指令 cache 体系结构级功耗控制策略研究

周宏伟[1,2],张民选[1]
(1. 国防科技大学计算机学院,湖南长沙 410073;2. 天津航海仪器研究所,天津 300131)

摘 要: 随着工艺尺寸缩小及处理器频率提高,功耗问题已成为当代微处理器设计面临的主要挑战.传统的指令 cache(I-Cache)功耗控制策略一般只单独降低指令 cache 的动态或者静态功耗.提出的两种改进的功耗控制策略,基于昏睡指令 cache 体系结构,能够更有效地同时降低指令 cache 的动态和静态功耗.一种称作"使用双预测端口路预测器的多路路预测策略",另一种称作"基于分阶段访问 cache 的按需唤醒预测策略",分别用于处理器前端流水线级数保持不变和可以增加额外前端流水线级数两种情形.实验结果表明:与传统的策略相比,提出的两种策略具有更优的能量效率,可以在不显著影响处理器性能的前提下,更有效地降低指令 cache 和处理器的功耗.

关键词: 指令; cache; 功耗; 体系结构
中图分类号: TP302.1 **文献标识码**: A **文章编号**: 0372-2112 (2008) 11-2107-06

The Research on Power Controlling Policies for Instruction Cache with Architecture Level Methods

ZHOU Hong-wei[1,2], ZHANG Min-xuan[1]
(1. *Department of Computer Science, National University of Defense Technology. Changsha, Hunan* 410073, *China*;
2. *Tianjin Navigation Instrument Research Institute. Tianjin* 300131, *China*)

Abstract: As feature size shrinks and the frequency increases, power dissipation has become the main restriction on microprocessor design. The traditional power controlling policies for instruction cache (I-Cache) are used for reducing the dynamic access power or the leakage power respectively. Two improved power controlling policy are proposed to reduce the dynamic and leakage power at the same time more efficiently. One is called "Multi-Way Way Prediction (MWWP) policy with a Two Prediction-ports Way Predictor (TPWP)" that is proposed for the case of keeping the original level of the front-end pipeline stages. The other is called "Phased cache with On-demand Wakeup Prediction, (POWP) policy" that is proposed for the case of allowing new stage is inserted into original front-end pipeline. The research results show that: compared with traditional power controlling policies, proposed policies have the better power efficiency. They can reduce the power of whole processor more efficiently with less performance degradation.

Key words: instruction; cache; power; architecture

1 引言

由于较高的访问频率,指令 cache 是微处理器动态功耗的重要来源之一,降低指令 cache 的动态功耗一直是降低指令 cache 功耗的主要目标.随着工艺尺寸的缩小,特别是当工艺尺寸降低到 70nm 以下,指令 cache 中的漏流功耗逐渐成为指令 cache 总功耗的主要组成部分.根据半导体工业协会的预测,当工艺尺寸降低到 70nm,处理器中静态功耗将与动态功耗相当[1].对于拥有大量晶体管的静态存储器,静态功耗越来越显著[2].静态功耗主要来源于芯片中晶体管亚阈泄漏电流[3]所产生的功耗,即亚阈漏流功耗,因此降低静态功耗以降低亚阈漏流功耗为主.通过使用 Wattch[4]功耗模拟器对 70nm 工艺条件下主频为 2GHz 的微处理器中指令 cache 进行功耗模拟可以发现:指令 cache 每周期所消耗的漏流能量与每次被访问时所消耗的动态能量相当.在超深亚微米工艺条件下,降低指令 cache 的功耗需要考虑同时降低它的静态漏流功耗和动态访问功耗.由于指令 cache 位于处理器流水线的前端,负责向流水线提供指令,访问指令 cache 的停顿会直接影响流水线的吞吐率,造成整个处理器性能下降.因此,如何在降低指令 cache 功耗的同时尽可能地减少性能损失,实现更优的

收稿日期:2007-05-23;修回日期:2008-06-03
基金项目:国家自然科学基金(No. 60703074)

处理器能量效率,是低功耗指令 cache 设计的关键.

