一种低功耗动态可重构 cache 算法的研究

任小西, 刘　清
(湖南大学 信息科学与工程学院, 长沙 410082)

摘　要: 动态可重构 cache 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 的访问进行预判,然后根据预判的结果确定当前程序段的 cache 结构。实验结果表明,此算法比传统四路组相联 cache 功耗降低 61%,而性能损失只有 2% 左右。与已有算法相比,功耗和性能都得到进一步的提高。

关键词: 低功耗; 可重构 cache; 程序段
中图分类号: TP302　　**文献标志码**: A　　**文章编号**: 1001-3695(2013)02-0414-03
doi:10.3969/j.issn.1001-3695.2013.02.026

Study of dynamically reconfigurable algorithm for low-power cache

REN Xiao-xi, LIU Qing
(*School of Information Science & Engineering*, *Hunan University*, *Changsha* 410082, *China*)

Abstract: Pynamically reconfigurable the algorithm used the instruction-cycle to detect phase changes. In each phase, the state machine used the average access time to predict the cache access and obtained optimal configuration. The experimental results show that the algorithm can reduce cache power dissipation by 61%, but increasing CPI by 2% on average compared with four-way conventional cache, and further saves the energy consumption and reduces the performance loss compared to the previous works.

Key words: low power consumption; reconfigurable cache; program phase

随着嵌入式微处理器的集成度和速度的发展,性能得到了极大地提升。然而,随着社会的进步和发展,能量的消耗在不断地增加,特别是集成电路芯片的能量消耗。因此,低功耗设计日渐成为嵌入式微处理器与数字信号处理器中的主要目标。而 CPU 运算速度和主存速度不匹配的情况在逐渐地扩大,因此由高速缓存 cache 来解决主存与 CPU 之间速度不匹配的问题。Cache 的能耗在嵌入式系统中占了非常大的比重[1],因此低缺失率、低功耗 cache 成为了研究的方向。

程序的时间和空间局部性原理是当前 cache 技术产生和发展的前提条件。随着低功耗 cache 成为人们研究的热点,降低 cache 功耗的技术就层出不穷。Inoue 等人[2]提出了路预测 cache 算法。其主要思想是直接访问预测表中的预测路、预测命中,把数据送给 CPU,预测失败,则访问 cache 中其他路的 tag 和 data。Raveendran 等人[3]提出的基于预测放置方法的路预测算法,通过增加最小预测位来减少 tag 的比较次数,增加路预测的准确率,从而减少 cache 的能量消耗。预测命中能降低功耗,预测失败则增加访问时间,性能损失,并且功耗不会降低。因此,预测的准确率是十分重要的。文献[4]提出了一种新的基于分支预测的方法,它主要是通过减少 branch target buffer 的不必要的访问来降低动态功耗以及改变选择 branch target buffer 组的模式来降低静态功耗。文献[5]在传统 filter cache 的基础上提出了带预测的多线缓冲器的指令 cache 方法。它通过预测机制预测 CPU 的访问是在指令 cache 还是多线缓冲器。多线缓冲器相当于 filter cache,同样减少了对指令 cache 的访问,降低了功耗。Hsu 等人[6]提出了功耗和性能最优化的 cache 架构,它有效地利用嵌入式系统中 cache 的设计空间去寻找适合功耗和性能最优化的 cache 容量大小。实际上,随着程序的运行,cache 的最优结构会不断地改变。所以,动态调节 cache 参数适应不同程序的需求能够有效地改善 cache 的性能和功耗。动态可重构 cache 技术是在程序运行过程中根据程序的需求动态地调整 cache 的结构,关闭 cache 中闲置未用部分的能量消耗,从而在对性能造成最小影响的状况下,较好地提高能量效率。

