一种面向超标量处理器的高能效指令缓存路选择技术

谢子超,陆俊林,佟 冬,王箫音,程 旭
(北京大学微处理器研究开发中心,北京 100871)

摘 要: 路选择技术可以有效降低指令缓存能耗开销,但已有方法通常会由于预测错误或更新机制复杂而引入额外的取指延迟,导致整体能效性降低.本文面向典型超标量处理器的指令缓存结构,提出了一种高能效的路选择融合技术(Combining Way Selective Cache,CWS-Cache).基于对路预测和路历史技术适用条件的分析,CWS-Cache 在不同的取指场景中选择使用最佳路选择策略,有效降低了指令缓存的取指能耗,并通过缩短非对齐取指组的访问延迟提升处理器性能.实验表明,CWS-Cache 将拥有 8 路组相联指令缓存的基础处理器取指能耗降低了 84.98%,性能提升了 3.50%.与已有的三种方法相比,CWS-Cache 能效性分别提升了 15.48%,14.13%和 8.76%.

关键词: 超标量处理器; 路预测; 路历史
中图分类号: TP302 **文献标识码:** A **文章编号:** 0372-2112 (2011) 11-2473-07

An Energy-Efficient Combining Way Selective Technique for the Instruction Cache in Superscalar Microprocessors

XIE Zi-chao, LU Jun-lin, TONG Dong, WANG Xiao-yin, CHENG Xu
(*Microprocessor Research and Development Center, Peking University, Beijing* 100871, *China*)

Abstract: Way selective technique could reduce the instruction cache energy consumption significantly. However, existing solutions usually bring extra fetch latency due to mispredictions or complicated updating mechanism, reducing the energy-efficiency. The paper presents an energy-efficient Combining Way Selective Cache for the instruction cache in superscalar processors (CWS-Cache). It combines the advantages of way prediction and way history techniques, and selects the best way selective mechanism for different situations. It not only reduces the instruction fetch energy effectively, but also improves performance by reducing the latency of misalignment fetch groups. Experimental results demonstrate that, on average, CWS-Cache reduces fetch energy consumption of the 8-way set-associative instruction cache in the baseline processor by 84.98%, and improves performance by 3.5%. Compared with three existing techniques, CWS-Cache improves the energy-delay product (EDP) by 15.48%, 14.13%, and 8.76%, respectively.

Key words: superscalar microprocessors; way prediction; way history

1 引言

当前处理器设计不再以性能或能耗为唯一驱动,而是综合考虑两种设计要素,以高能效为主要设计目标[1].为充分发掘指令级并行度,超标量处理器对指令缓存(Instruction Cache,ICache)的取指带宽和取指延迟均有着较高的要求,其需要每周期向流水线不间断地提供多条指令.而指令缓存的组相联结构往往会造成大量非命中路的能耗损失.因此,如何优化超标量处理器的指令缓存结构,在有效降低取指能耗的同时,缩短取指延迟,获得更高的能效性,是微处理器设计人员重点关注的问题.

路选择技术根据指令缓存的取指历史只对其一路进行访问,其主要包括路预测技术与路历史技术两类.然而,典型路预测方法[2]在降低 ICache 访问能耗开销的同时会由于预测错误带来明显的性能损失,而路历史方法[3~6]均只对单发射按序执行处理器进行了探讨和实验,未能充分考虑超标量处理器中路历史信息更新所造成的延迟.

本文面向典型超标量处理器的指令缓存结构,提

收稿日期:2011-01-03;修回日期:2011-04-01
基金项目:国家"核高基"重大专项:安全适用计算机 CPU 研发与应用项目(No.2009ZX01029-001-002)

出了一种高能效的路选择融合技术(Combining Way Selective Cache,CWS-Cache).基于对各种路选择技术适用场景的分析,CWS-Cache 在不同的取指场景中使用最优的路选择策略,在显著降低能耗的同时,通过缩短非对齐取指组的取指延迟来提高处理器性能.CWS-Cache 在性能和能耗两方面均进行了优化,因而可以达到高能效的设计目标.