为了减少性能损失,目前对于指令 cache 中的存储单元所使用的低漏流电路主要是昏睡 cache[5]中所采用的状态保留的低漏流 SRAM 电路.处于低漏流昏睡模式下的休眠 cache 块中存储的数据仍可以保持,额外的访问延时主要来自于低漏流模式与正常活跃模式之间的状态转换延时,即"唤醒延时".目前所提出的各种面向指令 cache 的漏流功耗优化策略都是在降低指令 cache 的漏流功耗与减少由唤醒延时所造成的性能损失之间寻找平衡点,以达到最佳的处理器能量效率.对于昏睡指令 cache 的体系结构级漏流功耗优化,休眠策略主要采用未访问(noaccess)策略[6].未访问策略以 cache 块为粒度进行低功耗控制,如果一个 cache 块被访问后经过一定的时间(衰退间隔)未被再次访问,则将它置为低功耗模式.LRU-Assist 策略[7]也是一种休眠策略,主要用于数据 cache.唤醒策略主要有唤醒顺序组(set)策略[8]、唤醒顺序块策略[9,11]、按需唤醒策略[10].在当前 cache 组被访问的同时,唤醒顺序组策略推断地唤醒顺序后继组中所有的 cache 块.唤醒顺序块策略则仅推断唤醒顺序后继组中可能被命中的那一个 cache 块,路预测器被用于预测哪一路可能被命中.唤醒顺序组策略与唤醒顺序块策略利用了访问 cache 块的地址具有较好顺序性的特征,其缺点是如果指令流发生分支跳转,则推断唤醒失败.按需唤醒块策略将分支预测器也作为唤醒预测器,唤醒分支目标方向的 cache 块,进一步提高了推断唤醒的精度.唤醒策略同时影响 cache 的功耗和性能,因此目前基于昏睡 cache 的功耗优化主要集中于唤醒策略的优化.张承义等提出的 PDSR(Periodically Drowsy Speculatively Recover)策略[8]采用了唤醒顺序组的唤醒策略.S W Chung 等通过改进前端流水线结构[10]实现了一种按需唤醒策略.N S Kim 等提出的 noaccess-JITA 策略[11]采用唤醒顺序块的唤醒策略.

为了降低多路组相联 cache 的动态功耗,目前所提出的体系结构级方法有分阶段访问 cache(Phased cache)[12]和路预测 cache(Way Predicting cache)[13],通过减少不必要的 cache 块访问降低 cache 动态访问功耗.在传统 cache 中,标识阵列(tag array)和数据阵列(data array)被同时访问.在 Phased cache 中,标识体与数据体分别在不同的流水线站访问.首先访问标识体,获得路命中信息,然后在下一站仅访问命中路 cache 块的数据单元(data block),其他路的数据单元不需要访问.在路预测 cache 中,路预测表用于记录每一个 cache 组的路命中情况,作为该组再次被访问时的路预测信息.对于每一次 cache 访问,首先根据路预测信息,推断地访问位于预测路的 cache 块,如果命中则不需要被访问其他路的 cache 块.路预测命中率影响 cache 的性能.

先前的体系结构级指令 cache 功耗降低方法较少将降低静态功耗和动态功耗结合考虑.为了最大程度地降低指令 cache 总功耗,静态功耗和动态功耗需要被同时优化.由于指令 cache 访问延时与整个处理器的性能有密切联系,因此在降低指令 cache 功耗的同时,需要尽量避免性能损失.本文从静态功耗优化与动态功耗优化相结合、降低 cache 功耗与减少处理器性能损失相平衡的角度,研究指令 cache 的功耗优化方法,提出两种改进的功耗控制策略:"使用双预测端口路预测器(Two Prediction-ports Way Predictor,简称 TPWP)的多路路预测(Multi-Way Way Prediction,简称 MWWP)策略"和"基于分阶段访问 cache 的按需唤醒预测(Phased cache with On-demand Wakeup Prediction,简称 POWP)策略",分别用于处理器流水线级数保持不变和可以增加额外流水线级数两种情形.