Alipour 等人[7]提出了一个可重构数据 cache 架构,有两个模式,即正常的 cache 模式和基于行的流处理模式,并考虑在不同的 filter 大小和行之间的数据重利用,是为了把数据访问引入到片外的 DRAM。郝玉艳等人[8]提出了一种基于程序段的自适应算法(PBSTA)。它通过工作集来监测程序段的变化,在程序段发生变化后,数据 cache 和指令 cache 分别根据动态配置策略来自适应地调整相应 cache 的关联度,从而重新配置相应 cache 的结构。数据 cache 和指令 cache 都用同样的重构方式进行重构,可以有效减少整个 cache 消耗的能量。Bani 等人[9]提出了一个新的可重构嵌入式数据 cache(RED)。其根据系统中不同应用程序的需求,RED 通过一个模式选择机制配置为直接映射,2 路或 4 路组相联。这个架构是用 HDL 语言实现,达到了最大运行频率。

1　动态可重构 cache 方案

1.1　程序段的监测机制

不同的程序甚至同一程序的不同运行阶段对 cache 的需

收稿日期: 2012-07-08; **修回日期**: 2012-08-16

作者简介: 任小西(1978-),女,湖南长沙人,副教授,博士,主要研究方向为嵌入式系统及编译器(happyrxx@hnu.cn);刘清(1987-),女,湖南岳阳人,硕士,主要研究方向为嵌入式系统.

求均不同,因此当程序的运行状态发生明显改变时是 cache 重构的最好时机。因此需要分析程序本身 cache 的一些行为特征。判断程序段的行为特征的参数有很多,如 cache 缺失率、程序执行指令数、工作集、程序运行时间、分支跳转数目等。

指令时间数结合了指令数和程序运行的周期,计算公式如式(1)所示,既能明显地反映出程序的变化,又能减少不必要的重构,因此采用指令时间数作为程序段监测的参数。

$$\text{指令时间数} = \frac{\text{程序执行的周期}}{\text{程序执行的指令数}} \tag{1}$$

整个监测机制可以用图 1 表示。

监测状态机有两个状态,即初始态和容量调整态。初始态获取当前程序段的分支指令频率,条件 A 是指分支指令频率(CBR)小于分支指令频率阈值(CBRL),此时状态保持不变;条件 B 是指 CBR 大于 CBRL,状态转换到容量调整态。在容量调整态,算法寻找程序段内 cache 的优化配置,条件 C 是指在找到优化配置后回到初始态,继续检测程序段的变化。

1.2　动态配置策略

可重构算法的动态配置策略是可重构算法的关键,因此需要通过动态配置策略完成 cache 结构的重新配置。程序段监测只是确定程序段是否发生了改变,而程序段的改变不一定说明程序需要调整 cache 结构以及如何调整,因此 BRDRC 算法的动态配置策略的目的主要有两个:a)在什么情况下确定重新配置 cache 的结构;b)用何种方式来改变 cache 的结构。

1)预判机制　在进入容量调整态后,需要确定当前阶段 cache 结构,因此引入了预判机制。预判机制主要是对额外路的命中情况和 LRU 路的命中情况进行统计,然后确定是增大容量、减小容量还是保持 cache 结构不变。

使用预判机制时 cache 的结构并没有改变,因此在可能增大容量时开启额外路,额外路命中时 cache 命中次数加 1;而在可能减小容量时如果 LRU 路命中,cahce 的缺失次数加 1。

2)可重构状态机　当判断程序发生变化后,就需要进一步地确定是否需要增大或减小相联度来重构 cache。因此本文构造了如图 2 所示的状态机。为了在不同状态下进行转换,需要定义一些性能参数:

a)cache 访问次数(CAN),记录 cache 每个状态下的访问次数。

b)cache 访问次数阈值(CAL),记录 cache 访问次数的阈值。

c)平均访问时间(AAT),记录程序段内 cache 的平均访问时间。计算公式为

$$\text{AAT} = 1 + \frac{t_{\text{miss}} \times N_{\text{miss}} + t_{\text{wb}} \times N_{\text{wb}}}{N_{\text{total}}} \tag{2}$$

