调整关闭率[13,14]. W. Zhang等人研究了软件辅助的控制策略, 编译器插入功耗相关指令界定指令 cache某块的最后一次访问, 控制其尽可能早地进入低漏流状态[15]. 本文通过对高相联度 cache中 LRU替换算法性能的分析, 提出了一种 LRU-assist算法, 几乎不增加任何硬件开销, 在保证性能几乎不受损失的情况下, 可以有效提高 cache关闭率, 大大降低了漏电流功耗.

2　LRU-assist算法

2.1　LRU替换算法

LRU (Least Recently Used)算法是组相联 cache中使用最普遍的替换算法, 可以有效地降低 cache失效率而且硬件实现简单, 每次替换都选择 LRU链最后一个块, 即最后一次使用的时刻离当前时刻最远的那个数据块. LRU 算法可以简单表示为一个双射函数: $LRU: \{0, 1, \cdots, n-1\} \rightarrow W$ i 其中 n为 cache相联度, W i为 cache第 i组 n路数据块的集合, LRU (0)为 MRU块, LRU ($n-1$)为 LRU块. 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访问有多路选择 (高相联度)时, 命中 LRU排序链中靠后的数据块的机会已经微乎其微.

我们利用 simplescalar模拟器[16]对 SPEC CPU2000的测试程序进行模拟, 并统计了 4路组相联 $L1$数据 cache命中每组 LRU链后半区 LRU ($\geqslant n/2$) (即集合 {LRU ($n-1$), LRU ($n-2$), …, LRU ($n/2$) })的访问次数, 图 1为模拟结果. 从中可以看出, 高相联度 cache中命中 LRU ($\geqslant n/2$)的比例极小, 平均 1% 左右, 最大不超过 3%. 可以推测, 在传统的基于计数器溢出的漏流控制策略中, 从数据块进入 LRU排序链后半区到最终由于局部计数器溢出而被关闭或睡眠, 这段时间内数据块一直处于高漏流但无访问的状态, 这是能量的浪费, 具有较大的优化空间.

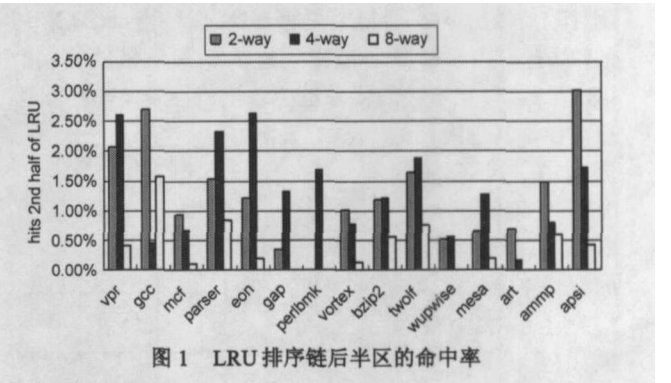


图 1　LRU排序链后半区的命中率

2.2　基本 LRU-assist算法

为了不增加额外的硬件和功耗开销, 可以直接利用 LRU阵列的信息, 控制 cache数据块尽早进入低漏流状态. 本质上, LRU阵列和计数器在记录的信息上存在一定的重复, 某组中 LRU (0)块的计数器值一定是该组中最小的, 最先溢出的计数器则一定是 LRU ($n-1$)数据块的计数器, 而任一时刻 LRU ($n-1$)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, 我们利用 LRU状态作为辅助决策的信息, 不但利用计数器溢出控制数据块进入节能模

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,这称为 LRU-assist算法.

基本的 w/nLRU-assist算法就是除了使用计数器溢出控制数据块进入低漏流模式之外,固定地将每组 LRU 排序链后 w/n 区域的 w 路立刻置为低漏流模式,w 则为 0到 n 中的某一个值. $w=0$即为原来的计数器溢出策略,$w=n$则相当于访问一个速度较慢的 cache(每次访问都需要一个唤醒延时,甚至要到下一级 cache取数),因此研究中我们取 $w=1, 2, \cdots, n-1$.