2 使用 TPWP 的 MWWP 策略

为了在降低指令 cache 漏流功耗的同时,降低指令 cache 的动态功耗,我们改进传统的只有一个端口用于预访问路预测的单预测端口路预测器[13]为双预测端口路预测器.在双预测端口路预测器中有两个路预测端口:一个是"预访问(pre-access)"端口,该端口与传统的单预测端口路预测器的读访问端口功能相同,用于对当前被访问组的一个或者多个路中的 cache 块进行推断

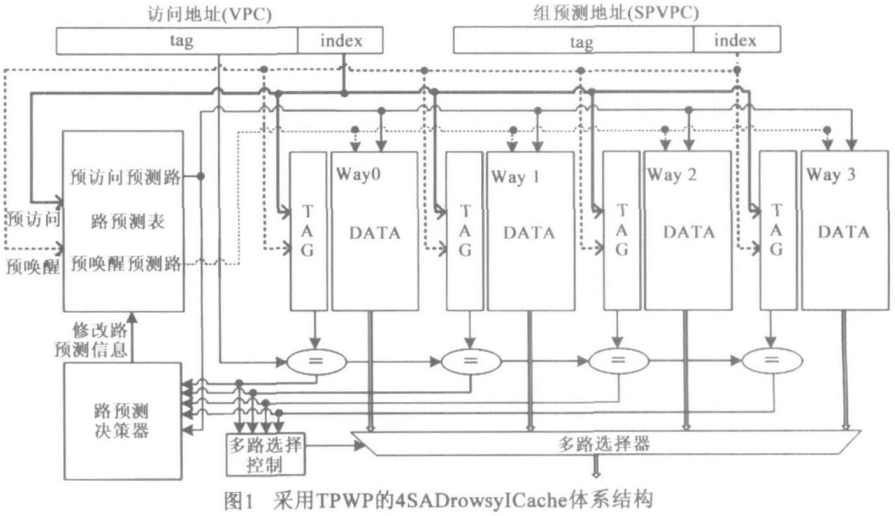

图1 采用TPWP的4SADrowsyICache体系结构

地访问;另一个是"预唤醒(pre-wakeup)"端口,该端口用于对当前被访问组的顺序后继组中的一个或者多个路中的 cache 块进行推断地唤醒.

采用 TPWP 后的 4 路组相联 Drowsy 指令 cache(4-way SetAssociation Drwosy I-Cache,4SADrowsyICache)的体系结构如图 1 所示.路预测表同时被两条访问通路访问,实线所表示的通路为预访问通路,虚线所表示的通路为预唤醒通路,访问地址分别为虚地址程序计数器(Virtual Program Counter,简称 VPC)与组预测的 VPC(Set Predicted VPC,简称 SPVPC).路预测表中保存了路预测信息(路预测标识),被预访问操作与预唤醒操作同时访问.路预测决策器根据每次访问时的 cache 命中情况修改路预测表中的路预测信息.由于预唤醒操作仅仅通过电压调整控制 cache 块的功耗状态转换,因此不会产生对路预测表中的路预测标识的修改.路预测表有两个读端口:一个读端口用于预唤醒操作的访问,即预唤醒端口,由预唤醒端口读出的路预测信息被称为预唤醒预测路(Pre-Wakeup Prediction Way,简称 PW-Pred-Way);另一个读端口用于预访问操作的访问,即预访问端口,由预访问端口读出的路预测信息被称为预访问预测路(Pre-Access Prediction Way,简称 PA-Pred-Way).由于在传统的处理器前端流水线设计中,指令 cache 与分支预测器被同时访问,因此很难在不调整前端流水线级数的前提下,在当前访问地址访问指令 cache 之前得到对该访问地址的分支预测目标地址.因此,在使用 TPWP 的 4SADrowsyICache 中,仅简单地预测当前被访问组的顺序后继组为下一个要访问的组,即 SPVPC 为 VPC 的顺序后继地址.

图 2 为使用 TPWP 的 4SADrowsyICache 的访问过程,假设唤醒延时与访问延时均为 1 个时钟周期.使用 TPWP 后,指令 cache 的访问延迟与预唤醒路预测和预访问路预测是否命中密切相关.由于预唤醒和预访问所使用的路预测信息相同,根据对访问过程的分析,可以得出结论:在 cache 命中的前提下,如果在 Cycle0 中指令地址未发生分支跳转,则在 Cycle0 中被预唤醒的 cache 块必然会在 Cycle1 中被预访问,因此 Cycle0 中的预唤醒与 Cycle1 中的预访问同时命中与失效,若命中则经过 ①②③⑤步后获得所需数据,否则根据匹配路中数据单元是否为活跃态,继续经过 ⑥⑦或 ⑥⑧⑨才能得到所需数据.如果在 Cycle0 的指令地址发生了分支跳转,则预唤醒必然失效(唤醒了错误分支方向上的 cache 块),根据预访问预测是否命中和被预访问的 cache 块是否处于昏睡状态,同样会通过前面 3 种不同情况最终获得所需数据.cache 命中最坏情况下需要重新唤醒然后访问命中的 cache 块,额外增加 2 拍访问延时,影响处理器性能.因此,在使用 TPWP 的 4SADrowsyICache 中提高路预测命中率是一个比在传统的路预测组相联 cache 中提高路预测命中率更为突出的问题.