本文实验环境基于超标量处理器的典型结构,使用 SPEC CPU 2000 基准程序集[7]进行评测.实验结果表明,CWS-Cache 将基础处理器的取指能耗降低了 84.25%,而性能提升了 3.5%.与已有的 SAWP[2],Way Memorization[3] 以及 WHOLE-Cache 方法[4] 相比,CWS-Cache 将处理器的能效性分别提高了 15.48%,14.13% 和 8.76%

2 路选择技术

2.1 基本原理

一个 Cache 行除非被替换(eviction),否则始终停留在 Cache 的同一路中.路预测技术依据该特性,使用额外硬件(即路预测位)保存以往取指的路命中结果.现有路预测技术[2]针对顺序行间取指情况(顺序取指,且当前指令与下一指令在同一 Cache 行中)和非顺序取指情况,分别在每个 Cache 行和分支目标缓冲器(Branch Target Buffer,BTB)的每一项中加入预测位,如图 1 所示.顺序行间取指时,当前 Cache 行的预测位负责预测下一条指令所在的路.非顺序取指时,当前 BTB 项的预测位负责预测分支目标地址所在的路.当路预测正确时,只读取并比较该路的 Tag 以验证预测的正确性.此时能耗开销为单路的 Tag 和数据的访问.如果发现预测错误,则舍弃本周期取得的指令,在下一周期执行传统的 I-Cache 访问,并更新之前使用的预测位,此时会造成一个周期的取指延迟和一路访问能耗损失.

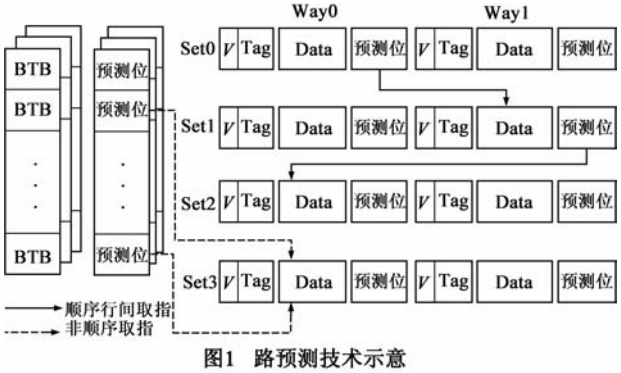

图1 路预测技术示意

路历史技术是对路预测技术的拓展.对 Cache 行为和结构的分析表明[3],通常情况下,只有当 ICache 进行行替换时才会破坏已有的路预测关联,导致预测错误.路历史技术利用此特性,对每个路预测位增加有效位,标识该预测是否正确.该有效位与预测位合称为路指针.在 ICache 出现行替换时,需将所有在 ICache 和 BTB 中指向替换行的路指针置为无效,避免使用这些路指针引起取指错误.当路指针无效时,使用传统的 ICache 访问机制进行访问,更新路指针并设置有效位.当路指针有效时,直接访问数据,无需 Tag 读出与比较.

基于以上介绍,两种路选择技术取指延迟和能耗开销的不同,主要是由于其各自的更新机制不同:路预测技术只有当预测错误时才对使用的预测位进行更新,而路历史技术不仅需要更新遇到的无效路指针,还需要在出现 ICache 行替换时,将所有指向该 Cache 行的路指针置无效.

2.2 相关工作

Calder[8]等人提出路预测技术的最初目的是使组相联 Cache 取得类似直接映射 Cache 的访问命中时间,从而缩短处理器取指延迟.之后,Powell 等将其应用到低能耗领域,提出了 SAWP 结构[2].SAWP 结构针对 ICache 和 BTB 分别建立路预测表,记录指令的取指历史.另一方面,周宏伟等将路预测技术与静态电流控制相结合,提出了 TPWP 和 MWWP[9]结构,有效降低动态访问功耗和静态漏流功耗.

Ma 等人提出了 Way Memorization 的路历史方法[3],将记录顺序执行和分支转移的路指针均保存在 ICache 中.该方法的缺点是每条指令都拥有一个路指针,因而需要大量额外的存储资源.为了节约额外的硬件开销,Ishihara 等人提出使用小容量的 MAB 结构[5]记录路历史信息,而 Inoue 等人提出的 HBTC-Cache 结构[10]将路历史信息记录在分支目标缓冲器中.综合以上结构,WHOLE-Cache[4]提出了一种分离式的路指针存储方式,对于不同的指令执行情况使用不同的部件提供路指针,以降低 ICache 的访问能耗.与其他结构融合方面,Hines 等在 Filter Cache 结构的基础上,借用部分路历史思想,提出了 TH-IC 结构[6],避免访问大容量的 L1-I-Cache 的能耗损失.然而,以上路历史方法均只针对单发射处理器进行了探讨和实验,并都假设在 ICache 出现行替换时,路指针的更新操作可以被忽略.

2.3 取指场合适用性分析

顺序行间取指时,两种路选择技术在取指延迟上存在差异.路预测会因预测错误增大取指延迟.而路历史避免了预测错误情况.当出现 ICache 行替换需要进行更新时,路历史直接在索引连续的上一个 Set 中,通过 Tag 比较找到指向被替换 Cache 行的路指针,并将其置无效.该过程易于实现且可以隐藏在 Cache 行替换过程中,因而不会增大取指延迟.