其中:1 指访问 cache 命中时需要的时钟周期数为 1;t_{miss} 是访问 cache 缺失需要的时钟周期数;N_{miss} 是访问 cache 缺失的次数;t_{wb} 是块写回需要的时钟周期数;N_{wb} 是块写回的次数;N_{total} 是总的访问次数。

d)平均访问时间阈值(AATL),判断 cache 程序段内 cache 平均访问时间的临界值。它有两个阶段的作用:确定是进入预增大容量的运行还是预减小容量的运行;确定是否重新配置。

图 2 说明了算法的具体实现。状态机由五元素构成,即状态集 J、输入集 K、转移函数 T、初始态 S 和终态集 F。

状态集为非空,有初始状态 S_0、重构预减小状态 S_1、重构预增大状态 S_2、重构状态 S_3。

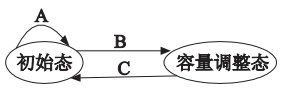

图1　程序段监测状态机

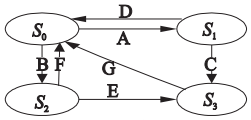

图2　IC-DRC算法的状态图

输入集包括条件 A、B、C、D、E、F、G。条件 A 指 ICN ≥ ICNL 并且 AAT < AATL;条件 B 指 ICN ≥ ICNL 并且 AAT ≥ AATL;条件 C 指 AAT ≥ AATL;条件 D 指 CAN ≥ CAL 并且 AAT < AATL;条件 E 指 AAT ≥ AATL;条件 F 指 CAN ≥ CAL 并且 AAT < AATL;条件 G 指重构操作。条件 D 和 F、C 和 E 在预判机制下,计算 cache 的访问次数和缺失次数的方式有所不同,因而 D 和 F、C 和 E 有所不同。

转移函数指在每个状态下,当对应的条件发生时转移到相应的状态下去,表示为 $J \times K \rightarrow J$。

初始态是 S_0,非空。

终态集是 S_0,非空。

算法中 CALAATL 和 ICNL 阈值决定了重构的难易程度,过大或过小都会影响本文算法的执行效果以及性能的损耗。因此通过动态调整 CAL、AATL 和 ICNL 的值来更加准确地确定重构:

a)当在重构预减小状态和重构预增大状态下,如果发现增大容量或减小容量后都会带来一定的性能损耗而回到初始状态时,那么在这个程序段内不需要进行重构,因而阈值可能过小,所以需要增大 AATL、CAL 和 ICNL 阈值。

b)当在重构预减小状态和重构预增大状态下,如果发现容量发生变化后性能得到提高而进入重构态时,那么说明 cache 结构发生了变化,阈值就可能不是特别准确,因此需要减小 AATL、CAL 和 ICNL 阈值。

1.3　硬件设计分析

在算法实现的过程中,CAN 和 CMN 以及阈值分析所需要的开销都很小,只需增加相应的寄存器作累加操作,并且 AAT 和 ICN 都可以从现有的计数器中获取,因此硬件开销比较小。算法中增加了一个额外路的 tag 开启的操作,这个操作会带来一定的能耗和面积开销,但只有在重构预增大状态时才对额外路的 tag 进行访问,同时 tag 开启消耗的能量比 data 开启要小很多,这样可以大大减小硬件开销和时延。

2　实验环境

实验主要采用模拟的方法在性能和功耗上进行估算。本文算法主要在 sim_panalyzer 上模拟。它是由 Michigan 和 Colorado 大学共同开发的一个工作集,目的是为了创造一个能耗模拟器,能检测能耗和性能的平衡。Sim_panalyzer 是在 SimpleScalar[10] 的 sim_outorder 模拟器组件上实现的,SimpleScalar 只能获取测试程序的性能数据。本文选择的目标机器是 ARM 指令集架构,研究的是可重构 cache 技术,因此需要修改 sim_panalyzer 软件,使其能支持可重构 cache 技术。