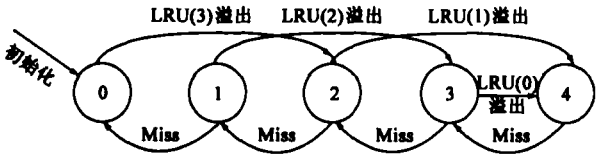


图 2 4路组相联 cache中 w_i 变化的状态机

2.3 自适应 LRU-assist算法

不难发现,基本的 LRU-assist算法在应用到诸如 gated-vdd这种会破坏原始数据的低漏流存储电路时,使用 w/n LRU-assist算法的 cache就相当于 w-n 路组相联 cache,没有什么实际意义. 因此,为了在 decay cache中使用 LRU-assist算法,我们对其进行了扩展,称为自适应 LRU-assist算法. 基本 LRU-assist算法中 w 是不变的,在设计时已固定,而自适应 LRU-assist算法中 w 是随着 cache的应用状态动态改变的,而且每组之间也各不相同. wi 最先赋初始值为 0,即不采用 LRU-assist辅助策略,仅使用空闲计数器溢出策略;伴随着处理器的运行,当 cache第 i 组的 LRU $i(k)$由于计数器溢出而进入低漏流模式时,预测 LRU $i(k-1)$在不久的将来也会溢出,更新 $wi=n-k+1$,提前控制其进入低漏流模式,接下来的 cache管理就受 LRU-assist和计数器溢出的双重控制;若第 i组发生一次失效替换,则更新 $wi=wi-1$,直到 $wi=0$. 如此反复,wi 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. 如图 2所示为 4路组相联 cache中 wi变化的状态机.

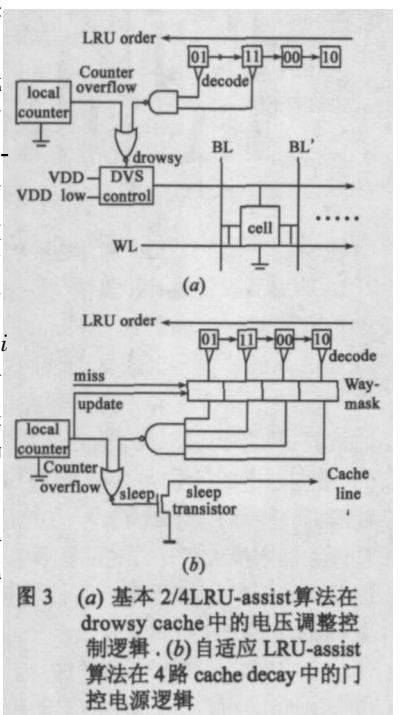


图 3 (a) 基本 2/4LRU-assist算法在 drowsy cache中的电压调整控制逻辑. (b)自适应 LRU-assist算法在 4路 cache decay中的门控电源逻辑

3 实现

LRU-assist算法实现非常简单,硬件开销很少. 基本的 LRU-assist算法只需要增加少量的控制逻辑,如图 3(a)为 LRU-assist应用在 4路组相联 drowsy cache中每个 cache块的电压控制逻辑. 自适应的 LRU-assist算法由于关闭比例可变,因此需要增加保存该比例的单元每组增加 n位掩码 way-mask及其外围控制逻辑,用于自适应算法的实现. 如图 3(b)为自适应 LRU-assist算法应用在 4路组相联 decay cache中的门控电源地逻辑. 在实际实现中,为了减少自适应 LRU-assist算法所需的硬件逻辑,减少额外的动态功耗,可以将分布的 way-mask更换为共享的 way-mask,即整个 cache共享一个 way-mask,效果依然明显.

