计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20160116
54(4):813-820, 2017

众核处理器的流水线紧耦合指令循环缓存设计

张 昆 过 锋 郑 方 谢向辉
(数学工程与先进计算国家重点实验室 江苏无锡 214125)
(zhang.kun@meac-skl.cn)

Design of a Pipeline-Coupled Instruction Loop Cache for Many-Core Processors

Zhang Kun, Guo Feng, Zheng Fang, and Xie Xianghui
(*State Key Laboratory of Mathematical Engineering and Advanced Computing*, *Wuxi*, *Jiangsu* 214125)

Abstract Energy efficiency is a great challenge in the design of future high performance computers. Since the many-core processor becomes a key choice of future high performance computers, the optimization of its micro-architecture is very important for the improvement of energy efficiency. This paper proposes a pipeline-coupled instruction loop cache for the many-core processor. The instruction loop cache is small sized so that it will provide more energy-efficient instruction storage. As an attempt of implementation-aware micro-architecture research, the loop cache is designed under constraints of hardware costs from the beginning. In order to alleviate the impact to the pipeline performance, the loop cache adopts a prefetching technique. The instruction loop cache prefetches the exit path of the loop into the cache when a loop is detected. The prefetching mechanism guarantees that the design of the loop cache in the pipeline can lead to the improvement of the energy efficiency. The instruction loop cache is implemented in the gem5 simulator. Experiments on a set of SPEC2006 benchmarks show that a typical configuration can reduce on average 27% of instruction fetching power and 31.5% power of the pipeline front-end.

Key words loop cache; many-core processor; energy-efficiency; instruction cache; architecture optimization

摘 要 能效比是未来高性能计算机需要解决的重要问题.众核处理器作为高性能计算机的重要实现手段,其微结构的优化设计对能效比提升尤为关键.提出了1种面向众核处理器的流水线紧耦合的指令循环缓存设计,以较小的L0指令缓存提供更加高能效的指令取指.作为体系结构研究同硬件可实现性紧密结合的1次尝试,设计始终考虑了硬件实现代价这一关键约束.为了控制L0指令缓存对流水线性能的影响,指令缓存采用了循环出口预取技术,以此保证指令缓存提供的低功耗的指令取指能够最终转化为流水线能效比的提升.在gem5模拟器上实现了对指令循环缓存的模拟.对SPEC2006的测试结果表明,在不影响流水线性能的前提下,设计的典型配置可以减少27%的指令取指功耗以及31.5%的流水线前段部件动态功耗.

关键词 循环缓存;众核处理器;能效比;指令缓存;结构优化

中图法分类号 TP302

收稿日期:2016-03-07;**修回日期**:2016-06-30

基金项目:国家"八六三"高技术研究发展计划基金项目(2015AA01A301);国家自然科学基金项目(91430214)
This work was supported by the National High Technology Research and Development Program of China (863 Program) (2015AA01A301) and the National Natural Science Foundation of China (91430214).

功耗是未来超级计算机的重大挑战.为了满足E级计算的目标,美国能源部提出了50GFLOPS/W的能效比目标[1].与之相比,目前Green500[2]的第1名,日本Shoubu超级计算机的能效比仅达到7GFLOPS/W.为了实现更高的能效比,需要在追求工艺进步的同时,在体系结构、特别是芯片微结构上做出更多的优化.

众核处理器是高性能计算机的重要实现手段之一.然而,由于片上集成了大量运算核心,众核处理器的设计在功耗和硬件开销上有很多限制.大量的运算核心导致众核处理器功耗较大,过高的功耗会给芯片散热和稳定性带来问题.同时,能效比是未来芯片的重要指标,甚至有可能成为比峰值性能更重要的指标,因此,需要进一步提升众核处理器的能效比.由于片上资源受限,无法采取通用多核处理器的复杂结构、大容量缓存等设计,众核处理器必须采用硬件开销较小的设计,这对提升其能效比又提出了更高的要求.

在冯•诺伊曼结构中,所有的运算均依赖于自存储器中读取的指令,指令的读取、译码这些辅助工作虽然不能直接提供计算能力,但却耗费了较多的芯片功耗[3].因此,本文提出1种面向众核处理器的L0指令缓存设计,同流水线紧耦合的低功耗指令循环缓存.通过同流水线紧耦合的L0指令缓存,可以提供较L1指令缓存更加高能效的取指访问,同时,本文的指令循环缓存位于流水线的取指和预译码站台之后,当取指命中L0缓存时,可以门控循环缓存之前的流水线站台以降低功耗.

