采用预测校正的动态可重构指令 Cache

曹向荣,张晓林
(北京航空航天大学电子信息工程学院,北京 100191)

摘 要: 本文提出一种兼顾性能与功耗的 cache 最优参数检索算法.通过运行时反馈的 cache 评价指数,预测校正 cache 参数检索空间与检索顺序,在保证检索效率的同时,提高结果的准确率.该算法可以减少穷举法近 80% 的迭代次数;同时以损失部分效率为代价,提高降维检索法 13.4% 的全参数准确率以及 40% 的容量参数准确率.

关键词: 高速缓冲存储器;动态重构;预测校正;低功耗
中图分类号: TP333 **文献标识码:** A **文章编号:** 0372-2112 (2014)05-0982-05
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.05.023

Dynamic Configurable Instruction Cache with Prediction Correction

CAO Xiang-rong, ZHANG Xiao-lin
(*School of Electronic and Information Engineering, Beihang University, Beijing* 100191, *China*)

Abstract: A cache optimal parameters search method which balances the performance and power dissipation is proposed. According to the cache evaluation index, prediction correction for parameter search space and search order is made to improve the result accuracy. It can reduce about 80% iterations of exhaustive method. At a modest loss of performance, it improves 13.4% of all parameters accuracy and 40% of capacity parameter accuracy of dimensionality reduction search method.

Key words: cache; dynamic configurable; prediction correction; low power dissipation

1 引言

Cache 主要参数:容量、块大小、关联度对性能与功耗有着重大影响[1].研究表明[2~5],不同的应用程序对应着不同的 cache 最优参数配置,如容量过小会降低性能,过大则会带来功耗损失.如何配置 cache 参数,使其能够适应不同的环境是 cache 设计的难题之一.为了应对这一挑战,人们提出了片上动态可重构 cache 技术,其主要思想是通过监测程序运行时行为,在线调整 cache 参数,使其适应当前环境,其架构如图 1 所示.

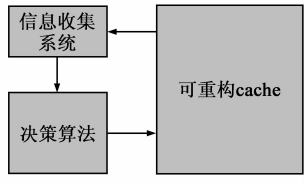

图1 动态可重构cache架构

2 问题分析与实验方法

2.1 问题分析

对 cache 最优化参数问题做如下数学定义:

C:容量集合,$C=\{c_i \mid i\in(1,m), m\in N\}$.

B:块大小集合,$B=\{b_j \mid j\in(1,n), n\in N\}$.

A:关联度集合,$A=\{a_k \mid k\in(1,p), p\in N\}$.

$L_{(c_i,b_j,a_k)}$:*cache* 参数集合,$\{L_{(c_i,b_j,a_k)} \mid b_j * a_k \le c_i, c_i \in C, b_j\in B, a_k\in A\}$.

$f(l)$:cache 评价指数,$l\in L$.

Cache 最优化参数问题归结为在 L 上求解 $f(l)$ 的最优值.便于分析与描述,本文对文字做如下规定:

(1)集合元素:小写字母表示,如 $l_{(c_1,b_1,a_1)}$ 表示 L 中元素 (c_1,b_1,a_1).

(2)集合:大写字母表示,如 $L_{(C,a_1,b_1)}$ 表示 $\{l_{(c_1,b_1,a_1)}, l_{(c_2,b_1,a_1)}\cdots l_{(c_m,b_1,a_1)}\}$.

(3)cache 参数格式:容量:块大小:关联度,如 1k:32:4.

(4)迭代:表示对 $f(l)$ 的一次采样.

文献[2,4,5]分别提出针对容量、块大小、关联度中某个参数的动态重构方法.文献[6]提出了一种兼顾容量与关联度的动态方法.文献[2,4~6]的决策算法都采

收稿日期:2013-03-28;修回日期:2013-08-21;责任编辑:马兰英
基金项目:国防科工局航天民用专项;北京市重点学科基金(No.XK100070525)

用穷举法,将失效率、IPC 或功耗作为 cache 评价指数.以上方法存在如下不足:

(1)穷举法需 $m * n * p$ 次迭代才能得到最优解.随着 m, n, p 的增加,穷举法显然无法满足要求.