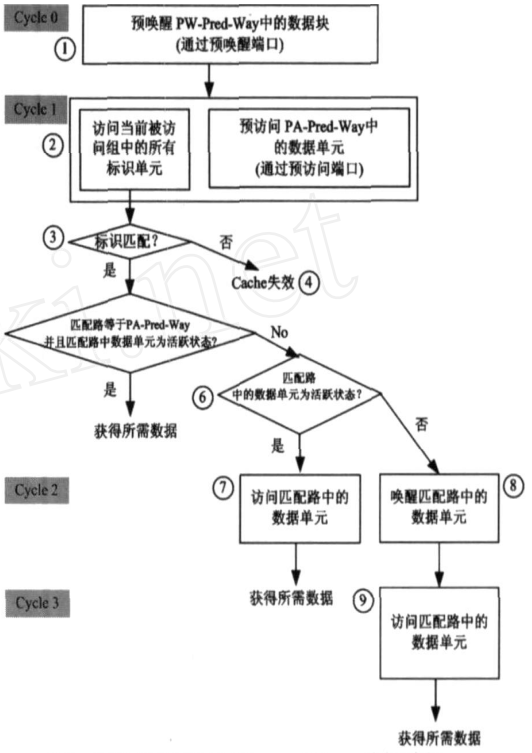

图2　采用TPWP的4SADrowsyICache访问过程

根据程序访存的时间局部性,在多路组相联 cache 中,当最近最多被访问(Most Recently Used,简称 MRU)路失效时,仅次于 MRU 路的最近次多被访问(Second Recently Used,简称 SRU)路被访问的概率最大.为了提高路预测命中率,我们改进传统的单路路预测策略,增加每次访问时被预测路的数目,称为多路路预测策略.例如对于 4 路组相联指令 cache,路预测器对每一次访问可以同时预测两个最有可能被访问的路(即 MRU 路和 SRU 路),我们称该策略为双路路预测(2-Way Way Prediction,2WWP)策略.以此类推,N 路路预测策略将推断地访问最近可能被访问的 N 个路.与单路路预测 4 路组相联 cache(1WWP-4SACache)相比,2WWP-4SACache 中的路预测标识及其选择逻辑、访问控制器的数目增加了一倍.对于三路路预测 4 路组相联 cache(3WWP-4SACache)的实现,可以仅在路预测表中保存用于指示 LRU 路的路预测信息,每次访问时,除被预测的路外推断地访问剩余的路.因此 3WWP-4SACache 中的路预测表容量与 1WWP-4SACache 中的路预测表容量相同,仅需要增加更多的访问控制逻辑.对于 N 路组相联 cache 来说,实现 1 路到 N-1 路路预测机制时,为每一个 cache 块增加的路预测标识最多为 $(N/2)\log_2 N$ 位.