为了降低L0缓存的硬件复杂度,以适应众核处理器紧张的硬件资源,本文的L0指令缓存采取了指令循环缓存设计.指令循环缓存技术是经过广泛研究并被数款商用处理器采用的技术,但是本文的循环缓存结合了众核处理器的实际要求,具有3个创新和特点:

1) 本文的指令缓存同流水线紧耦合,不仅作为L0级指令缓存,也作为流水线前段和后段之间的解耦合缓存,一方面减少流水线前段的功耗,另一方面流水线前段和后段的解耦合有利于缓解流水线实现的时序压力.

2) 循环缓存的1个问题是循环出口处带来的流水线气泡,本文提出了循环出口预取技术,有效解决流水气泡问题,使得循环缓存的引入对流水线性能几乎不造成影响.同时,不同于之前的设计,本文循环缓存技术可以在循环的第2次执行开始从缓存中提供指令.

3) 如前文所述,循环缓存的设计必须以较低的硬件开销为前提.本文的工作是体系结构研究同硬件可实现性紧密结合的1次尝试,不仅考虑了指令缓存的可实现性,还讨论了指令缓存同1个示例流水线的紧耦合方式.

1 相关工作

在循环缓存方面有较多的先前工作,本文同这些工作的首要区别在于本文采用了同流水线紧耦合的循环缓存设计,这就解决了几乎所有循环缓存都需要经历"检测循环、装填循环、输出循环"这3个步骤的问题,即直到第3次循环迭代,循环缓存才可以向流水线后段输出指令.

文献[4]中提出了利用指令流中的sbb(short backward branch)指令来判断指令流是否命中循环缓存,sbb指令定义为转移距离较短的回跳转移指令,当转移指令的偏移量小于循环缓存的容量时,可以认为循环体命中了循环缓存.文献[4]使用1个指令计数器来判断循环缓存是否已经输出到sbb指令,这也就意味着文献[4]的结构中只能够包含1个循环体而无法处理内嵌循环.文献[4]面向嵌入式处理器进行设计,且使用了嵌入式应用测试集进行评测,其结果表明在超过32条指令之后其设计的循环缓存命中率就基本上不再增加.

文献[5]提出了1种基于指令队列的程序循环代码动态检测技术.其设计类似于文献[4]的循环缓存实现,区别在于该设计不再使用sbb定义而是通过跳转目标PC同循环缓存存放的第1条指令PC值的关系来判断是否命中循环缓存,但是其同样需要到第3次循环迭代时才可以输出循环.

文献[6]针对DSP处理器设计了1种两路循环缓存,该设计直接存储回跳转移指令的PC值来判断循环缓存是否输出至循环尾部指令.文献[6]同样使用1个循环计数器来判断循环体是否已经输出结束,因此也导致其无法处理嵌套循环.

文献[7]利用加入新的指令以及编译器的帮助,在编译过程中插入适用于循环缓存的辅助指令,以此应对循环体内部的前向转移指令.编译器在进行程序切片时,选择执行次数多的循环插入辅助指令以提示硬件逻辑对循环进行装载.同时,该文给出了1种分层次的译码后指令存储,用以减小循环缓存的容量需求.文献[7]的设计需要编译器和指令集的支持,一定程度上限制了其兼容性和实用性,另外该

技术将前向转移指令的 2 个方向都进行缓存,在某些应用中可能造成功耗的浪费.

文献[8]中利用 BTB 存储前向跳转的信息,按照 BTB 中的预测方向处理循环中的前向跳转.这种设计避免了对编译器和指令集的修改,但是,其对前向跳转的处理依赖于 BTB 的预测成功率,在众核处理器的简单核心中包含 1 个足够大小的 BTB 以及其相应控制逻辑对硬件资源的需求较高.

2　指令循环缓存结构

指令循环缓存实现为 1 个先入先出的队列,存储动态指令流,当检测到包含在循环缓存中的循环时,下一次循环执行的指令将直接从指令循环缓存中读取,进而门控自流水线头部取指部件到预译码部件之间的大部分逻辑,有效减少流水线前段的动态功耗.

2.1　总体结构

包含指令循环缓存的示例流水线前段(front-end,本文指流水线的取指和预译码部分)如图 1 所示.示例流水线包含 2 级取指站台和 3 级译码站台,指令循环缓存位于预译码部件之后.离开预译码部件的指令全部写入指令缓存中,之后再从指令缓存中读出,送至流水线后段.