对于超标量处理器中特有的非对齐取指情况(特

殊的顺序行间取指情况),已有的路选择方法均未能很好的予以解决.当一个取指组的指令分布在两个地址连续的 Cache 行中时,会出现取指组非对齐情况(Fetch Group Misalignment),此时处理器需要 2 次取指将全部指令取回.该问题是超标量处理器中 ICache 获得最大取指带宽的重要障碍[11].本文面向典型超标量处理器结构(配置见表 1),针对 SPEC CPU 2000 测试集[7]统计非对齐取指比例.统计结果如图 2 所示,处理器中平均接近 30%* 的取指会出现取指组非对齐情况.现有路预测方法可能由于预测错误,导致后一部分指令无法读出.而已有路历史方法只有当取到 Cache 行最后一条指令时,才会读取路指针,对下一次取指所在的路进行预测,因此无法同时读取两部分指令.

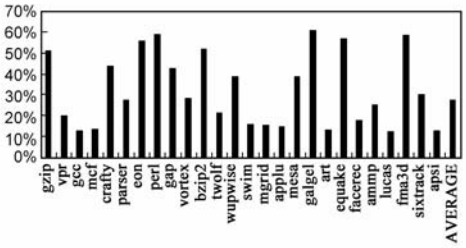

图2　本文实验环境下处理器非对齐取指占全部取指比例

在非顺序取指时,路历史技术的更新机制更为复杂,会导致取指暂停以及丢失有用路指针的问题.使用路历史技术时,BTB 中会有多个路指针指向同一个 Cache 行,如图 1 中虚线所示.当 ICache 出现行替换时,需要立即将所有指向被替换 Cache 行的指针置无效.根据已有研究[3,4]的分析,由于可能出现 BTB 中大量路指针指向同一个 Cache 行的情况,因此必须实现 BTB 中路指针整体置无效操作.由于超标量处理器的分支转移预测部件一般较大,例如,Intel Pentium M 中 BTB 共 4K 项[13],如果要将其中所有路指针全部置无效,按文献[4]中 BTB 中路指针的组织结构,一个周期最多无效 32 个路指针,全部置无效需要 128 个周期.在整体置无效完成前,分支指令不能进行预测,否则会出现本该被置无效的路指针被错误使用的情况.分支预测停顿会引起取指暂停,影响流水线中指令的发射效率.本文参考 SPEC 公布值[12],统计测试集中各程序运行时 ICache 的替换率,并选取替换率大于 0.5% 的程序,评测取指停顿带来的性能损失.结果如图 3 所示,该情景会带来较为明显的性能损失.另外,整体置无效操作也会将无关的有效路指针置无效,降低路选择访问的比例.相比而言,路预测技术无需整体置无效操作.并且,本文通过实验评测发现,测试集中非顺序取指比例平均只有 12.96%.且已有研究[3,4]发现,当 Cache 行替换时,BTB 中路指针需要置无效的比例较小,这说明非顺序取指时预测错误的可能性较小,性能影响不大(具体评测见 4.2 节).

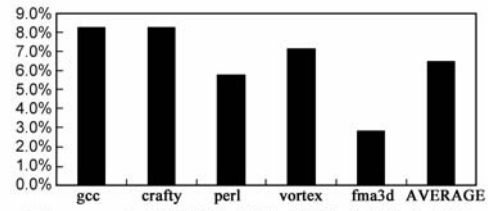

图3　BTB中路指针整体置无效造成的性能损失

综合以上分析,对于顺序行间取指情况,由于避免了错误预测带来的性能损失,路历史技术更为适合.而非顺序取指情况,由于避免了取指暂停和路指针丢失,路预测技术是更好的选择.

3　高能效的路选择融合技术(CWS-Cache)

本节面向典型超标量处理器中指令缓存,提出一种高能效的路选择融合技术(Combining Way Selective Cache,CWS-Cache).该技术充分发掘、利用不同路选择技术机制的特点,并将它们应用到不同的取指场景中,以达到更高的能效性.

3.1　CWS-Cache 机制

CWS-Cache 的整体结构,如图 4 所示.其中,灰色部分表示在传统 ICache 中所增加的部件.CWS-Cache 对顺序行间取指情况,采用路历史方式,避免错误预测,并缩短非对齐取指组的取指延迟.而对于分支转移引起的非顺序取指,采用路预测方式,有效避免 BTB 中整体更新操作.