(2)采用单一指标作为 cache 评价指数,无法体现性能与功耗并重的设计要求.

文献[7]提出了一种动态配置所有参数的方法.为了减少迭代次数,其采用降维检索法,在检索某个参数时,保持其他参数不变,将原来基于 L 的 3 维搜索降为 1 维.该算法只需 $m + n + p$ 次迭代,但却存在精度问题,实验表明,其结果往往非最优解,详见表 4.

针对以上不足,本文提出一种兼顾性能与功耗,具有较高精度与效率的 cache 最优参数检索算法—预测校正检索法.

2.2 实验方法

本文采用 SimpleScalar 3.0[8,9]作为实验仿真平台,其性能功耗相关参数如表 1 所示,其余参数同 SimpleScalar 默认设置.

表 1　平台配置

主　频	200MHz	电　压	3.3V
工　艺	0.18μm	温　度	310K

本实验将 level-1 I-cache 作为实验对象,从 CPU SPEC 2006 与 MediaBench 中选取 15 个测试样本.通过穷举法找出每个样本的 cache 最优参数,以此为标准检验其他算法的有效性.Cache 参数配置范围如表 2 所示,由于 1k:256:8 不满足 L 约束,所以共有 179 种参数组合.

表 2　参数配置范围

参数	单位	配置范围
C	kB	1,2,4,8,16,32,64,128,256
B	byte	16,32,64,128,256
A		1,2,4,8

3 预测校正检索法

3.1 Cache 评价指数

失效率与功耗是 cache 的重要指标,低失效率意味着高性能,但片面强调性能会带来无谓的功耗损失,合理的 cache 评价指数应综合考虑这两项指标.

失效率与功耗是不同体系的评价指标,必须通过归一化才能建立两者的联系.本文提出 cache 评价指数 f,如式(1~3)所示.M_l, P_l 表示 l 下的 cache 失效率与功耗;$\overline{M_l}, \overline{P_l}$表示 M_l, P_l 归一化处理;L'为采样集合;

$$f(l) = (\overline{M_l} + \overline{P_l})/2, l \in L' \tag{1}$$

$$\overline{M_l} = M_l / \mathrm{Max}(M_{L'}) \tag{2}$$

$$\overline{P_l} = P_l / \mathrm{Max}(P_{L'}) \tag{3}$$

遍历 L,得到每个参数配置对应的 $f(l)$,无论是失效率还是功耗,都要求越低越好,所以 $f(l)$的最小值即对应最优化 cache 参数,见图 2.

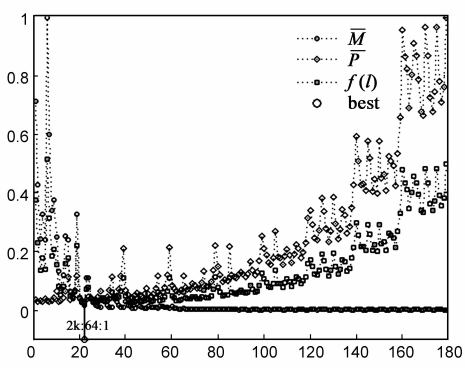

图2　bzip2 $M_l, P_l, f(l)$统计

3.2 功耗评估

计算 $f(l)$需要运行时 cache 失效率与功耗.失效率可以通过 SimpleScalar 计算,功耗则只能建模估算.片上 cache 模型 CACTI[10]根据 cache 参数给出优化的 cache 结构与物理特性,如访问时间,动态功耗等.基于 CACTI 与 Simplescalar 给出动态可重构 cache 功耗估算模型[11],如式(4~7).