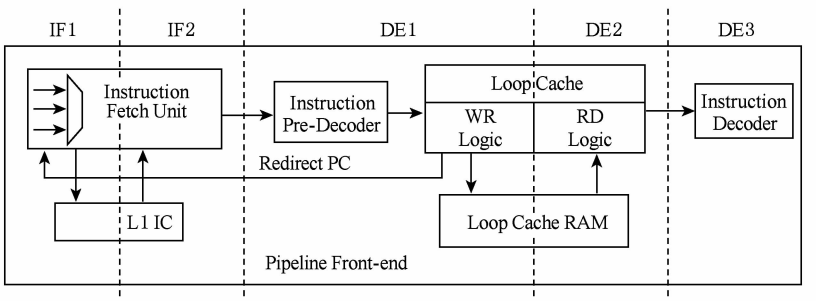


Fig. 1　Pipeline front-end with the loop cache

图 1　包含循环缓存的流水线前段

指令循环缓存的总体结构如图 2 所示.循环缓存中存放连续的顺序指令流,并维护所存顺序指令流的首条指令 PC 值(下文称为 HeadPC).当作为循环体结尾的转移指令离开循环缓存时,将根据其跳转目标 PC 值判断该循环是否包含在循环缓存中,若目标 PC 值大于等于 HeadPC 时,表示循环命中缓存,当拍产生跳转目标指令的读地址,下一拍循环缓存开始从循环体头部输出指令.

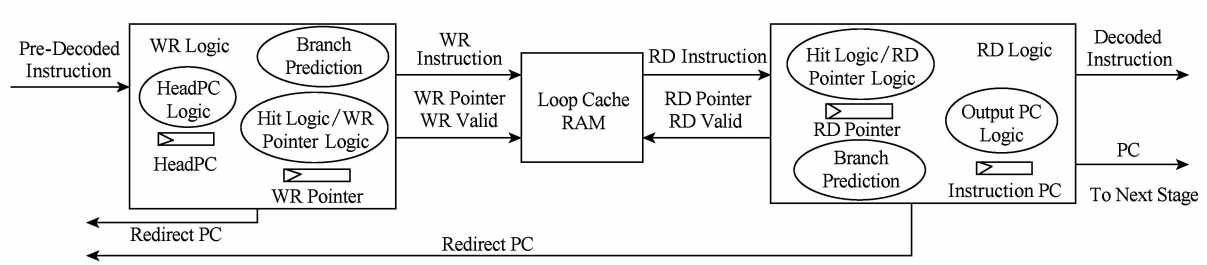


Fig. 2　The structure of the loop cache

图 2　指令循环缓存总体结构

循环缓存分为写入控制逻辑、读出控制逻辑和存储体 3 部分组成.众核处理器的运算核心一般采用简单的流水线设计,因此,不同于基于 BTB 转移机制的流水线中直接根据取指 PC 值进行下一拍取指的重定向,本文中的流水线前段重定向信号来自预译码的结果.进一步地,重定向信号来自于循环缓存的写控制逻辑,当写控制逻辑发现不命中循环缓存的转移指令时,即可发出取指重定向信号.由于命中循环缓存后,循环体的多次执行将直接由读控制逻辑从缓存存储体中取指,指令不再流经写控制逻辑,因此,在写控制逻辑和读控制逻辑中,均含有转移预测部件.

循环缓存一般设计为"检测到循环"、"装填循环体"、"读取循环体"3 个步骤,这种设计的缺点是直到循环体的第 3 次执行才可以利用循环缓存.本文采取了同流水线紧耦合的循环缓存设计,流水线的动态指令流均需要写入循环缓存,因此当读控制逻辑发现有效命中的循环体时,循环的第 2 遍执行就

可以从指令循环缓存中取指.

同一般的 L0 级指令缓存设计不同,本文的指令循环缓存不仅可以在循环命中时作为 L0 级指令缓存使用,由于其放置在流水线的预译码部件之后,可以有效将流水线的“生成地址、取指部分”同“译码、分派、执行部分”解耦合.解耦合有利于减少流水线站台之间的控制信号依赖,缓解流水线实现的时序压力.

本文的循环缓存设计考虑了硬件实现代价,如图 2 所示,循环缓存主要包括 4 组触发器,即首指令 PC(HeadPC)、写指针、读指针和输出指令 PC.逻辑电路主要基于读写指针的操作,因此硬件复杂度较低,在 2.2~2.4 节中将对各部分逻辑的硬件实现做简要分析.