使用 sim-panalyzer 构建的低功耗动态可重构 cache 算法的仿真环境如图 3 所示,通过 output files 可以获得程序的功耗和性能方面的数据。在实验中,仅对一级指令 cache 进行重构,采用了指令和数据分离的 cache 结构,一级指令和数据 cache 的容量都是 32 KB,并且参数相同,都为四路组相联,块大小为 32 Byte,组数为 256。一级指令 cache 采用 LRU 替换策略,相联度在 1、2 和 4 路之间变化。本文选取的测试程序集是 Mibench,它在嵌入式系统领域成为了教学科研的常用工具。

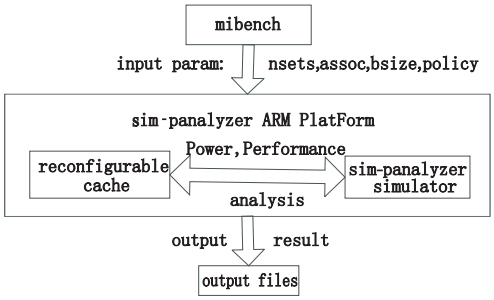

图3 可重构cache的sim_panalyzer模拟框图

3 实验结果

为了比较本算法(IC-DRC 算法)的性能和功耗情况,本文选取 PBSTA 和基于 Toumament cache 的动态可重构算法(TCDRA)[11]作为参考模型。TCDRA 分为正常模式、小模式和大模式三个模式,在这三种模式下根据三个计数器监测不同程序段的 cache 访问情况,确定相应程序段的 cache 相联度,然后用一个配置寄存器重新配置 cache。TCDRA 和 PBSTA 都是采用路关闭策略重新配置 cache 结构,因此具有可比性。

本文采用平均访存功耗来衡量 cache 的功耗。以未采用重构算法的 cache 结构的功耗为基准,将 IC-DRC 算法、PBSTA 和 TCDRA 的功耗进行归一化处理,得到图 4,图中横坐标为 Mibench 的测试程序。

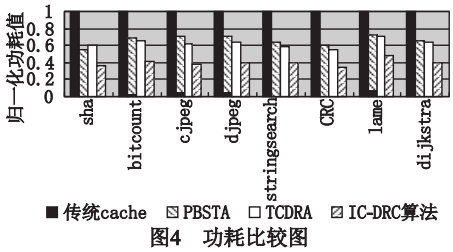

图4 功耗比较图

从图 4 可以看出,IC-DRC 算法的功耗明显优于 TCDRA 和 PBSTA,在 TCDRA 指导下,除了 sha 程序的功耗低于 PBSTA 外,其他测试程序都优于 PBSTA 一点点。IC-DRC 算法的功耗降低了 61%,TCDRA 功耗降低了 38%,PBSTA 的功耗降低了 35%。PBSTA 通过指令工作集签名监测程序段的状态,当程序段监测机制监测到程序段发生变化而使程序段监测状态机进入容量调整状态,在此状态中如果监测到工作集不稳定,表明程序段可能发生了改变而使程序段监测状态机回到不稳定状态,容量调整状态下的所有操作都是无效的,从而增加一些不必要的功耗。PBSTA 下容量调整状态的运行长度在 2 048 ~ 8 192 之间,而 IC-DRC 算法在程序段内的试运行新配置的长度在 150 ~ 300 之间,因此无效操作的时间比较长。而 TCDRA 确定 cache 的优化配置需要在大模式和小模式之间循环试运行 cache 的新配置,因此程序段的长度比较长,容易错过重构时机。

