

基于分支执行历史的循环缓冲低功耗方法

李 泉 泉^{1,2}, 张 铁 军², 王 东 辉², 侯 朝 焕²

(1 中国科学院大学, 北京 100049; 2 中国科学院 声学研究所, 北京 100190)

摘 要: 针对嵌入式处理器中指令 Cache 功耗显著的特点, 提出了一种基于分支执行历史的循环缓冲低功耗方法. 利用分支指令当前信息与分支执行历史信息之间的关系, 实现了应用程序中循环的动态检测与加载. 通过对取指通道的精确控制, 该方法能够过滤大部分不必要的指令 Cache 访问, 有效降低了指令 Cache 的功耗. 在 SuperV_EF01 DSP 上的实验结果表明, 采用该方法后, 在处理器性能没有损失的情况下, 指令 Cache 功耗平均降低 32.58%, 面积仅增加 8.31%.

关键词: 分支执行历史; 低功耗; 循环缓冲; 指令 Cache

中图分类号: TP302

文献标识码: A

文章编号: 1000-7180(2014)09-0007-04

Branch Execution History Based Low Power Loop Buffering

LI Quan-quan^{1,2}, ZHANG Tie-jun², WANG Dong-hui², HOU Chao-huan²

(1 University of Chinese Academy of Sciences, Beijing 100049, China; 2 Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Instruction cache is one of the main power consumption components in embedded processor. This paper proposes a loop buffering method based on branch execution history to save the instruction cache power consumption. The loops in application program are detected and loaded dynamically by using the relationship between branch current information and branch execution history. With precise control of the instruction fetch datapath, this approach can filter a majority of unnecessary instruction cache accesses, thus saving the instruction cache power consumption significantly. Experimental results of SuperV_EF01 DSP show that this approach could save 32.58% of instruction cache power consumption, with only 8.31% of instruction cache area increasing and no performance degradation.

Key words: branch execution history; low power; loop buffering; instruction cache

1 引言

随着半导体工艺技术的发展, 处理器芯片的性能和集成度都得到了很大提升, 由此带来的功耗问题也变得很突出. 特别是对于嵌入式处理器, 由于便携性和电池容量的原因, 功耗已经成为设计的重要限制因素. 作为提高处理器内核取指性能的关键部件, 指令 Cache 由于具有很高的访问频率而功耗显著, 如 Strong-ARM SA110 中指令 Cache 的功耗约占处理器总功耗的 27%^[1]. 有效降低指令 Cache 的功耗对于低功耗嵌入式处理器的设计有着重要

意义^[2].

在嵌入式应用程序中, 循环是一种主要的组织形式^[3]. 对 Powerstone 基准测试程序^[4]的分析表明, 32 条指令以内的短循环执行时间占用了整个程序执行时间的 51%^[5]. 基于应用程序呈现的这一特征, 研究人员提出了多种低功耗指令 Cache 设计方法. 其中主流的方法是通过在处理器内核和指令

Cache 之间增加一个小的缓冲区来减少指令 Cache 的访问次数, 从而降低指令 Cache 的功耗^[6]. Filter Cache^[7]的结构与普通 Cache 相同, 只是容量更小; 但是它的缺失率较高, 降低了处理器的性能.

Loop Cache^[8]根据指令代码中的短后向跳转指令来识别循环,但是它不能处理循环体内包含分支的情况;文献[9]中对其进行改进,但是需要的硬件代价太大.预加载 Loop Cache^[10]将占用执行时间最多的循环预先加载到循环缓冲区中,对于特定的应用程序,可以达到很好的降低指令 Cache 功耗的效果,但是该方法不是动态的,使用范围很受限.

针对现有方法存在的问题,本文提出一种基于分支执行历史的循环缓冲低功耗方法.该方法利用分支指令的信息动态检测和加载应用程序中的循环,在处理器性能没有损失的情况下,以较小的硬件代价减少指令 Cache 的访问次数,降低指令 Cache 的功耗.