2.2 写控制逻辑

写控制逻辑主要包含循环缓存命中判断逻辑、写指针控制逻辑和 HeadPC 控制逻辑.

写控制逻辑接收预译码部件送来的指令组,判断指令组中是否有转移指令且转移指令是否命中循环缓存.若命中循环缓存,则通知流水线前端暂停取指,循环体的下一次迭代可以从循环缓存中直接取指;若不命中循环缓存,则通知流水线的取指部件进行取指目标重定向.需要特别说明的是,对于向前跳转的条件转移指令,若预测其不跳转,则在本文的写控制和读控制逻辑中将其当作顺序指令处理,因此本文的结构是部分支持循环体中内嵌前向转移指令的.

写控制逻辑根据当前写入的有效指令数目,为各条指令分配其在存储体中的写入位置,同时写指针相应地增加.在本文的循环缓存中,写指针值一直取模增加,写指针增加后,指向下一拍到达的首条指令需要写入的位置.

当程序初始运行时,HeadPC 设置为程序的首 PC(即首次到达写控制逻辑的有效指令的 PC 值),一旦发生不命中循环缓存的转移,则将 HeadPC 置为该转移指令的目标 PC,之后当取指填满循环缓存并开始覆盖最先指令时,HeadPC 随之递增.

硬件开销方面,循环缓存的命中判断逻辑核心为一个比较器,比较器用于判断转移指令的跳转偏移量是否小于循环缓存的容量;写控制逻辑的核心为一个宽度为写指针宽度的加法器,按照每次写入指令的数目增加;HeadPC 控制逻辑实现为多路选择器,根据不同的情况对 HeadPC 进行赋值.

2.3 读控制逻辑

读控制逻辑包含命中判断逻辑、读指针控制逻辑以及输出 PC 控制逻辑.

由于在读控制逻辑中存在转移预测部件,自循环缓存中读出指令的预测方向可能异于该条指令在写入存储体之前的预测方向,因此,转移指令需要在读控制逻辑中重新判断其是否命中循环缓存.

当循环体的结尾转移指令在写控制逻辑判断为命中且在读控制逻辑中也判断为命中时,下一次循环体执行可以直接从循环缓存中取指;当循环体的结尾转移指令在写控制逻辑中判断为命中而读控制逻辑判断为不命中,则意味着循环体到达循环出口,需要通知流水线前段从循环出口重定向取指;同时,存在在写控制逻辑中判断为不命中而在读控制逻辑中判断为命中的转移指令,这是因为对于同一条指令,其转移预测方向可能改变,这种情况属于内嵌的循环,本文的循环缓存完全支持这种内嵌循环而不需要任何软件支持.

读控制逻辑根据读写指针的关系判断缓存中是否存在未读取的指令,只要读指针位于写指针之前,则发起有效的读动作.读控制逻辑产生存储体的读地址,并根据存储体输出的指令命中循环缓存的情况,置位下一拍的读地址.对于命中循环缓存的转移指令,其目标指令在存储体中的位置可以通过转移指令偏移量计算而得.当读控制逻辑发现需要取指重定向时,直接将读指针置为写指针并等待重定向取指到达循环缓存即可.

为了减小存储体开销,指令写入存储体时,不再保存其 PC 值,而是通过读控制逻辑产生.对于顺序执行的指令,PC 值逐条递增即可;当在读控制逻辑发现转移指令并预测为跳转时,PC 值将置为跳转的目标地址;当执行部件发现转移预测失败时,PC 值将设置为流水线恢复后首条到达写控制逻辑的指令 PC 值.

硬件开销方面,读指针逻辑的核心为一个多路选择器,用于选择递增地址或是命中后的循环首条指令地址;读控制逻辑中的命中判断逻辑包含若干种条件(在写控制逻辑中是否预测为跳转、在读控制逻辑中是否预测为跳转等)的综合;指令 PC 值逻辑也是一个多路选择器和加法器,用于选择加法器运算得到的顺序递增 PC 值或是转移指令的跳转目标地址.

2.4 循环出口预取

当循环体命中指令循环缓存时,后续的若干次迭代可以从循环缓存中取指,当循环体尾部转移指令在读控制逻辑中预测为不跳转时,意味着循环体到达循环出口,需要执行其尾部转移指令的顺序后继指令,由此产生流水线取指重定向,而重定向取指将带来自流水线头部到循环缓存站台之间若干拍的

气泡.若应用中的循环体迭代次数较少,则上述的气泡将造成性能损失.