3　POWP 策略

前面所提出的策略适用于保持原有流水线级数不

变的情况,POWP 策略主要针对可以增加额外前端取指流水线站的情况.在传统的高性能多路组相连指令 cache 中,为了提高取指速度,在一拍内获得指令,tag 阵列和 data 阵列被同时访问,而不像数据 cache 中那样被分阶段地访问.我们将 tag 和 data 阵列被同时访问的 cache 称为非分阶段访问 cache (Non-Phased cache).Sung Woo Chung 等提出的按需唤醒预测策略基于这种 cache 体系结构[10],我们称为基于非分阶段访问 cache 的按需唤醒预测(Non-Phased cache with On-demand Wakeup Prediction,简称 NPOWP)策略.在使用 NPOWP 策略的前端取指流水线[10]中,取指站之前额外增加一级唤醒站,专门用于对即将访问 cache 块进行按需唤醒.分支预测信息除被用于分支预测外,同时用于对 cache 块的唤醒预测,以提高唤醒预测的命中率.除了被唤醒和被访问 cache 块的 tag 和 data 单元外,其他 cache 块的 tag 和 data 单元均处于昏睡状态.路预测器用于对被唤醒的 cache 块所在路进行预测.NPOWP 策略对处理器性能的影响主要由路预测失效和分支预测失效引起.如果路预测命中,则唤醒延迟可以通过流水线的重叠得到隐藏,不会在流水线中产生气泡.如果路预测失效,则在前端流水线的执行过程中会产生 2 个气泡,增加 2 个时钟周期的延时:一个气泡的产生是由于需要唤醒当前被访问组中除被预测的 cache 块外剩余的 cache 块;另一个气泡的产生是由于需要访问这些剩余的 cache 块.当发生分支预测失效时,额外增加的流水线站会增加重启前端流水线的开销,导致 1 个时钟周期的额外唤醒延迟.因此,虽然 NPOWP 可以在更大程度上提高昏睡 cache 块的数目,但是由于程序执行时间的增长而产生的额外能量消耗会部分抵消 NPOWP 策略对整个处理器能量的节省.

为了能够在功耗和性能之间找到一个最佳的平衡点,我们改进 NPOWP 策略,结合分阶段访问 cache 体系结构,充分利用增加的唤醒站,提出了 POWP 策略.如图 3 所示,与 NPOWP 策略不同,在 POWP 策略中,对指令 cache 标识阵列的访问从取指站转移到唤醒站,对数据阵列的访问仍在取指站进行.标识阵列一直处于活跃状态,而且每次访问,被访问 cache 组中所有路的标识单元同时被访问.在唤醒站,以"组"为粒度进行唤醒,被访问组中所有路的 cache 块的数据单元被同时唤醒.如果标识比较的结果是命中,则在取指站只需访问命中路的 cache 块的数据单元就可以获得所需的数据.在 POWP 策略中不再使用路预测器,因为对指令 cache 当前被访问组中 cache 块数据单元的访问是按照真实的标识匹配信息进行.

POWP 策略与 NPOWP 策略一样,每次 cache 访问,在被访问组中只有命中的 cache 块的数据单元才会被访问,其他 cache 块的数据单元不需要被访问,因此节省了 cache 中对不必要的 cache 块数据单元的动态访问能量.但是,当使用 POWP 策略时,由于没有对标识阵列的漏流功耗进行控制,并且在每次访问时被访问组中所有 cache 块的数据单元被提前唤醒,而不像使用 NPOWP 策略时那样只有一个数据单元被提前唤醒,因此总的来说,使用 POWP 策略时对 cache 漏流功耗的优化效果不如使用 NPOWP 策略时的优化效果.POWP 策略的优势在于使用真实的标识比较结果作为访问 cache 块数据单元的路选信息,相当于实现了 100 %的路预测命中率,因此完全消除了传统的 NPOWP 策略中由于路预测失效所带来的流水线气泡,从而降低了对处理器性能的影响.