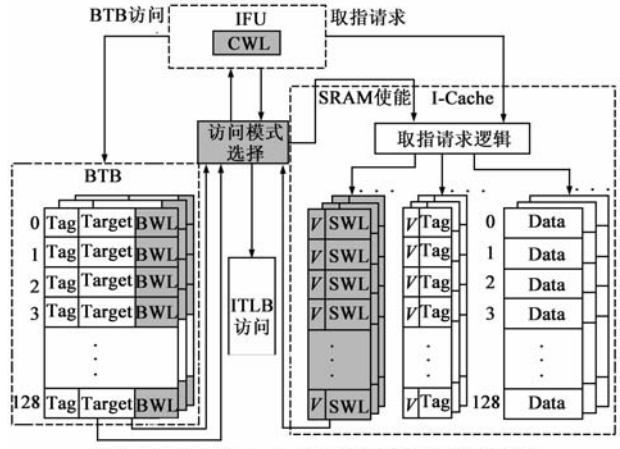

图4　采用CWS-Cache的超标量处理器取指结构

对于顺序取指且均在同一 Cache 行内的情况,CWS-Cache 在指令取指单元(Instruction Fetch Unit,IFU)中实现当前路指针(Current Way Linker,CWL),记录当前指令在 Cache 中所在的路.

对于顺序行间取指情况,CWS-Cache 对 ICache 中每

---

* 文献[11]中面向 SPEC Int 92 测试集进行了类似比例统计,评测数据与该数据近似,约为 23%.

一个 Cache 行增加一个顺序路指针(Sequential Way Linker,SWL).一个 SWL 根据 Tag 历史比较结果(即命中路数)指明该 Cache 行最后一条指令顺序执行时,下一条指令所在的路.一个 SWL 需要 LogN 位预测位(N 是 I-Cache 的组相联度)及 1 位有效位.如果 SWL 有效,则下一指令在 ICache 中的位置就是唯一确定的,CWS-Cache 直接访问所求数据,无需所有 Tag 以及其他路 Data 的访问(路历史访问模式).否则,下一条指令采用传统的 Cache 访问方式.

对于特殊的行间取指情况 – 非对齐取指,CWS-Cache 使用提前判断和检查 SRAM 是否相同的方式予以解决.首先,需要提前判断并准备取指组非对齐情况.CWS-Cache 每次取指时,首先根据 PC 中低位,判断该取指组是否对齐,并在每次进入一个新 Cache 行取指时,提前读取并保存其 SWL.其次,当判断出非对齐取指组保存在不同 SRAM 中时,同时获取两部分指令.本文实验环境下,对 SPEC 测试集[7]进行统计,发现非对齐取指组的两部分指令保存在相同 SRAM(即相同路)中的比例平均仅有 3%,如图 5 所示.当指令保存的 SRAM 不相同时,这两个 SRAM 可同时激活并取得其中保存的指令.CWS-Cache 将本 Cache 行的 SWL 与 CWL 进行比较,以判断 SRAM 是否相同.如果 CWL 与 SWL 均有效且值不同,即 SRAM 不相同.否则,均假设两个 SRAM 相同.SRAM 相同时,CWS-Cache 只将在当前 Cache 行的指令取回.CWS-Cache 使用以上方法的优点在于只利用现有部件,而不对 Cache 组织结构进行修改,如更改数据排布方式或增加读口.

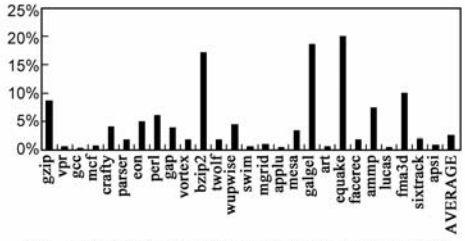

图5 非对齐取指组所在2个SRAM相同的比例

对于非顺序取指情况,CWS-Cache 对 BTB 中的每一项扩展一个 LogN 位的分支预测位(Branch Way Predictor,BWP).在 BTB 建立新项时,预测位需要一同建立.在处理器取指过程中,如果 BTB 访问命中,则同时得到预测的分支目标地址和其预测的 ICache 路.CWS-Cache 此时直接访问预测路,同时进行检查(路预测访问模式).如果预测正确,则继续下一组取指操作.如果预测错误,则暂停下一组指令的取指,下一周期执行传统 I-Cache 访问,选出正确的路,并更新 BTB 中 BWP.