2 循环缓冲结构设计

本文提出的循环缓冲技术实现算法如图 1 所示.它的工作原理是:当有效循环分支指令第二次执行且第一次执行与第二次执行期间没有执行其它分支指令时,开始加载循环;当有效循环分支指令第三次执行且第二次执行与第三次执行期间没有执行其它分支指令时,循环加载完成;循环加载完成后指令 Cache 关闭,处理器内核从循环缓冲中取指令.

```

while(branCh instructioyz is encountered)
    if(branch is not a valid loop brabch)
        loop loading is stopped;
        /* braYzch counter ano( b, anch address register is
        cleared */
    else if(branch target is not in the loop)
        if(hit the branch address register)
            branch counter is increased by one;
            if(the counter value is one)
                loop loading is sta, - ted;
            else
                loop load e d is finished;
                goto LABELI;
        else
            loop loading is stopped;
            /* sto, e the b, anch Clddr ess / ito the br Clnch add
            ess , egister */
    else
        LABELI; fetch instructio l from the loop buffer;
        if(fetch address is out of the loop)
            loop buffer is disabled;
            /* fetch i, tist, action jrom the i, is truc t ion
            cache */

```

图 1 循环缓冲技术实现算法

循环检测与加载控制电路如图 2 所示,其中分支偏移高位位宽根据分支指令的分支偏移量位宽和循环缓冲容量来确定.当分支偏移高位全为 1 时,可

以确定该分支指令转移的最大距离;如果该分支指令的最大转移距离小于循环缓冲的容量,则将该分支指令作为有效循环分支指令.分支地址寄存器用来记录已执行有效循环分支指令的地址.

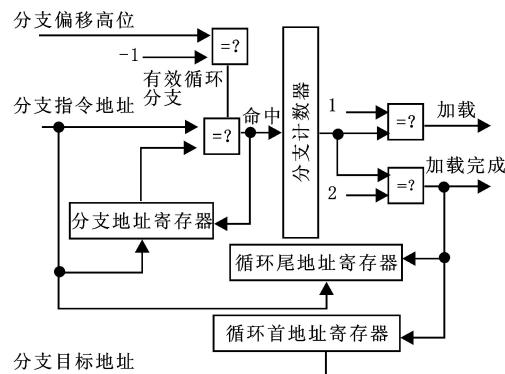


图 2 循环检测与加载控制电路

在应用程序执行过程中,如果检测到分支偏移高位的值为-1,即它的每一位都为 1,则该分支指令为有效循环分支指令.如果该分支指令的地址与分支地址寄存器中的值相等,则分支计数器加 1;否则将该分支指令的地址记录到分支地址寄存器中,并将分支计数器清零.当分支计数器的值为 1 时,开始加载循环;当分支计数器的值为 2 时,循环加载完成.当检测出分支指令不为有效循环分支指令时,分支计数器清零,分支地址寄存器清零.循环加载完成后,将循环首地址和循环尾地址分别存入循环首地址寄存器和循环尾地址寄存器中.在循环加载过程中,如果执行了其他分支指令,则循环加载失败;此时循环缓冲的内容已经被修改,它存储的已经不再是原来的循环.因此,开始加载循环时需要将循环首地址寄存器和循环尾地址寄存器清零,表示现在开始加载新的循环.

循环缓冲的工作状态转换图如图 3 所示.在空闲状态,循环缓冲不工作,处理器内核从指令 Cache 中取指令.当循环检测与加载控制电路输出的加载信号有效时,状态机从空闲状态转换到加载状态开始加载循环;在加载状态,处理器内核从指令 Cache 中取指令,从指令 Cache 中返回的指令发送给处理器内核的同时写入循环缓冲中.当循环检测与加载控制电路输出的加载完成信号有效时,循环加载完成,状态机从加载状态转换到工作状态,处理器内核开始从循环缓冲中取指令,指令 Cache 关闭.当检测到取指地址大于循环尾地址寄存器中的值时,循环执行完成,状态机从工作状态转换到空闲状态,处理器内核开始从指令 Cache 中取指令.

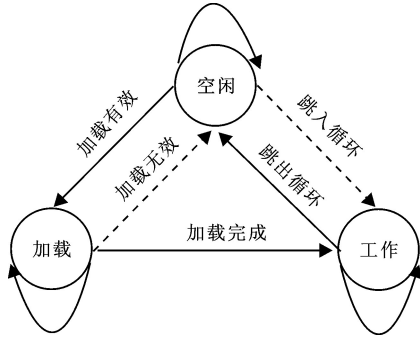


图3 工作状态转换图

在加载状态,如果执行了其他分支指令,即循环体内包含分支指令并在加载过程中执行,循环检测与加载控制电路输出的加载信号无效.为了使得循环加载正确,状态机会从加载状态转换到空闲状态,重新开始检测有效循环分支指令.在工作状态,如果执行了其他分支指令且该分支指令的目标指令地址不处于循环缓冲中循环体的地址范围内,状态机从工作状态转换到空闲状态;在空闲状态,如果检测到分支指令且该分支指令的目标指令地址处于循环缓冲中循环体的地址范围内,状态机从空闲状态转换到工作状态.