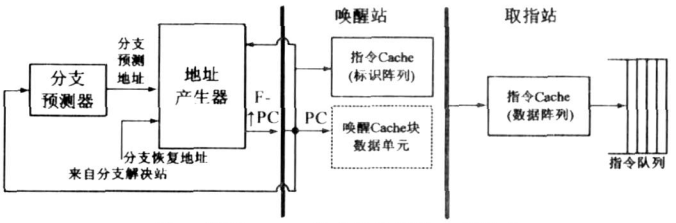

图3 使用POWP策略的前端流水线结构

4 实验及结果分析

我们使用 HotLeakage 功耗模拟器[14]对使用各种策略时指令 cache 和处理器的功耗进行模拟.HotLeakage 以 SimpleScalar3.0 作为模拟引擎,将 HotLeakage 漏流功耗模型与 Wattch 动态功耗模型集成在了同一个工具包中,并对其工艺参数按照 70nm 工艺进行了扩展,因此可以方便地评估漏流功耗控制策略对漏流功耗以及动态功耗的影响,对功耗优化方法进行更合理的综合评价.由于使用功耗控制策略会额外引入部分控制逻辑和寄存器,因此会带来额外的动态功耗和静态功耗,我们在模拟器中对这些硬件的功耗进行了建模:寄存器的漏流功耗和动态功耗按照 HotLeakage 功耗模型中 SRAM 的功耗计算方法估算,组合逻辑的功耗利用现成的数据按比例伸缩.模拟过程中处理器参数模型的选取参考高性能微处理器 Alpha 21264 的参数,一级指令 cache 容量为 64KB,4 路组相联,块大小为 32B,采用昏睡 cache 体系结构,唤醒延迟为 1 个时钟周期.功耗与能量参数基于 70nm/0.9V 工艺.

我们在 I-Cache 中除设计实现了提出的使用 TPWP 的 2WWP 策略(简称 2WWP with TPWP)和 POWP 策略,还实现了传统的不使用唤醒预测的 noaccess 策略[6],PDSR 策略[8],noaccess-JITA 策略[11]和 NPOWP 策略[10],用于和提出的策略进行比较.其中除 NPOWP 和 POWP 策略外,其他策略中的休眠策略均使用未访问策略,衰

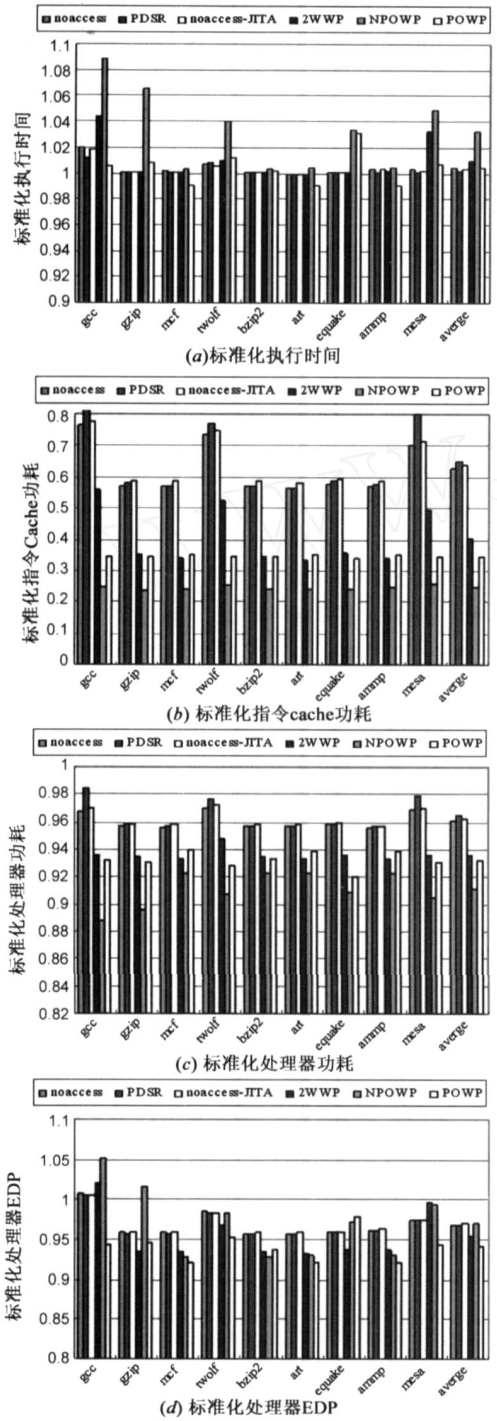

图4　使用各策略时的标准化执行时间、指令cache功耗、处理器功耗和处理器EDP

退间隔设置为 32 K 时钟周期,I-Cache 中标识阵列同数据阵列一起被休眠或唤醒. 为了便于比较,我们将 HotLeakage 的模拟结果代入文[15]中提出的 cache 和处理器功耗和性能估算模型,计算出标准化执行时间(Normalized Execution time)、标准化指令 cache 功耗(Normalized Power of I-Cache)、标准化处理器功耗(Normalized Power of Processor)和标准化能量延迟积(Normalized Energy Delay Product),不仅评价各策略对指令 cache 功耗的降低效果,还评价其对整个处理器的功耗、性能和能量效率的影响. 在功耗估算模型[15]中,将不采用任何 cache 功耗优化策略的处理器模型作为基准模型,标准化 cache(或处理器)功耗定义为:对于给定程序,使用功耗优化策略后的 cache(或处理器)的功耗与基准模型中的 cache(或处理器)的功耗的比值. 标准化程序执行时间为使用功耗优化策略后的程序执行时间与基准模型的程序执行时间的比值. 标准化处理器 EDP 为使用优化策略后与基准模型中处理器 EDP 的比值. 使用标准化值进行比较的优点是可以直观地比较各策略的相对优劣,尽量减小体系结构级模拟器进行功耗估算时产生的绝对误差的影响.