3.2 实现结构

在传统的 ICache 结构上使用 CWS-Cache 技术对原有结构的改动不大,如图 6 所示:

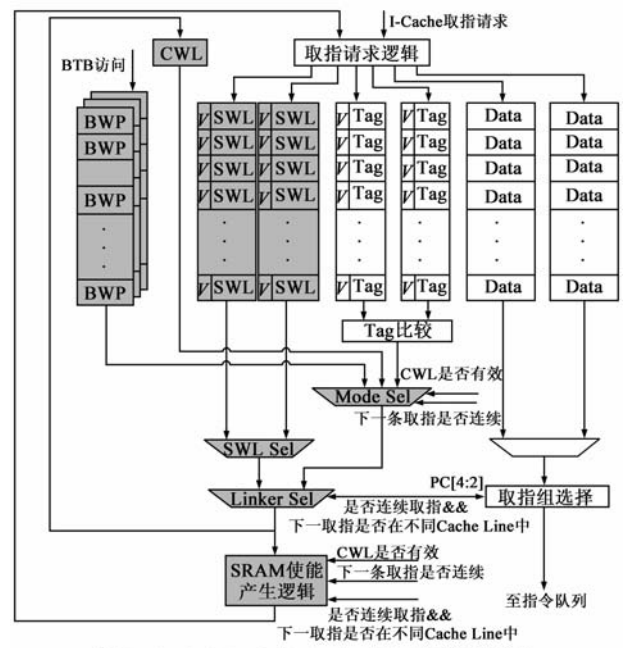

图6 CWS-CCache在ICache中的具体实现结构

(1)在取指单元中加入 CWL 寄存器.

(2)ICache 和 BTB 中分别加入 SWL 和 BWP 阵列.

(3)加入 Mode Sel 多选器,其根据 CWL 是否有效以及下一取指是否连续,选择使用的访问模式.

(4)加入 SWL Sel 和 Linker Sel 多选器,在路历史访问模式下选择使用 SWL 或 CWL.

(5)加入 SRAM 使能产生逻辑.针对不同访问模式,产生各个 Tag RAM, Data RAM 以及 SWL RAM 的使能信号.当判断可以进行非对齐取指时,同时激活 2 个 Data RAM.

以上对处理器取指逻辑的修改,不会对时序造成明显的影响.SWL Sel,Linker Sel 多选器可以与原有组相联 Data 选择逻辑并行执行.Mode Sel 与 SRAM 使能产生逻辑的加入,会使 ICache 取指延迟增大.正在设计中的 UniCore-3 超标量处理器中采用了 CWS-Cache 技术.该处理器目标频率为 1.2GHz,拥有 32KB,8 路组相联 ICache.TSMC 65nm 工艺下时序评估表明,加入 CWS-Cache 后的取指逻辑不是处理器的关键路径,不会对处理器频率产生影响.

4 实验评估

本文使用 SimpleScalar 模拟器[14]对处理器进行建模,使用 Wattch[15]功耗模型进行能耗评估.表 1 描述了用作比较的基础处理器结构.本文对 SPEC CPU 2000 基准程序集[7]中所有程序进行评测,运行 SimPoint[16]选取的 100M 条指令构成的有代表性的程序片段.

表 1　基础处理器配置参数

参数	配置值
流水线	1.2GHz,,8 级,4 发射乱序执行
取指单元	每周期取 4 条指令
分支转移预测器	BTB:4 路组相连,512 项 PHT:GShare 方法,2048 项
L1 ICache L1 DCache	32KB,8 路组相联,每行 32 字节 2 周期访问延时
L2 Unified Cache	512KB,16 路组相联 每行 64 字节,12 周期访问延时
指令 TLB 数据 TLB	8 表项全相联一级 TLB 64 表项 4 路组相联二级 TLB
主存	80 周期平均访问延时

本文选取相关方法中具有代表性的 SAWP[2]、Way Memorization[3]以及 WHOLE-Cache[4]方法在基础处理器模型上加以实现,作为本文的比较对象.表 2 列出了各种方法所需添加的额外硬件资源.表 3 列出了本文使用的能耗评估参数.