本文利用算法的 CPI(指令的周期数)的比值来得出归一化延迟,以传统 cache 的 CPI 为基准,如图 5 所示。

对于 PBSTA,sha、bitcount 和 CRC 程序的性能基本不变,而其他程序性能损失比较明显,从 1% ~8%。TCDRA 的 dijkstra 程序的性能损失达到了 18%,其他程序和 IC-DRC 算法下相应的程序的性能损失基本持平。IC-DRC 算法除了 bitcount 的性能损失高于 PBSTA, stringsearch 和 lame 的性能损失高于 TCDRA 外,其他测试程序基本要低于这两个算法。IC-DRC 算法的性能损失只有 2%。

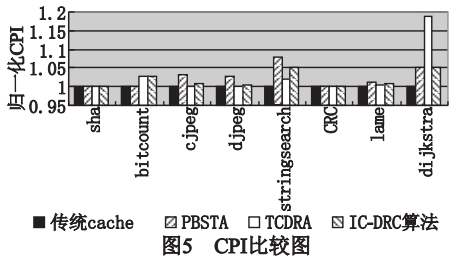

图5 CPI比较图

综上所述,IC-DRC 算法的性能损失很少,并且明显地降低了功耗。

4 结束语

低功耗动态可重构 cache 算法使用指令时间数作为程序段的监测参数,然后通过基于平均访问时间 AAT 的预判机制来动态改变 cache 的相联度,从而重新配置 cache 结构,并且通过在 sim-panalyzer 上模拟,获得性能和功耗方面的数据。该算法增加了很少的参数,并在程序运行中动态地配置参数,使得程序段长度可以动态改变,重构时机也更加准确。通过实验验证,算法有效地降低了功耗,性能损失很少。

参考文献:

[1] 文桦,张亚军. 嵌入式系统低功耗设计研究[J]. 现代电子技术,2009,32(22):20-25

[2] INOUE K,ISHIHARA T,MURAKAMI K. Way-predicting set-associative cache for high perfomance and low energy consumption[C]// Proc of International Conference on Low Power Electronics and Design. New York: ACM Press,1999:273-275.

[3] RAVEENDRAN B K, SUDARSHAN T S B,PATIL A,*et al*. Predictive placement scheme in set-associative cache for energy efficient embedded systems[C]//Proc of International Conference on Signal Processing, Communications and Networking. Washington DC: IEEE Computer Society, 2008:152-157.

[4] LEVISON N, WEISS S. Low power branch prediction for embedded application processors[C]//Proc of the 16th ACM/IEEE International Symposium on Low-Power Electronics and Design. New York: ACM Press,2010:67-72.

[5] ALI K,ABOELAZE M,DATTA S. Energy efficient I-cache using multiple line buffers with prediction[J]. Computers&Digital Techniques,2008,2(5):355-362.

[6] HSU P H, CHIEN S Y. Reconfigurable cache memory architecture for integral image and integral histogram applications[C]//Proc of International Conference on Signal Processing Systems. Washington DC: IEEE Computer Society,2011:151-156.

[7] ALIPOUR M, MOSHARI K, BAGHERI M R, *et al*. Performance per power optimum cache architecture for embedded applications, a design space exploration[C]//Proc of the 2nd IEEE International Conference on Networked Embedded Systems for Enterprise Applications. Washington DC: IEEE Computer Society, 2011:1-6.

[8] 郝玉艳,彭蔓蔓. 混合 Cache 的低功耗设计方案[J]. 计算机工程与应用,2009,45(20):68-70.

[9] BANI R R, SARAJU P M, ELIAS K, *et al*. Design of a reconfigurable embedded data cache[C]//Proc of International Symposium on Electronic System Design. Washington DC: IEEE Computer Society, 2010:163-168.

[10] AUSTIN T,LARSON E,EERST D. Simplescalar: an infrastmcture for computer system modeling[J]. IEEE Computer,2002,35(2):59-67.

[11] 赵欢,苏小昆,李仁发. 一种低功耗动态可重构 Cache 方案[J]. 计算机应用,2009,29(6):1446-1448.