图 4(*a*)为使用各种指令 cache 功耗优化策略时的标准化程序执行时间. 由于路预测失效开销,使用 2WWP with TPWP 策略后,处理器性能受到的影响比使用传统的 noaccess、PDSR、noaccess-JITA 策略时大. 使用传统的 NPOWP 策略需要增加额外的流水线站,当发生路预测失效或者分支预测失效时,流水线中的气泡或者重启流水线的开销导致程序执行性能显著下降,平均性能损失超过 3 %. POWP 策略能够有效改善 NPOWP 策略对性能的影响,而且避免了使用 2WWP with TPWP 策略时路预测失效所带来的性能损失,平均性能损失低于使用 NPOWP 策略和 2WWP with TPWP 策略时的结果,接近使用传统策略时的结果.

图 4(*b*)表示各策略对 I-Cache 功耗的优化能力. 在不改动流水线级数的设计中,提出的 2WWP with TPWP 策略优于传统的 noaccess、PDSR 和 noaccess-JITA 策略,平均可以降低更多的 I-Cache 功耗,主要原因是 2WWP with TPWP 策略可以同时对 I-Cache 的动态功耗与静态功耗进行优化,因此更显著地降低了 I-Cache 功耗. 在增加前端流水线级数的设计中,NPOWP 和 POWP 策略都可以对指令 cache 的动态与静态功耗同时优化,由于 NPOWP 能够对标识阵列进行功耗控制,而且通过路预测器仅唤醒预测路的 cache 块,因此仅从降低 I-Cache 功耗考虑,NPOWP 策略最优. 图 4(*c*)为使用各策略时的标准化处理器功耗,用于评价各指令 cache 功耗控制策略最终对处理器功耗的影响. 在不考虑性能损失的前提下,NPOWP 策略最优,达到最低的标准化处理器功耗. 我们提出的 2WWP with TPWP 策略和 POWP 策略优于除 NPOWP 外的所有其他策略. 虽然 NPOWP 策略可以最大程度地降低 I-Cache 功耗并使处理器功耗最低,但是以显著的性能损失为代价. 我们提出的两种策略能够在功耗与性能之间实现更优的平衡.

标准化处理器 EDP 用于综合评价各策略对处理器功耗和性能的影响. 如图 4(*d*)所示,使用提出的 2WWP

with TPWP 策略和 POWP 策略时平均标准化处理器 EDP 均优于使用其他策略时的结果. 使用 NPOWP 策略时由于显著的性能损失造成其标准化处理器 EDP 的改善并不明显,比使用提出的两种策略时的效果差,与使用其他传统策略时的结果相当. 可见,提出的两种策略可以有效地权衡处理器的功耗与性能,达到更优的能量效率. 相对而言,2WWP with TPWP 策略的硬件实现比 POWP 策略的硬件实现简单,不需要对传统的流水线结构做大的改动. 而 POWP 需要增加额外的流水线站,增加额外的流水线控制与站间寄存器逻辑,但可以达到更高的能量效率.

5 小结

随着工艺尺寸缩小与处理器频率提高,功耗成为制约微处理器发展的重要因素. 本文提出了两种体系结构级的指令 cache 功耗控制策略:使用 TPWP 的 MWWP 策略和 POWP 策略. 前者为不改变处理器流水线级数情况下的功耗控制策略,后者则需要通过增加额外的流水线站来实现. 两种策略都可以同时对指令 cache 的动态访问功耗和静态漏流功耗进行有效控制. 实验表明:在 4 路组相联昏睡指令 cache 中,使用提出的两种功耗控制策略比可以在不显著影响处理器性能的前提下,有效降低指令 cache 的静态功耗和动态功耗,进一步改善处理器的能量效率.