表 2　相关方法的结构描述及资源需求

方法	硬件结构	存储容量
SAWP	ICache 和 BTB 中分别包含 1024 项和 512 项路预测位	4.5K
Way Memorization	每两条指令拥有一个 Way Linker,使用 Zero-Link 机制	25K
WHOLE-Cache	每个 Cache 行和 BTB 项拥有一个 WPG.0-Target 策略	6K
CWS-Cache	每个 Cache 行一个 SWL,每个 BTB 项一个 BWP	5.5K

表 3　TSMC Dophin Library 65nm 在 0.9V,125℃条件下处理器取指部件中 SRAM 时序和功耗参数

Components	SRAM Cell	Timing			Power(1000Mhz,100% act 100% switch)		
		T_{Setup}(ns)	T_{Hold}(ns)	$T_{Clk\text{-}to\text{-}Q}$(ns)	Read_power (mw)	Write_power (mw)	Leakage (nw)
ICache Tag	128×21	0.315	0.077	0.836	4.7	5.9	84858.07
ICache Data	128×256	0.315	0.078	1.050	40.9	54.9	705375.75
SWL &BTB Data	128×32	0.315	0.077	0.871	6.4	8.1	106673.55
BWP	128×8	0.315	0.077	0.855	2.7	3.2	59964.68
BTB Tag	128×24	0.315	0.077	0.866	5.2	6.5	91103.92

4.1　能耗

本文将取指阶段总能耗(E_{Fetch})作为评测标准:

$E_{Fetch} = E_{Tag} + E_{Data} + E_{BTB} + E_{Ext}E_{Tag}$包括 Tag 以及 ITLB 访问的能耗开销.$E_{Data}$为 ICache 中 Data 访问的能耗开销.$E_{BTB}$和$E_{Ext}$分别为 BTB 以及额外硬件的能耗开销.由于基础处理器中 ICache 使用物理标签,虚拟索引的组织方式,并且用于索引的虚拟地址部分不受虚实地址转换过程的影响,因此当使用路历史模式访问时,ITLB 访问与 Tag 访问一样可以被省略.

CWS-Cache 取指阶段能耗开销,如图 7(*a*)所示.相比于基础处理器,其能耗开销平均降低了 84.98%.CWS-Cache 各种模式访问比例,如图 7(*b*)所示,此时 I-Cache 绝大部分情况下为路选择访问,传统方式比例非常低,平均只有 1.57%.以上两个实验结果表明,取指能耗的明显降低,主要是由于 ICache 中几乎全部为路选择访问,从而避免了绝大部分 Tag、ITLB 比较以及几乎全部的非必要路的 Data 访问.另一方面,对比表 2 中对额外存储资源的需求,CWS-Cache 使用额外资源(存储 SWL、BWP 的 SRAM 阵列)较少,因而此部分能耗开销平均只有 2.41%.从处理器整体考虑,CWS-Cache 可以使处理器整体能耗平均降低 15.20%.

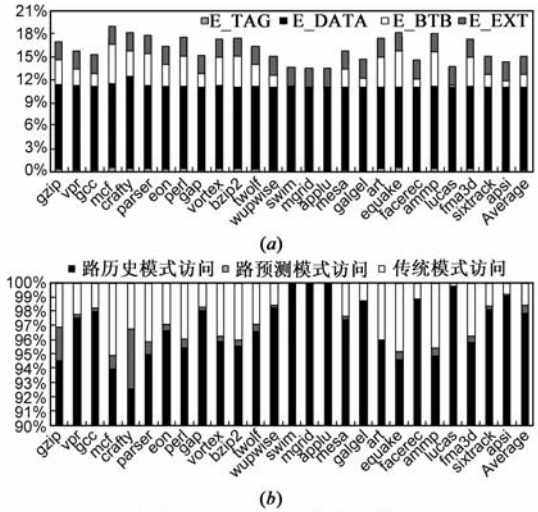

图7　CWS-Cache能耗开销

CWS-Cache 与相关方法的能耗开销比较,如图 8(*a*)所示.由于 SAWP 即使在预测正确时也需要读取 Tag,并且错误预测也会带来额外的能耗,因此 SAWP 低能耗效果相对较差.而 Way Memorization、WHOLE-Cache 等路历史方法由于需要对 BTB 使用整体置无效,因而会丢失很多有效路指针,而导致传统访问比例增加.另外,Way Memorization 方法使用了较多的额外硬件资源,引入了大量的静态能耗.CWS-Cache 避免了上述方法的诸多问题,因此在每个程序上都取得了最佳低能耗效果.

4.2　性能

CWS-Cache 与其他方法在对处理器整体性能的比较,如图 8(*b*)所示.SAWP、Way Memorization 以及 WHOLE-Cache 方法对超标量处理器的性能,分别降低了 7.78%,4.73%以及 3.00%.它们分别受限于错误预测带来的延

迟以及整体置无效 BTB 中路指针引起的取指暂停.CWS-Cache 虽然也会受到预测错误的影响,但是这种影响只有在非顺序取指时才可能发生,平均只有 0.50%.并且 CWS-Cache 有效缩短了非对齐取指组的取指延迟,减少了取指队列空闲的情况,从而避免了其造成的流水线发射暂停,因此性能相对于基础处理器上升了 3.50%.