为了解决循环出口带来的性能损失,本文的循环缓存设计引入了循环出口预取机制,通过提前将循环出口的若干条指令写入循环缓存,可以有效去除循环执行到出口时的性能损失.

循环出口预取由读控制逻辑发起,当循环体尾部转移指令首次到达读控制逻辑并判断为命中时,意味着循环体即将开始第 2 次的迭代,此时读控制逻辑通知流水线头部取指逻辑进行循环出口地址的预取.预取取指的数目采用信用管理,读控制逻辑根据当前读写指针关系,计算出在不覆盖已命中循环体的前提下存储体能够容纳的预取指令数目,通知流水线的取指部件.流水线取指部件按照信用取指若干次后再次暂停取指.

循环出口预取不产生任何额外的开销,因为循环出口是一定会被执行的.循环出口预取机制的示意如图 3 所示.预取取指到达存储体后,将使写指针增长.循环迭代过程中,每次到达循环体尾部转移指令并预测为跳转,读指针移动至循环体头部位置重新开始输出循环体指令;循环体最后 1 次迭代,到达尾部指令并预测为不跳转时,读指针可以直接移动到顺序的下一个位置,即可正确读取预取的指令.同时,读控制逻辑通知流水线取指部件恢复取指.

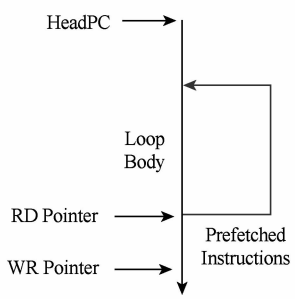


Fig. 3　The prefetching of the loop exit

图 3　循环出口预取示意图

硬件开销方面,循环出口预取逻辑需要计算预取信用值,通过计算读指针同循环体头部指令位置之间的相对关系,可以得到是否有足够的空间进行出口预取.

3　实　　验

3.1　实验方法

本文使用 gem5[9] 模拟器对指令循环缓存结构进行建模.众核处理器一般采用大量精简计算核心,因此本文在双发射顺序流水线上实现循环缓存结构.循环缓存的命中判断取决于循环体最后 1 条转移指令在流水线中的转移预测结果,因此,转移预测机制的准确度对循环缓存的性能测试有所影响.由于不同的众核处理器采用不同的转移预测实现思路,为了消除转移预测对测试带来的影响,本文采用理想的转移预测器,即保证每次的转移预测结果都是正确的.

本文测试中使用的主要配置情况如表 1 所示:

Table 1　The Configuration of the Simulator

表 1　模拟器配置

Parameter	Configuration
Pipeline	In-order, 2-way
Branch Predictor	ideal
Fetch Width/Instructions per Cycle	2
Number of Stages in the Front-end	4
Capacity of the L1 IC/KB	16

表 1 中流水线前段级数指流水线首部到指令循环缓存之间的站台数,实验中按图 1 所示结构的数据,该级数将影响循环出口预取的性能,因此特别列出.

本文使用 SPEC2006 测试集[10] 对指令循环缓存结构进行性能测试.gem5 模拟器无法正确运行 SPEC2006 测试集中所有的程序[9].虽然其中部分程序经过对模拟器的修改可以完成运行,为了保证测试结果的一致性,本文选用了 SPEC2006 中的 12 道程序进行测试,具体程序如表 2 所示.测试包括引入循环缓存之后的流水线性能、循环缓存的命中率、循环缓存带来的功耗降低以及循环出口预取等测试.

Table 2　Benchmark List

表 2　测试程序列表

Benchmark	Language	Application
bzip2	ANSI C	Bzip2 v1.0.3
mcf	ANSI C w/libm	MCF v1.2 Combination Optimization
gobmk	C	The Go Game
hmmer	C	Genome Sequencing
libquantum	ISO/IEC 9899:1999	Quantum Computing
h264ref	C	Video Compression
milc	C	MILC Quantum Physics
gromacs	C & Fortran	GROMACS Biological Chemistry
leslie3d	Fortran 90	LESlie3d Hydrodynamics
namd	C++	NAMD Biology/Mechanics
calculix	Fortran90&C w/SPOOLES Code	CalculiX Structural Mechanics
soplex	C++	Linear Programming