4 性能评价

前面两节介绍了 LRU-assist算法的执行流程和硬件实现方法,了解了基本 LRU-assist算法和自适应 LRU-assist算法各自的优势和局限性. LRU-assist算法本质上是一种基于统计规律的预测,或者叫赌博,但这种预测不是盲目的,而是性能、静态功耗、动态功耗三者的权衡. 本节我们通过对真实测试程序的模拟,研究 LRU-assist算法在保证性能、降低功耗等方面的效果.

4.1 模拟器环境和评价指标

我们选用 simplescalar3.0模拟器开展性能评价工作,为了更真实的反映实际情况,处理器模型配置成类 Alpha21264的机器(表 1),主要以 L1 DCache为例进行功耗优化的评价. 功耗模型选用普林斯顿大学的 Wattch模型[17,18]以及弗吉尼亚大学开发的 HotLeakage漏电流功耗估算模型[19],工艺参数以 70nm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.

表 1 Simple Scalar模拟器配置

Processor core	
Instruction Window	80-RUU, 40-LSQ
Issue width	4
Function Unit	4-IntALU, 1-IntMULT 2-FpALU, 1-FpMULT 2-MemPort
Memory Hierarchy	
L1 DCache	64KB, 4-way, 32Bblock, WB
L1 ICache	64KB, 2-way, 32Bblock, WB
Unified L2 Cache	1MB, 4-way, 32Bblock, WB, 6-cycle latency
Memory	100 cycles, 16 bus width

我们在 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,输入数据集采用 ref.为了缩短模拟周期并保证获得最有价值的程序行为,选用 Simpoint[20]进行加速.

由于 cache的漏电流功耗主要取决于 cache中活跃部

分的比例, cache的关断率 off-ratio即处于低漏流状态的比例可以用来代表算法优化漏流功耗的能力. 为了考察 LRU-assist相对 decay或 drowsy的优越性, 我们用 off-ratio-increase作为统计指标; 而算法对性能的影响则用 IPC的降低 (IPC-decrease)来衡量. 本文假设漏流功耗的降低同 cache关断率成正比, 因为性能的损失很小 (在下文将看到), 因此导致的额外动态功耗和静态功耗也很小, 可以忽略不计.

4.2 LRU-Assist Drowsy Cache

基本的 LRU-assist一般用在状态保留的漏流控制策略中, 如 drowsy cache. 我们在 Simple Scalar 3.0上针对 64KB的 L1D cache实现了基本 LRU-assist drowsy cache, 并对不同相联度情况下 LRU-assist所带来的关闭率的提升以及对性能的影响进行了模拟, w 固定为 $n/2$. 如图 4所示. 从图中可以看出 LRU-assist策略对处理器的性能影响非常低 (IPC的降低不超过 0.004), 但漏流功耗平均可以进一步节约 15%左右. 图中 vortex和 wupwise的效果不明显是因为它们使用原始基于计数器的 drowsy策略的关闭率已经达到了 80% ~ 90%.