$$E_{dynamic} = N_{read} \times E_{read\_per\_port} + N_{write} \times E_{write\_per\_port} \tag{4}$$

$$E_{static} = P_{leakage} \times Cycle \tag{5}$$

$$E_{\mathrm{module}} = E_{dynamic} + E_{static} \tag{6}$$

$$E_{total} = \sum E_{\mathrm{module}} \tag{7}$$

$E_{read_per_port}, E_{write_per_port}$表示模块一次读,写操作功耗;$P_{leakage}$表示模块每周期静态功耗;$N_{read}$与 N_{write}表示模块的读写次数,反映了不同行为导致的功耗差别;$Cycle$ 表示程序执行周期数.模块的功耗由 $E_{dynamic}$与 E_{static}组成,而系统的总功耗为所有模块功耗之和.

3.3 预测校正检索法

传统降维检索法在 $L_{(C,b_1,a_1)}$上检索 c_{best}时,指定块大小 b_1,关联度 a_1,需 m 次迭代.但同样可以从空间 $L_{(C,b_2,a_1)}, L_{(C,b_1,a_2)}\cdots$上检索 c_{best},实际上共存在 $n * p$ 个可选空间.选取哪个空间作为检索空间,是一个值得探讨的问题.

假设 cache 最优参数为(c_5, b_4, a_3),如在 $L_{(C,b_4,a_3)}$, $L_{(c_5,B,a_3)}$, $L_{(c_5,b_4,A)}$ 上检索,最终结果必是最优解.$L_{(C,b_4,a_3)}, L_{(c_5,B,a_3)}, L_{(c_5,b_4,A)}$称为最优检索空间,$c_5, b_4, a_3$ 称为最优检索基底.文献[7]对所有的程序都采用固定的检索空间 $L_{(C,b_1,a_1)}, L_{(c_{best},B,a_1)}, L_{(c_{best},b_{best},A)}$,只要检索空间不是最优的,就可能无法得到最优解,这是降维检

索法产生误差的原因之一.

另外降维检索法也采用固定的参数检索顺序,但由最优化理论可知,利用降维法在多参数系统中求解最优解,应优先检索权重大的参数,称为最优检索顺序.文献[7]没有对 cache 参数权重进行讨论,这是其产生误差的另一个原因.

对动态可重构 cache 而言,最优检索空间与检索顺序是未知的,它们与程序的自身特性相关.预测校正检索法,利用程序运行时信息,对检索空间与检索顺序做出校正,然后按校正后的检索顺序与检索空间求解 cache 最优参数.

首先获取样本在 $L_{(C,b_1,a_1)}$, $L_{(c_1,B,a_1)}$, $L_{(c_1,b_1,A)}$ 上的 $f(l)$ 值,然后根据该信息做如下校正:

(1)校正检索空间

最优参数是指使 $f(l)$ 值最小的 l,选取 $f(l)$ 在 $L_{(C,b_1,a_1)}$, $L_{(c_1,B,a_1)}$, $L_{(c_1,b_1,A)}$ 上的最小值对应的 c_{pbest}, b_{pbest}, a_{pbest} 作为预测最优基底.若检索顺序为 $c-b-a$,则校正检索空间为 $L_{(C,b_{ybest},a_{pbest})}$, $L_{(c_{best},B,a_{pbest})}$, $L_{(c_{best},b_{best},A)}$.其他检索顺序,与之同理.