参考文献:

[1] ITRS organization, International Technology Roadmap for Semiconductors 2002 Update [DB/OL]. http://public.itrs.net/, 2002-03-18.

[2] H Hanson, S W Keckler, et al. Tech Report TR2001-18 [R]. Texas, USA: The University of Texas at Austin, 2001.

[3] K Roy, et al. Leakage current in deep-submicron CMOS circuits [J]. Journal of Circuits, Systems, and Computers, 2002, 11(6): 575 - 600.

[4] D Brooks, V Tiwari, et al. Wattch: A framework for architectural-level power analysis and optimization [A]. SIGARCH. Proceedings of the 27th Annual International Symposium on Computer Architecture [C]. Washington: IEEE Computer Society, 2000. 83 - 94.

[5] K Flautner, N S Kim, et al. Drowsy caches: simple techniques for reducing leakage power [A]. SIGARCH. Proceedings of the 29th annual international symposium on Computer architecture [C]. Washington: IEEE Computer Society, 2002. 148 - 157.

[6] N S Kim, K Flautner, et al. Circuit and microarchitectural techniques for reducing cache leakage Power [J]. IEEE Transaction on VLSI Systems, 2004, 12(2): 167 - 184.

[7] 张承义,张民选,等. LRU-Assist:一种高效的 cache 漏流功耗控制算法[J]. 电子学报, 2006, 34(9): 1626 - 1630.
ZHANG Cheng-yi, ZHANG Min-xuan, et al. LRU-Assist: An efficient algorithm for cache leakage power controlling [J]. Acta Electronica Sinica, 2006, 34(9): 1626 - 1630. (in Chinese)

[8] C Zhang, H W Zhou, et al. Architectural leakage power reduction method for instruction cache in ultra deep submicron microprocessors [A]. The 11th Asia-Pacific Computer Systems Architecture Conference [C]. Berlin, Heidelberg: Springer-Verlag, 2006. 588 - 594.

[9] J Hu, et al. Exploiting program hotspots and code sequentiality for instruction cache leakage management [A]. International Symposium on Low Power Electronics and Design (ISLPED'03) [C]. Berlin, Heidelberg: Springer-Verlag, 2003. 25 - 27.

[10] S W Chung, K Skadron. Using branch prediction information for near-optimal I-Cache leakage [A]. The 11th Asia-Pacific Computer Systems Architecture Conference [C]. Berlin, Heidelberg: Springer-Verlag, 2006. 24 - 37.

[11] N S Kim, K Flautner, et al. Single-VDD and single-VT super-drowsy techniques for low-leakage high-performance instruction caches [A]. International symposium on low power electronics and design (ISLPED'04) [C]. Berlin, Heidelberg: Springer-Verlag, 2004. 54 - 57.

[12] A Hasegawa, et al. SH3: High code density, low power [J]. IEEE Micro, 1995, 15(6): 11 - 19.

[13] K Inoue, T Ishihara, et al. Way-predicting set-associative cache for high performance and low energy consumption [A]. Proceedings of 1999 International Symposium on low power Electronics and Design (ISLPED'99) [C]. Berlin, Heidelberg: Springer-Verlag, 1999. 273 - 275.

[14] Y Zhang, D Parikh, et al. Hotleakage: An Architectural, Temperature-aware Model of Subthreshold and Gate Leakage [R]. Virginia, USA: Department of Computer Sciences, University of Virginia, 2003.

[15] Zhou Hongwei, Zhang Chengyi, et al. Improved way prediction policy for low-energy instruction cache [A]. International Conference on Embedded software and system [C]. Berlin, Heidelberg: Springer-Verlag, 2007. 425 - 436.

作者简介:

周宏伟 男,1980 年生于陕西宝鸡. 国防科技大学计算机学院博士,天津航海仪器研究所博士后,主要研究方向为超深亚微米微处理器体系结构级功耗优化技术、高性能微处理器存储系统设计等.
E-mail: hongw.zhou@gmail.com

张民选 男,1954 年生于湖南邵东. 国防科技大学计算机学院教授、博士生导师,主要研究方向为计算机体系结构与实现技术、大规模集成电路设计等.