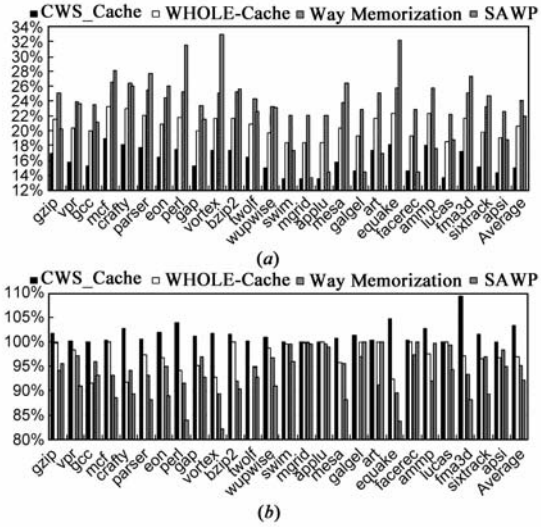

图8 CWS-Cache与相关方法能耗开销及性能影响比较

4.3 能效性

本节采用能耗延迟积(Energy Delay Product, EDP)[17]作为处理器能效性的度量指标,衡量处理器性能与能耗两个方面的综合结果.EDP 越小表明能效性越高,即在性能与能耗两个方面达到了更优的整体结果.以基础处理器的 EDP 为基准值,将 SAWP、Way Memorization、WHOLE-Cache 方法和本文技术的 EDP 作规格化,结构如图 9 所示.在平均情况下,SAWP 反而使 EDP 上升了 0.50%,而 Way Memorization 和 WHOLE-Cache 方法分别使基础处理器的 EDP 降低 0.85% 和 6.22%.实验数据说明,已有方法只针对能耗或性能一个方面进行优化,并不能明显提高整体能效性.而 CWS-Cache 从能耗和性能两方面进行了优化,因此将基础处理器的 EDP 明显降低.相比于其他方法,CWS-Cache 使能效性分别提高了 15.48%,14.13% 和 8.76%.

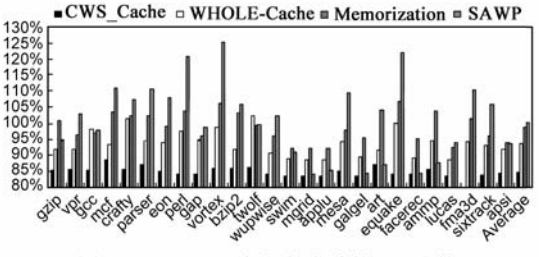

图9 CWS-Cache与相关方法的EDP比较

4.4 不同 ICache 组相联度 CWS-Cache 能耗分析

本节使用 CACTI 5.3[18]得到基本的能耗评估参数.在保持基础处理器 ICache 容量以及 Cache 行大小不变时,取指能耗随 ICache 组相联度从 2 路组相联到 32 路组相联的变化趋势,如图 10 所示.实验表明,随着 I-Cache 组相联度的增大,CWS-Cache 所获得的能耗降低比例逐渐增大:CWS-Cache 将 N 路组相联 ICache 的能耗开销降低为 1/N.但随着 N 的增大,用于存储 SWL、BWL 的额外硬件容量也在增加,导致其动态能耗与静态能耗均有所上升,整体取指能耗降低效果受到影响.但从总体上来看,随着组相联度的增大,CWS-Cache 在降低处理器取指能耗上有着更为明显的效果.

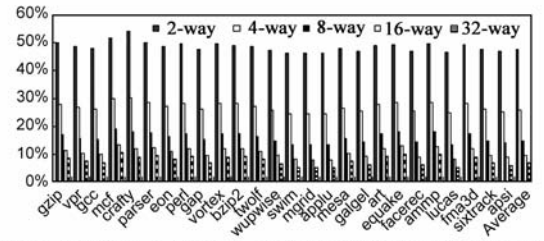

图10 CWS-Cache随ICache组相联度变化的取指能耗开销

5 结论

本文面向超标量处理器中指令缓存,提出了一种高能效的路选择融合技术 CWS-Cache.该技术在不同取指场景中使用不同路选择策略,从整体上取得了更好的能效性.顺序取指时,使用路历史方式避免错误路预测,并缩短取指组非对齐延迟.而对于非顺序取指,使用路预测方式在保留较多预测信息的同时,避免了使用路历史技术时 BTB 中路指针立即更新带来的取指停顿.CWS-Cache 在降低能耗的同时,提升了处理器性能,因而获得了比现有方法更高的能效性.本文方法消耗额外硬件资源较少,实现结构简单,并已在实际处理器中设计实现.