(2)参数权重分析

参数权重反映了参数对系统的影响.如果 cache 参数变化引起的 $f(l)$ 波动越大则代表其权重越重.根据数理统计知识,标准方差反映了统计量波动大小,我们将 $f(l)$ 在 $L_{(C,b_1,a_1)}$, $L_{(c_1,B,a_1)}$, $L_{(c_1,b_1,A)}$ 上的标准方差 δ_c, δ_b, δ_a 作为参数权重度量,如式(8~10), E 表示数学期望.

$$\delta_c=\sqrt{\frac{\sum_i^m(f(l_{(c_i,b_1,a_1)}-E(f(L_{(C,b_1,a_1)})))^2}{m}} \tag{8}$$

$$\delta_b=\sqrt{\frac{\sum_j^n(f(l_{(c_1,b_j,a_1)}-E(f(L_{(a_1,B,a_1)})))^2}{n}} \tag{9}$$

$$\delta_a=\sqrt{\frac{\sum_k^p(f(l_{(c_1,b_1,a_k)}-E(f(L_{(a_1,b_1,A)})))^2}{p}} \tag{10}$$

预测校正检索算法如图 3 所示,因为增加了采样过程,算法需要 $2*(m+n+p)$ 次迭代.

3.4 算法硬件实现

软件方式实现算法,不仅效率低,而且会改变 cache 行为,影响评估结果.硬件方法以并行方式实现算法,不仅高效,而且不会影响 cache 行为,本文给出预测校正检索法的硬件设计.

如图 4 所示,iter-inst num 定义了一次迭代的指令数.Hit/miss energies 寄存器组保存不同参数对应的 cache 命中/失效功耗.Static energy 保存每周期静态功耗.Hit/miss/static energy, iter-inst num 由处理器初始化.Hit num, miss num, total cycle 记录系统运行信息,由 FSM 管理.

预测校正检索算法

Input: 容量 C, 块大小 B, 关联度 A

Output: a_{best}, b_{best}, c_{best}

begin:

1. 在 $L_{(C,b_1,a_1)}$, $L_{(c_1,B,a_1)}$, $L_{(c_1,b_1,A)}$ 上对 $f(l)$ 进行采样.

2.分析采样数据, 预测检索空间基底 c_{pbest}, b_{pbest}, a_{pbest}; 计算 δ_c, δ_b, δ_a, 确定参数检索顺序.

3. 按预测的参数检索顺序在校正的检索空间检索最优参数.

Output: a_{best}, b_{best}, c_{best}

图3 预测校正检索算法

控制信号决定是否进入检索状态.Flush 信号在每次迭代时重置 cache,避免前一次迭代载入的数据影响下一次采样.虽然 flush 会降低 cache 的性能,但这只发生在算法检索阶段.

图 5 描述 FSM 的详细实现.$P0$, $P1$, $P2$, $P3$ 为顶层状态机,对应初始化,数据采样,最优基底与权重分析,求解最优解这四个阶段.$P0$ 主要对信息系统的寄存器进行初始化;$P10$ 为采样循环控制,$P11$, $P12$, $P13$ 分别对应 $L_{(C,b_1,a_1)}$, $L_{(c_1,B,a_1)}$, $L_{(c_1,b_1,A)}$ 上 $f(l)$ 采样阶段;$P20$ 根据采样结果计算预测基底;$P21$ 计算 δ_c, δ_b, δ_a,确定参数检索顺序.$P30$, $P31$, $P32$ 分别检索最优容量、块大小、关联度,其检索空间与检索顺序由 $P2$ 决定.