对于嵌套循环,如果最内层循环加载到了循环缓冲中,则可能出现同一段循环代码重复加载到循环缓冲中的情况.为了避免这种情况的发生,在空闲状态会检测有效循环分支指令的地址是否与循环尾地址寄存器中的值相等;如果相等,则表示该循环已经加载到了循环缓冲中,状态机从空闲状态转换到工作状态,处理器内核直接从循环缓冲中取指令.

3 指令 Cache 功耗分析

传统指令 Cache 结构消耗的能量可以近似为式(1),其中 N_1 为处理器内核访问指令 Cache 的次数, N_2 为出现指令 Cache 缺失的次数, E_{IC_r} 为读一次指令 Cache 平均消耗的能量, E_{IC_w} 为写一次指令 Cache 平均消耗的能量.

$$E_{IC} = N_1 \times E_{IC_r} + N_2 \times E_{IC_w} \quad (1)$$

加入循环缓冲后,指令 Cache 的访问次数会减少,减少的次数为循环缓冲的访问次数.由于循环缓冲中的内容实际上是指令 Cache 的一个子集,它只是改变了处理器内核的取指通道,因此它对处理器内核的取指次数没有影响,对指令 Cache 的缺失次数也没有影响.新的指令 Cache 结构消耗的能量可以近似为公式(2),其中 N_3 为加入循环缓冲后处理器内核访问指令 Cache 的次数, N_4 为出现循环缓冲

加载的次数, E_{LB_r} 为读一次循环缓冲平均消耗的能量, E_{LB_w} 为写一次循环缓冲平均消耗的能量.新的指令 Cache 结构相对传统指令 Cache 结构获得的能量节省比例可以表示为式(3).循环缓冲加载次数 N_4 相对 N_1 很小,可以忽略不计.由于循环缓冲容量比指令 Cache 容量小的多,因此 E_{LB_r} 比 E_{IC_r} 小的多. μ 主要受 N_3 的影响,即处理器内核访问指令 Cache 的次数越少,获得的能量节省比例越大.

$$E_{IC_{new}} = (N_1 - N_3) \times E_{LB_r} + N_4 \times E_{LB_w} + N_3 \times E_{IC_r} + N_2 \times E_{IC_w} \quad (2)$$

$$\begin{aligned} \mu &= \frac{E_{IC} - E_{IC_{new}}}{E_{IC}} \\ &= \frac{(N_1 - N_3) \times (E_{IC_r} - E_{LB_r}) - N_4 \times E_{LB_w}}{E_{IC}} \end{aligned} \quad (3)$$

4 实验结果

以 SuperV_EF01^[11] 数字信号处理器 (digital signal processor, DSP) 为基础,设计实现了本文提出的循环缓冲低功耗方法. SuperV_EF01 DSP 片内包含一个数据位宽为 128 bit、容量为 8 kB 的直接映射方式的指令 Cache,工作频率为 500 MHz.将循环缓冲的数据位宽设计为 128 bit,与指令 Cache 数据位宽相同.实验所用的测试程序来自于 DSPstone 基准测试程序和 Powerstone 基准测试程序,其分支指令占总指令的比例如图 4 所示.

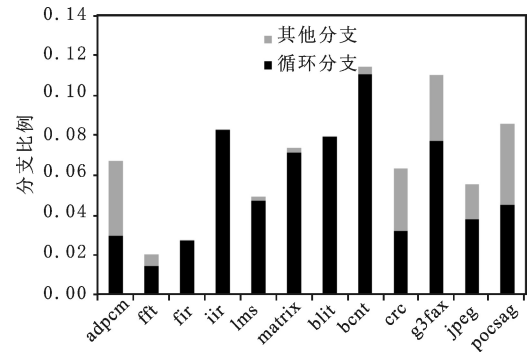


图4 分支指令占总指令的比例

为了说明循环缓冲大小对指令 Cache 访问次数的影响,实验测试了循环缓冲大小从 8 条指令到 256 条指令变化时,指令 Cache 的访问次数,如图 5 所示.对于 DSPstone 基准测试程序,当循环缓冲大小为 64 条指令后,指令 Cache 访问次数变得稳定;对于 Powerstone 基准测试程序,当循环缓冲大小为 32 条指令后,指令 Cache 访问次数就趋于稳定,这与文献[7]中的结果相吻合.