本文采用 CACTI 存储器功耗模型[11]对指令缓存的访问功耗进行评估. 本文对 32 条指令、64 条指令、128 条指令、256 条指令这 4 种配置的循环缓存进行测试,实验中流水线的 L1 指令缓存容量设为 16 KB. 上述 5 种容量的存储器访问功耗如表 3 所示:

Table 3 The Energy of Memory Accesses

表 3 存储器访问功耗

Capacity	Access Energy/nJ
128B: 32 Instructions	0.000 01
256B: 64 Instructions	0.002 56
512B: 128 Instructions	0.002 59
1KB: 256 Instructions	0.002 59
16KB: L1 IC	0.018

表 3 中 128 B 到 1 KB 存储器数据使用单体 RAM,行宽度为 8 B;16 KB 存储器数据使用 4 体 RAM,行宽度为 64 B. 表 3 中数据基于 32 nm 工艺.

3.2 实验结果

本文分别对容量为 32 条指令、64 条指令、128 条指令、256 条指令这 4 种配置的循环缓存进行了命中率测试. 图 4 展示了 4 种配置下的命中率结果,横着给出了 12 道测试程序,纵轴为各程序在循环缓存中的命中率,不同色块表示不同容量的指令循环缓存. 从图 4 可以看出,循环缓存容量的翻倍能够带来 5%左右的命中率提升,当缓存容量到达 256 条指令时,其平均命中率达到 37%. 文献[4]中的平均命中率在循环缓冲容量到达 32 条指令之后就不再有效增长,本文的结果同文献[4]结果的差异主要在于 2 点:1)同文献[4]中给出的结果相比,本文使用的是 SPEC2006 的测试集,由于文献[4]是面向嵌入式处理器而设计的循环缓存,因此其使用的是针对嵌入式系统的 PowerStone[12]测试集,因此在命中率的绝对值上有差异;2)本文的循环缓冲支持内嵌的循环和内嵌的预测为不跳转的前跳转移指令,当容量扩大时,满足该条件的大循环可以被装载入循环缓冲,因此本文缓存命中率随着缓存容量的增大而持续增加. 为了降低实现复杂度,本文的循环缓存没有完全支持循环中内嵌前向跳转的转移指令,因此本文的循环缓存命中率在某些应用中不足 10%. 但是后续实验表明,由于采用了循环出口预取,本文的循环缓存对流水线性能不会带来影响,因此循环缓存带来的能效收益始终是存在的.

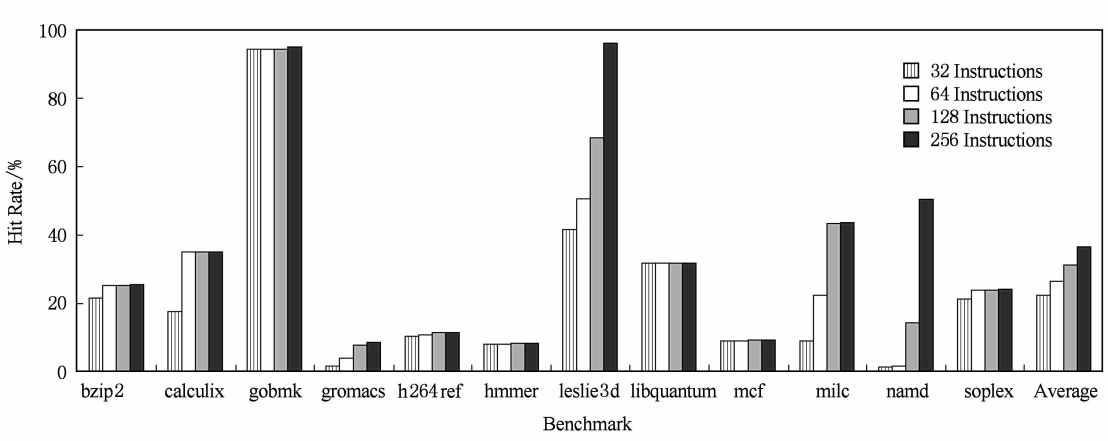


Fig. 4 Hit rate of the loop cache

图 4 指令循环缓存命中率

本文采用同流水线紧耦合的循环缓存设计,并通过循环出口预取来消除循环退出时的流水线气泡,对循环出口预取的测试如图 5 所示. 图 5 中给出了使用或不使用预取机制时循环缓存对流水线性能带来的影响,例如,在没有预取机制且缓存容量为 32 条指令时,循环缓存会造成流水线性能 1.48%的损失. 没有预取机制时,循环缓存容量的增大会导致性能损失略微增加. 这是因为更大的循环缓存将带来更高的命中率,而使得更多的循环体在循环缓存中被读出,因此遇到循环出口的次数也会增加,如 2.4 节所述,更多的循环出口会带来更大的性能损失. 由图 5 可以看出,采用循环出口预取可以有效地减少循环出口退出时的性能损失. 当循环缓存容量达到 64 条指令或以上时,采用循环出口预取技术可以认为循环缓存不再对流水线性能带来影响.