参考文献

[1] O Azizi, et al. Energy-performance tradeoffs in processor architecture and circuit design: A marginal cost analysis [A]. Proc of 37th Ann Int'l Symp Computer Architecture [C]. New York, USA: ACM Press, 2010. 26 – 36.

[2] M D Powell, et al. Reducing set-associative cache energy via way-prediction and selective direct mapping [A]. Proc of 34thAnn Int'l Symp. on Microarchitecture [C]. Göteborg, Sweden: IEEE Society, 2001. 54 – 65.

[3] A Ma, M Zhang, K Asanovic. Way memorization to reduce fetch energy in instruction caches [A]. ISCA Workshop on Complexity Effective Design [C]. Göteborg, Sweden: IEEE Society, 2001.

[4] Z Xie, D Tong, X Cheng. WHOLE: A low energy ICache with separate way history [A]. IEEE Int'l Conference on Computer

Design 2009 [C]. Squaw Valley, USA: IEEE Society, 2009. 138 – 143.
[5] T Ishihara, F Fallah. A way memorization technique for reducing power consumption of caches in application specific integrated processors [A]. Proc of Conf on Design, Automation and Test in Europe 2005 [C]. Munich, Germany: IEEE Society, 2005. 1530 – 1591.
[6] S. Hines, D. Whalley, and G. Tyson. Guaranteeing hits to improve the efficiency of a small instruction cache [A]. Proc of 40th Ann Int'l Symp on Microarchitecture [C]. Illinois, USA: IEEE Society, 2007. 433 – 444.
[7] J Henning. SPEC2000: Measuring CPU performance in the new millennium [J]. IEEE Computer, 2000, 33(7): 28 – 35.
[8] B Calder, D Grunwald, J Emer. Predictive sequential associative cache [A]. Proc 7th Int'l Symp. High Performance Computer Architecture [C]. San Jose, USA: IEEE Society, 1996. 244 – 253.
[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 level methods [J]. Acta Electronica Sinica, 2008, 36(11): 2107 – 2112. (in Chinese)
[10] K Inoue, et al. A low-power I-cache design with tag-comparison reuse [A]. Proc of Int'l Symp on SoC [C]. Tampere, Finland: IEEE Society, 2004. 61 – 67.
[11] T M Conte, et al. Optimization of instruction fetch mechanisms for high issue rates [A]. Proc of 22th Ann Int'l Symp. Computer Architecture [C]. Los Alamitos, USA: IEEE Society, 1995. 333 – 344.
[12] Cache Performance for SPEC CPU2000 Benchmarks [DB/OL]. http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data/ , 2003.
[13] D. Genossar, Y Shamir. Intel Pentium M processor power estimation, budgeting, optimization, and validation [J]. Intel Techonology Journal, 2003, 7(2): 44 – 49.
[14] T Austin, E Larson, D Ernst. SimpleScalar: An infrastructure for computer system modeling [J]. IEEE Computer, 2002, 35(2): 59 – 67.
[15] D. Brooks, V Tiwari, M Martonosi. Wattch: A framework for architectural-level power analysis and optimizations [A]. Proc. of 27th Ann. Int'l Symp. Computer Architecture [C]. Vancouver, Canada: IEEE Society, 2000. 83 – 94.
[16] E Perelman, G Hamerly, B Calder. Picking statistically valid and early simulation points [A]. 12th Int'l Conf on Parallel Architecture and Compilation Techniques [C]. New Orleans, USA: IEEE Society, 2003. 244 – 255.
[17] R Gonzalez, M Horowitz. Energy dissipation in general purpose microprocessors [J]. IEEE Journal of Solid-State Circuits, 1996, 31(9): 1277 – 1284.
[18] S Thoziyoor, et al. CACTI 5.1 [R]. Technical Report HPL-2008-20. 2008.

作者简介

谢子超　男,1984 年生于北京.现为北京大学信息科学技术学院系统结构专业博士研究生.主要研究方向为微处理器前端取指结构、分支预测结构的优化和分析.

陆俊林　男,1980 年生于浙江省.现为北京大学信息科学技术学院讲师.主要研究方向为微处理器设计、软硬件协同设计和片上通信结构.
E-mail: lujunlin@mprc.edu.cn

王箫音　女,1983 年生于河北省.现为北京大学信息科学技术学院博士后.研究方向包括微处理器结构、低功耗高速缓存设计和存储系统性能优化.　E-mail: wangxiaoyin@mprc.pku.edu.cn