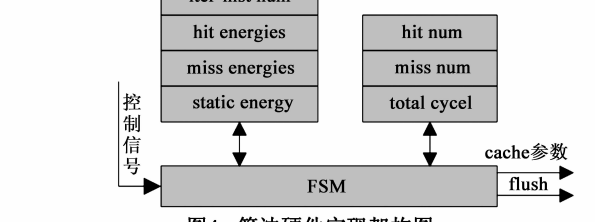

图4 算法硬件实现架构图

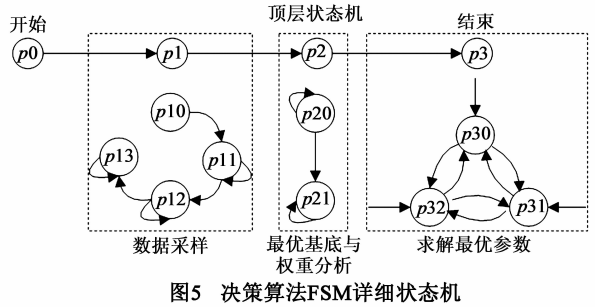

图5 决策算法FSM详细状态机

4 实验结果分析

为了评估算法的硬件开销,基于欧航局发布的开源处理器 Leon3[12]实现算法 RTL 代码.在 SMIC 0.18 μm

CMOS 工艺下用 Synopsys DC 综合,如表 3 所示,检索算法只增加了 3.4% 的面积与 9.2% 的功耗.

为了进一步验证算法的有效性,在 SimpleScalar 整合该算法并测试,实验数据见表 4(Iter-inst num 设为 5000 万,功耗寄存器按 2.2 节模型初始化).

表 3　综合结果对比

	原始 Leon3	改动后 Leon3(raito%)
面积	456325	471840(103.4%)
功耗	6.82mW	7.45mW (109.2%)

表 4　实验数据

	最优参数	降维检索	$c_{best}b_{best}a_{best}$	$c_{pbest}b_{pbest}a_{pbest}$	δ_c	δ_b	δ_a	预测校正
Bzip	2**k**:64:1	**2k:64:1**	$c_2\ b_3\ a_1$	$c_2\ b_3\ a_4$	0.16	0.10	0.23	**2k:64:1**
Gobmk	8**k**:256:1	16**k**:256:1	$c_4 b_5 a_1$	$c_5\ b_4\ a_3$	0.13	0.11	0.06	**8k**:128:2
Hummer	4**k**:32:4	8**k**:128:1	$c_3\ b_2\ a_3$	$c_4\ b_4\ a_3$	0.17	0.13	0.05	**4k:32:4**
Mcf	16**k**:256:1	32**k**:128:1	$c_5\ b_5\ a_1$	$c_6\ b_5\ a_3$	0.11	0.15	0.02	**16k**:128:4
Milc	16**k**:256:1	32**k**:128:2	$c_5\ b_5\ a_1$	$c_6\ b_5\ a_3$	0.12	0.15	0.02	32**k**:64:4
Sjeng	4**k**:128:1	2**k**:128:1	$c_3\ b_4\ a_1$	$c_2\ b_4\ a_2$	0.18	0.12	0.22	**4k:128:1**
adpcm_c	8**k**:256:1	**8k:256:1**	$c_4\ b_5\ a_1$	$c_4\ b_5\ a_2$	0.13	0.09	0.08	**8k**:128:1
adpcm_d	8**k**:256:1	**8k:256:1**	$c_4\ b_5\ a_1$	$c_4\ b_5\ a_2$	0.16	0.15	0.10	4**k**:128:2
Epic	2**k**:64:2	4**k**:128:1	$c_2\ b_3\ a_2$	$c_3\ b_5\ a_2$	0.20	0.13	0.23	4**k**:128:1
Unepic	**k**:128:1	8**k**:128:1	$c_2\ b_4\ a_1$	$c_4\ b_5\ a_3$	0.18	0.16	0.22	**2k:128:1**
gsm_de	8**k**:256:1	32**k**:128:4	$c_4\ b_5\ a_1$	$c_6\ b_5\ a_2$	0.11	0.16	0.01	**8k**:128:1
gsm_en	4**k**:128:1	**4k:128:1**	$c_3\ b_4\ a_1$	$c_3\ b_5\ a_2$	0.19	0.20	0.01	4**k**:128:1
jpeg_de	4**k**:128:1	8**k**:128:1	$c_3\ b_4\ a_1$	$c_4\ b_5\ a_1$	0.16	0.16	0.04	**4k:128:1**
jpeg_en	8**k**:128:2	16**k**:256:1	$c_4\ b_4\ a_2$	$c_5\ b_4\ a_1$	0.14	0.12	0.06	16**k**:256:1
Rasta	1**k**:256:1	**1k:256:1**	$c_1\ b_5\ a_1$	$c_1\ b_5\ a_2$	0.15	0.19	0.00	**1k:256:1**