利用循环缓存的命中率结果以及表 3 中的数据,可以对循环缓存带来的存储器访问功耗降低进行计算,如图 6 所示. 如表 3 所示,不同容量的存储器访问功耗不同,当一部分指令取指命中在循环缓存中时,其访问功耗将显著降低,按照式(1)可以

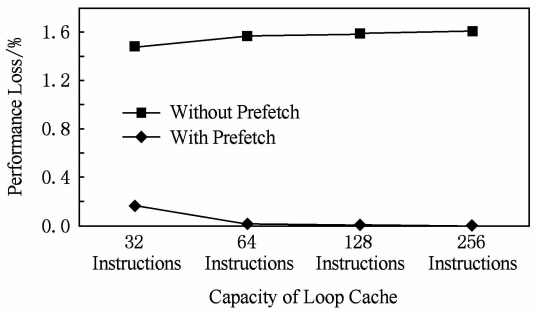


Fig. 5　The effect of loop exit prefetching

图 5　循环出口预取机制的效果

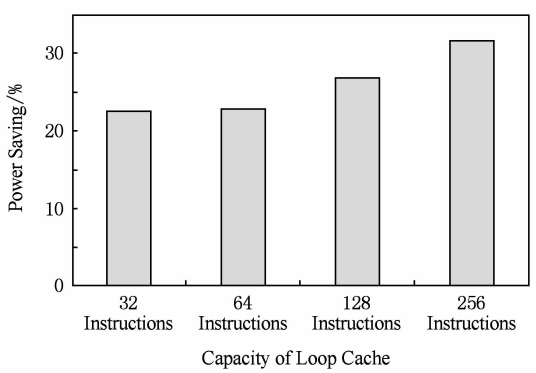


Fig. 6　The power reduction of pipeline's instruction unit from the loop cache

图 6　循环缓存对流水线指令部件功耗的降低

计算得出 1 次指令缓存访问的平均耗能. 其中 E_* 表示指令访问的耗能,P_* 表示取指在循环缓存或者 L1 指令缓存中的命中率. 利用式(1),可以计算出循环缓存技术可以降低 30%左右的指令缓存访问功耗.

$$E_{ave}=P_{L0}\times E_{L0}+P_{L1}\times E_{L1}. \tag{1}$$

由于在循环缓存中命中后,可以对流水线前段站台进行门控,进而节省流水线前段站台的功耗. 显然,门控前段站台的比例等于取指命中循环缓存的比例(如图 4 中 Average 所示),因此循环缓存可以进一步降低 22%～37%的流水线前段站台功耗.

4　结束语

本文作为体系结构研究同硬件可实现性紧密结合的一次尝试,提出了一种同流水线紧耦合的指令循环缓存设计. 指令循环缓存不仅充当流水线前段与后段之间的一个解耦合指令队列,也可以作为 L0 指令缓存以提供高能效的指令取指. 实验结果表明,在不牺牲流水线性能的前提下,以容量为 128 条指令的循环缓存为例,指令循环缓存可以降低 27%的指令取指存储器功耗和 31.5%的流水线前段逻辑动态功耗.

未来工作可以在考虑有效控制实现复杂度的同时,增加循环缓存对内嵌有前向跳转指令循环的全面支持,由此可以带来更大的能效收益. 同时,本文的循环缓存实现在流水线的译码部件之前,存放的是译码前的指令,后续的工作可以考虑将循环缓存放置在译码部件之后,但是译码部件后的指令循环缓存同流水线的耦合方式将需要重新考虑.