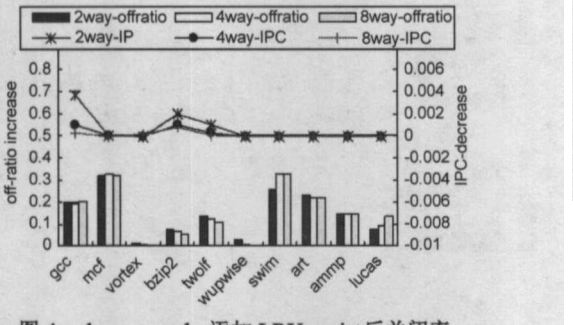


图 4　drowsy cache添加 LRU-assist后关闭率的提升和对性能的影响

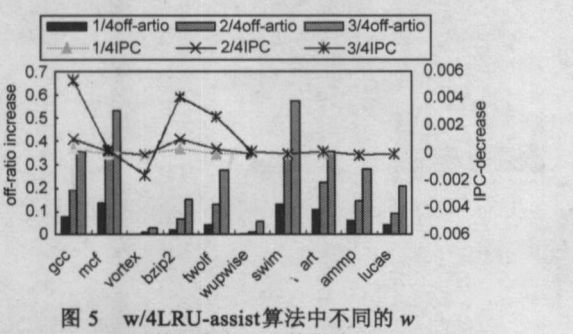


图 5　w/4LRU-assist算法中不同的 w 对关闭率和 IPC的影响

此外我们还研究了 w 的变化对 LRU-assist算法节能效果的影响. 图 5是 4路组相联 drowsy cache中 0/4(基于计数器的 drowsy策略)、1/4、2/4、3/4 LRU-assist对关闭率和 IPC的影响. 可以看出 $(n-1)/n$ LRU-assist算法能够最大限度的提升 cache关闭率, 而且 IPC没有太大变化, 最大不超过 0.006

4.3 自适应 LRU-assist cache decay

对于不能保留原始数据的 cache漏流控制策略, 如 cache decay, 自适应 LRU-assist算法可以达到降低 cache漏流功耗的效果. 对于不同相联度的 decay cache, 我们扩展了自适应 LRU-assist算法, 并对其在功耗和性能方面的影响进行了模拟, 如图 6所示. 同基本的 LRU-assist算法相同, 自适应算法对 cache关闭率的提升平均也达到了 15%. 由于 cache块在进入漏流模式后数据丢失, 再次的访问需要到下一级 cache取数, 因此 IPC下降相对较多 (有的达到了 0.03). 但有趣的是, 在某些情况下 (特别是 8路自适应 LRU-assist cache decay) IPC非但没有下降, 反而得到了改善, 这是因为数据块在进入低漏流模式之前要将脏块写回, 这种主动的写回是在后台进行的, 不占用 cache访问的带宽, 而且可以降低 cache失效时由于写回带来的失效开销.

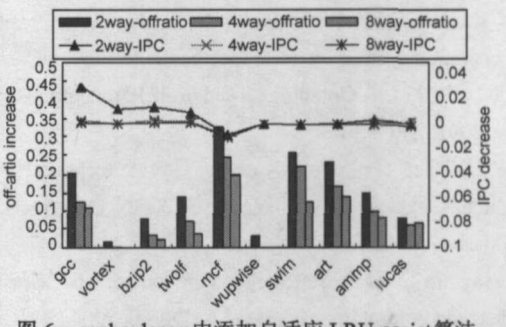


图 6　cache decay中添加自适应 LRU-assist算法后关闭率的提升和对性能的影响

5 结论

在使用 LRU替换算法的组相联 cache中, 由于 LRU块的访问概率极低, 消耗了大量漏电流功耗, 因此本文在 drowsy cache, cache decay等原始基于计数器的 cache漏电流功耗控制策略的基础上提出了一种改进, 即利用既有的 LRU信息提高 cache的低漏流比例, 称为 LRU-assist算法, 并针对 drowsy cache和 cache decay各自的特性分别定制了基本 LRU-assist算法和自适应 LRU-assist算法, 给出了电路实现. 模拟结果表明, 该算法硬件开销小, 对处理器性能几乎不造成任何影响, 最大可以提高 35%的 cache关闭率 (平均 15%), 可以大幅降低 cache的漏电流功耗.

参考文献:

[1] 周润德, 等, 译. 数字集成电路 ——电路、系统与设计 (第二版) [M]. 北京: 电子工业出版社, 2004. 10.

[2] A Keshavarzi, K Roy, C Hawkins. Intrinsic leakage in low power deep submicron CMOS ICs[A]. IEEE International Test Conference[C]. Washington DC, USA: IEEE Computer Society, 1997. 146-155.

[3] S Borkar. Design challenges of technology scaling[J]. IEEE Micro, 1999, 19(4): 23-29.