在本文中,集合元素按递增顺序排列,如 $c_1 = 1 \cdots c_9 = 256$. 定义基底偏差系数 γ,如式(11),γ 越小表示与最优基底越接近,*Ind* 为元素下标. 由图 6 可知,除 bzip2, unepic 外,预测校正基底均优于降维检索基底 (c_1, b_1, a_1).

$$\gamma = \frac{|Ind_{c_{pbest}} - Ind_{c_{best}}|}{m} + \frac{|Ind_{b_{pbest}} - Ind_{b_{best}}|}{n} + \frac{|Ind_{a_{pbest}} - Ind_{a_{best}}|}{p} \tag{11}$$

降维检索有 5 个样本的结果与最优解一致,表中粗体所示,准确率为 33.3%. 预测校正检索有 7 个样本得到最优解,准确率为 46.7%,提升了 13.4%. 另外,容量是设计者最为关心的参数,预测校正检索有 11 个样本找到了最优容量,准确率为 73.3%,远高于降维检索的 33.3%.

图 7 显示了最优参数、降维检索、预测校正检索的 $f(l)$ 值. 降维检索、预测校正检索与最优 $f(l)$ 的平均偏差分别为 12.2% 与 7.2%,说明在一般情况下,预测校正检索结果优于降维检索.

图 8 反映了样本在 256**k**:32:4 通用 cache 与预测校正动态重构 cache 下的功耗与失效率. 动态重构 cache 通过选择合理的 cache 配置能够降低 90% 以上的功耗,

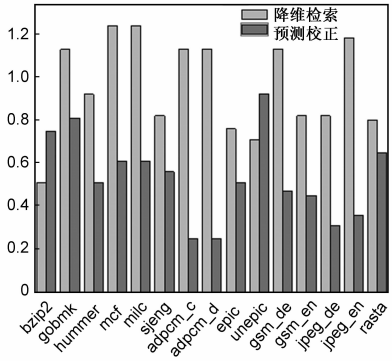

图6　降维检索与预测校正的 γ 值

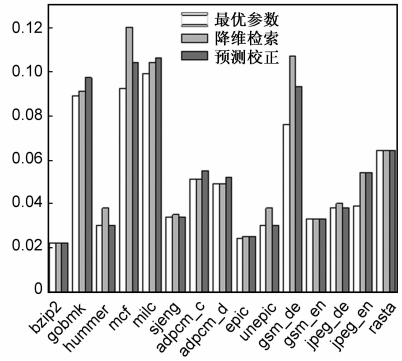

图7　最优参数/降维检索/预测校正 $f(l)$

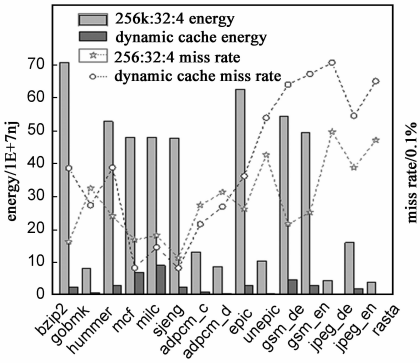

图8　功耗与失效率对比

而对失效率的影响不超过 6‰.

从算法复杂度分析,穷举法需 179 次迭代,降维检索需 18 次,预测校正检索需 36 次.相比穷举法,本算法减少了近 80% 的迭代次数.相比降维检索法,以增加迭代为代价换取了准确率的有效提高.由于检索算法并不需要一直运行,其性能损失的影响将被弱化.