参　考　文　献

[1] Hemsoth N. Future challenges of large-scale computing [EB/OL]. (2013-04-15) [2016-04-21]. http://www.hpcwire.com/2013/04/15/future_challenges_of_large-scale_computing

[2] The Green500 Organization. The Green500 list - November 2015 [EB/OL]. (2015-11-08) [2016-04-21]. http://www.green500.org/greenlists

[3] Manne S, Grunwald D, Klauser A. Pipeline gating: SPECulation control for energy reduction [C] //Proc of the 25th Int Symp on Computer Architecture. Los Alamitos, CA: IEEE Computer Society, 1998: 132-141

[4] Lea H L, Moyer B, Arends J. Instruction fetch energy reduction using loop caches for embedded applications with small tight loops [C] //Proc of the IEEE Int Symp on Low Power Electronics and Design. Los Alamitos, CA: IEEE Computer Society, 1999: 267-269

[5] Zhang Chengyi, Gao Jun, Sun Caixia, et al. Program loop code and execute dynamic detection method based instruction queue: China, CN102968293 A [P]. 2013-03-13 (in Chinese)
(张承义, 高军, 孙彩霞, 等. 基于指令队列的程序循环代码动态检测及执行方法: 中国, CN102968293 A [P]. 2013-03-13)

[6] Anderson T, Agarwala S. Effective hardware-based two-way loop cache for high performance low power processors [C] //Proc of the Int Conf on Computer Design. Los Alamitos, CA: IEEE Computer Society, 2000: 403-407

[7] Gu J, Guo H. Enabling large decoded instruction loop caching for energy-aware embedded processors [C] //Proc of the 2010 Int Conf on Compilers, Architectures and Synthesis for Embedded Systems. New York: ACM, 2010: 247-256

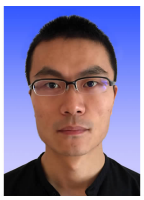
[8] Tein B H, Wu I W, Chung C P. Instruction fetch energy reduction using forward-branch bufferable innermost loop buffer [C] //Proc of the Int Conf on Computer Design. Las Vegas, GA, USA: CSREA Press, 2006: 91-96

[9] Binkert N, Beckmann B, Black G, et al. gem5 simulator: A modular platform for computer-system architecture research [EB/OL]. (2016-03-23) [2016-04-21]. http://www.gem5.org

[10] Standard Performance Evaluation Corporation. SPEC CPU2006 benchmark [EB/OL]. (2015-08-18) [2016-04-21]. http://www.spec.org/cpu2006

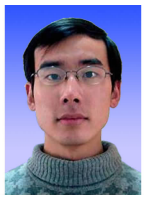
[11] HP Corporation. CACTI: An integrated cache and memory access time, cycle time, area, leakage, and dynamic power model [EB/OL]. [2016-04-21]. http://www.hpl.hp.com/research/cacti

[12] Scott L, Lee L, Arends J, et al. Designing the low-power M-CORE architecture [C] //Proc of the IEEE Power Driven Micro Architecture Workshop at ISCA98. Los Alamitos, CA: IEEE Computer Society, 1998: 145-150



Zhang Kun, born in 1984. PhD candidate, engineer. His main research interests include high performance computing and processor micro-architecture.

Guo Feng, born in 1977. PhD, engineer. His main research interests include high performance computing and processor micro-architecture (gf90966@gmail.com).



Zheng Fang, born in 1984. PhD, engineer. His main research interests include high performance computing and processor architecture (zheng.fang@meac-skl.cn).

Xie Xianghui, born in 1958. PhD, professor, PhD supervisor. His main research interests include computer architecture and parallel computing (xie.xianghui@meac-skl.cn).

《信息安全研究》期刊简介

习近平总书记指出“没有网络安全就没有国家安全,没有信息化就没有现代化”.数字时代信息安全工具的大众化是不可阻挡的历史潮流.大众化的信息安全已经直接影响到我们每个人的利益,信息安全已成为国家、地方区域经济结构优化提升和转型发展的新机遇.在信息安全上升为国家战略、行业迎来崭新发展机遇形势下,《信息安全研究》期刊应时代而生.

《信息安全研究》是由国家发改委主管、国家信息中心主办的中文学术期刊,其宗旨是集中展示和报道国际、国内网络和信息安全研究领域研究成果及最新应用,传播信息安全基础理论和技术策略,服务国家信息安全形势发展需要.所刊登的论文均经过专家严格评审.

《信息安全研究》于2015年10月创刊发行.刊期为月刊,每期96页,由《信息安全研究》杂志社出版,国内外公开发行.

《信息安全研究》将以研究致以应用,搭建信息安全领域的学术交流平台,愿意和同行业及社会各界建立联系,友好合作,共赢美好未来.欢迎大家积极投稿、赐稿,洽谈合作.

投稿信箱:ris@cei.gov.cn

编辑部联系人:崔先生(185 0008 6481)

马先生(158 1058 2450)