[4] Semiconductor Industry Association. International technology roadmap for semiconductors 2004 update[DB/OL]. http://www.itrs.net/Common/2004Update/2004Update.htm, 2005-01-10.

[5] Gowan M K, et al. Power considerations in the design of the alpha 21264 microprocessor[A]. DAC1998[C]. Los Alamitos, CA, USA: ACM PRESS, 1998. 26-31.

[6] C McNairy, D Soltis. Itanium 2 processor microarchitecture[J]. IEEE Micro, 2003, 23(02): 44-55.

[7] M D Powell, et al. Gated-vdd: a circuit technique to reduce leakage in deep-submicron cache memories[A]. ISLPED2000[C]. Rapallo, Italy: ACM PRESS, 2000. 90-95.

[8] T Pering, T Burd, R Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms[A]. ISLPED1998[C]. Monterey, CA: ACM PRESS, 1998. 76-81.

[9] S Kaxiras, Z Hu, M Martonosi. Cache decay: exploiting generational behavior to reduce cache leakage power[A]. ISCA2001[C]. Goteborg, Sweden: IEEE Computer Society, 2001. 240-251.

[10] K Flautner, et al. Drowsy caches: simple techniques for reducing leakage power[A]. ISCA2002[C]. Anchorage, Alaska, USA: IEEE Computer Society, 2002. 147-157.

[11] Yingmin Li, et al. State-preserving vs. non-state-preserving leakage control in caches[A]. DATE 2004[C]. Paris, France: IEEE Computer Society, 2004. 22-29.

[12] Yan Meng, et al. On the limits of leakage power reduction in caches[A]. HPCA 2005[C]. San Francisco, CA, USA: IEEE Computer Society, 2005. 154-165.

[13] H Zhou, et al. Adaptive mode control: a static-power-efficient cache design[A]. PACT2001[C]. Barcelona, Spain: IEEE Computer Society, 2001. 61-70.

[14] S Velusamy, et al. Adaptive cache decay using formal feedback control[A]. Proceedings of the Workshop on Memory Performance Issues, held in conjunction with ISCA29[C]. Anchorage, Alaska, USA: IEEE Computer Society, 2002. 1-8.

[15] W Zhang, et al. Compiler-directed instruction cache leakage optimization[A]. IEEE Micro35[C]. Istanbul, Turkey: IEEE Computer Society, 2002. 208-218.

[16] D Burger, T Austin. The simplescalar tool set, version 2.0[J]. Computer Architecture News, 1997, 25(3): 13-25.

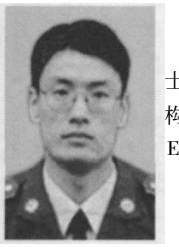
[17] D Brooks, et al. Wattch: a framework for architectural-level power analysis and optimization[A]. The 27th ISCA[C]. Vancouver, BC: IEEE Computer Society, 2000. 83-94.

[18] P Shivakumar, N P Jouppi. Cacti 3.0: an integrated cache timing, power, and area model[R]. Palo Alto, CA: HP Labs, Dec, 2001. Technical Report WRL-2001-2.

[19] Y Zhang, et al. Hotleakage: an architectural, temperature-aware model of subthreshold and gate leakage[R]. Virginia, USA: Department of Computer Sciences, University of Virginia, Mar, 2003. Technical Report CS-2003-05.

[20] T Sherwood, et al. Automatically characterizing large scale program behavior[A]. ASPLOS-X[C]. San Jose, USA: ACM Pess, 2002. 45-57.

作者简介:



张承义 男, 1978年 11月生于河北故城, 博士研究生, 主要研究方向为高性能微处理器体系结构及其低功耗设计技术.
E-mail: chengyizhang@nudt.edu.cn

张民选 男, 1954年 3月生于湖南邵阳, 现为国防科技大学教授, 博士生导师, 主要研究方向为计算机系统结构、工程实现、大规模集成电路设计等. E-mail: mxzhang@nudt.edu.cn