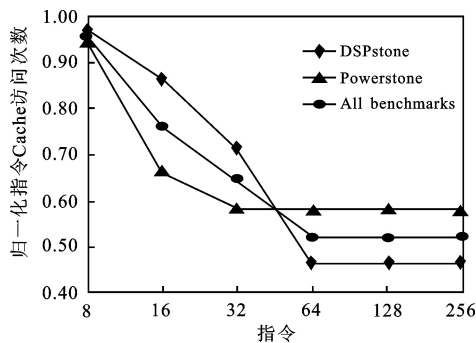


图5 循环缓冲大小的影响

在设计中选择循环缓冲大小为 64 条指令,实验测试了采用本文方法后指令 Cache 的功耗和测试程序的执行时间,以传统指令 Cache 结构下的测试结果为基准,对采用本文方法后的测试结果进行归一化,如图 6 所示.采用本文方法后,测试程序的执行时间没有发生变化,指令 Cache 的功耗平均降低了 32.58%.在 SMIC 90 nm 工艺下,采用本文方法后指令 Cache 的面积为 $264862.08\mu\text{m}^2$,与传统指令 Cache 相比,面积仅增加 8.31%.

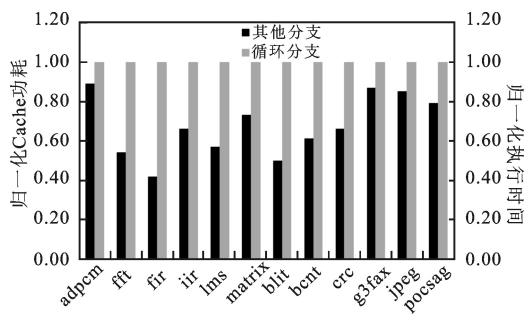


图6 对功耗和性能的影响

5 结束语

本文针对嵌入式应用程序中普遍存在短循环的特点,提出了一种基于分支执行历史的循环缓冲低功耗方法.该方法的硬件开销很小,能够动态检测和加载应用程序中的循环,且能够有效处理循环体内包含分支指令的情况和嵌套循环的情况.在 SuperV_EF01 DSP 上的实验结果显示本文方法在处理器性能没有损失的情况下可以平均节省 32.58% 的指令 Cache 功耗,面积仅增加 8.31%.

参考文献:

[1] Montanaro J, Witek R T, Anne K, et al. A 160-

MHz, 32-b, 0.5-W CMOS RISC microprocessor[J]. Journal of Solid-State Circuits, 1996, 31(11): 1703-1714.

[2] 朱伟成,周莉,喻庆东.一种低功耗高效率的 AHB-AXI 双总线结构联合 Cache 的 IP 设计[J].微电子学与计算机,2012,29(5):46-49.

[3] 孟建熠,严晓浪,葛海通,等.基于指令回收的低功耗循环分支折合技术[J].浙江大学学报,2010,44(4):632-638.

[4] Malik A, Moyer B, Cermak D. A low power unified cache architecture providing power and performance flexibility[C]//Proceedings of the 2000 International Symposium on Low Power Electronics and Design. Rapallo, Italy: ACM Press, 2000: 241-243.

[5] Villarreal J, Lysecky R, Cotterell S, et al. A study on the loop behavior of embedded programs[R]. Riverside: University of California. Department of Computer Science and Engineering, 2002.

[6] 龚帅帅,吴晓波,孟建熠,等.基于历史链接关系的指令高速缓存低功耗方法[J].浙江大学学报,2011,45(3):467-471.

[7] Kin J, Gupta M, Mangione-Smith W H. Filtering memory references to increase energy efficiency[J]. IEEE Transactions on Computers, 2000, 49(1): 1-15.

[8] Lee L H, Moyer B, Arends J. Instruction fetch energy reduction using loop caches for embedded applications with small tight loops[C]//Proceedings of the 1999 International Symposium on Low Power Electronics and Design. USA: San Diego, CA, ACM Press, 1999: 267-269.

[9] Wu C T, Hsieh A C, Hwang T T. Instruction buffering for nested loops in low-power design[J]. IEEE Transactions on Very Large Scale Integration Systems, 2006, 14(7): 780-784.

[10] Gordon-Ross A, Cotterell S, Vahid F. Exploiting fixed programs in embedded systems: a loop cache example[J]. IEEE Computer Architecture Letters, 2002, 1(1): 2-5.

[11] 王琪,高瑛珂,华斯亮,等.可复用微处理器片上调试功能的设计与实现[J].计算机辅助设计与图形学学报,2012,24(10):1369-1374.

作者简介:

李泉泉 男,(1987-),博士研究生.研究方向为嵌入式处理器体系结构、数字集成电路设计与验证.