5 结论

本文提出了一种采用预测校正检索算法的片上动态 cache 设计.该方法通过对运行时程序状态的监控,对参数检索空间与检索顺序做出预测校正,能够在兼顾效率的同时有效提高算法的准确率.该算法可以通过硬件并行实现,不影响 cache 行为,使得 cache 能够灵活适应多变的应用环境.

参考文献

[1] J L Hennessy, DA Patterson. Computer Architecture: A Quantitative Approach[M]. 4th ed. Beijing: Publishing House of Electronics Industry, 2007. 198 – 214.

[2] David H. Albonesi. Selective cache way: on – demand cache resource allocation[A]. Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture [C]. Haifa: IEEE Computer Society, 1999. 248 – 259.

[3] 张宇弘,王界兵,等.标志预访问和组选择历史相结合的低功耗指令 cache[J].电子学报,2004,32(8):1286 – 1289.
Zhang Yu-hong, Wang Jie-bing, et al. Pre-visiting tag and keeping way history to reduce power in instruction cache[J]. Acta Electronica Sinica, 2004, 32(8): 1286 – 1289. (in Chinese)

[4] Alexander V. Veidenbaum, Weiyu Tang, et al. Adaptingcache line size to application behavior[A]. Proceedings of the 13th international conference on Supercomputing [C]. Rhodes: ACM, 1999. 145 – 154.

[5] 贾宝锋,高德远,等.低功耗动态可配置 Cache 设计[J].计算机测量与控制,2008,16(7):1017 – 1020.
Jia Baofeng, Gao Deyuan, et al. Design of low power dynamically configurable cache[J]. Computer Measurement & Control, 2008, 16(7): 1017 – 1020. (in Chinese)

[6] 张毅,汪东升.一种嵌入式处理器的动态可重构 cache 设计[J].计算机工程与应用,2004,40(8):94 – 97.
Zhang Yi, Wang Dongsheng. A dynamic configurable cache design ofembedded processor[J]. Computer Engineering and Applications, 2004, 40(8): 94 – 97. (in Chinese)

[7] Chuanjun Zhang, Vahid, F, et al. A self-tuning cache architecture for embedded systems[J]. ACM Transactions on Embedded Computing Systems, 2004, 3(2): 407 – 425.

[8] Austin T, Larson E, et al. Simplescalar: an infrastructure system modeling[J]. IEEE Computer, 2002, 35(2): 59 – 67.

[9] 周宏伟,张民选.指令 cache 体系结构级功耗控制策略研究[J],电子学报,2008,36(11):2107 – 2112.
Zhou Hong-wei, Zhang Min-xuan. The research on power controlling policies for instruction cache with architecture leveI methods[J]. Acta Electronica Sinica, 2008, 36(11): 2107 – 2112. (in Chinese)

[10] HP Labs. CACTI: an integrated cache and memory access time, cycle time, area, leakage, and dynamic power model [DB/OL]. http://www.hpl.hp.com/research/cacti/, 2009.

[11] Yi-Ying Tsai, Chungho Chen. Energy-efficient trace Reuse cache for embedded processors[J]. IEEE Transactions on Very Large Scale Integration (VLSI) System, 2011, 19(9): 1681 – 1694.

[12] Jiri Gaisler. A portable and fault-tolerant microprocessor based on the SPARC V8 architecture [A]. Proceedings of the 2002 International Conference on Dependable Systems and Networks[C]. Washington, DC: IEEE Computer Society, 2002: 409 – 415.

作者简介

曹向荣(通信作者) 男,1980 年 3 月生于江苏启东.现为北京航空航天大学通信与信息系统博士研究生.主要从事导航一体化 SoC 芯片研究.
E-mail: jackycxr@163.com

张晓林 男,1951 年 1 月生于北京市.北京航空航天大学教授、博士生导师,国家卫星导航应用工程研究中心副主任、教育部国家集成电路人才培养基地负责人.主要研究方向为集成电路设计、飞行器遥测遥控与卫星导航系统.
E-mail: zxl@buaa.edu.cn