改进的基于O-GEHL预测技术的EDGE块预测器

苟鹏飞，王诗博，杨 兵，喻明艳

(哈尔滨工业大学微电子中心 哈尔滨 150001)

【摘要】块预测器赋予了EDGE体系结构高效的控制流推测能力。针对基于O-GEHL预测技术的块预测器的缺点，提出了两种改进策略：1) 无选择器的O-GEHL出口预测；2) 直接使用OGEHL预测器的二值预测能力单独对每个出口进行预测。性能评价结果表明，无选择器的O-GEHL出口预测方案比文献[4]中的O-GEHL块预测器性能平均提高0.7%；对8个出口分别使用二值O-GEHL预测器进行预测的方案，在硬件资源较多时，性能提高3%；只对前4个出口采用二值O-GEHL预测器的方案，性能平均提高2%。

关 键 词 块预测器; 分支预测器; EDGE体系结构; 出口预测; O-GEHL预测技术; 性能评估

中图分类号 TP338.1 文献标识码 A doi:10.3969/j.issn.1001-0548.2012.02.025

Improved Next-Block Predictor Based on O-GEHL for EDGE Architectures

GOU Peng-fei, WANG Shi-bo, YANG Bing, and YU Ming-yan

(Microelectronic Center, Harbin Institute of Technology Harbin 150001)

Abstract Next-block predictors enable high efficiency control-flow speculation of EDGE architecture. This paper analyzes defects of next-block predictor based on O-GEHL prediction technology, and proposes two improvements: the exit prediction without chooser and the exit prediction with binary O-GEHL prediction. Performance evaluation results show that: the one without chooser in the exit prediction stage outperforms previously published one by 0.7%; the one using 8 conditional O-GEHL predictor improves the performance by 3% with the largest resource; the one using 4 conditional O-GEHL predictor only for the first 4 exits improves the performance by 2%.

Key words block predictor; branch predictor; EDGE architecture; exit Prediction; O-GEHL predictor; performance evaluation

显式数据流图执行(EDGE)体系结构[1]被认为是较有前景的下一代处理器结构之一，其以指令块为单元进行取指/执行/提交(block-atomic fetch/execute/commit)。当代的EDGE处理器，如TRIPS和TFlex，通过单入口、多出口的甚块(hyperblock)维持块原子性[2-3]。为了保证高效地执行模型，EDGE处理器采用控制流推测技术取得甚块。

为保证高效的指令块层面的控制流推测，文献[4-6]提出了针对EDGE体系结构的块预测器。该预测器使用出口预测和目标地址预测相结合的方式，实现对指令块预测中“*N*选1”问题的解决。经过性能评估后发现，块预测器中的出口(exit)误预测约占所有误预测次数的50%[4-5]。这预示着出口预测是块预测器的性能瓶颈，并为进一步优化块预测器留下了空间。

文献[7]提出了O-GEHL预测器，其相对简单的结构和良好的预测效果对于块预测技术有着很高的研究价值。

本文分析了目前O-GEHL块预测器的缺点，针对不足提出了两种主要的改进方案：1) 无选择器的出口预测；2) 直接利用传统O-GEHL二值预测能力的出口预测。评估结果表明，在相同配置情况下，采用无选择器的出口预测方案比文献[4]中的O-GEHL块预测器性能平均提高了0.7%；采用8个传统O-GEHL预测器的块预测器在最高配置(1 MB)的情况下性能提高3%；对前4个出口采用传统O-GEHL预测器的块预测器则在资源范围从16 KB～1 MB的情况下将性能平均提高了2%。

收稿日期：2011 – 07 – 15； 修回日期：2011 – 11 – 15

作者简介：苟鹏飞(1983 –)，男，博士生，主要从事高性能微处理器研究.

1 背景知识与相关工作

1.1 块预测器

对有单个/多个出口的类甚块预测研究已有数十年[5,10,12-14]。在这些研究中，文献[5-6]和文献[10]研究的出口预测是最流行的技术之一。因此，目前的块预测器基本由两部分组成：出口预测器和类型/目标地址预测器[5]。一般地说，出口预测器是由目前最先进的二值分支预测器修改得到的，它预测甚块的实际出口而不是分支指令的跳转方向。文献[4]中使用了一系列目前最先进的二值预测器设计EDGE的出口预测器，为出口预测器提供了广阔的、系统的研究视角。

1.2 O-GEHL预测技术及其在块预测器中的应用

O-GEHL预测器[7]使用一组呈几何级数增长的全局历史索引不同的表。该方式能有效地捕捉到不同历史长度的相关信息。O-GEHL预测器示意图如图1所示。当前流行的TAGE预测技术是在借鉴O-GEHL部分技术的基础上发展起来的，其在存储资源较大的情况下性能不如O-GEHL预测技术[15-16]。

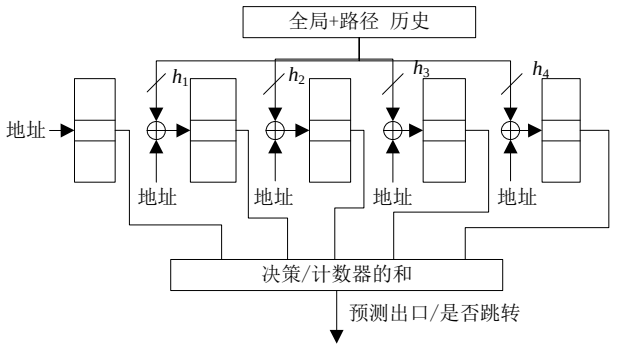

图1 O-GEHL预测器示意图

文献[4]实现了O-GEHL预测技术在块预测器中的应用，其将O-GEHL映射到出口预测器上的策略主要有：1) 将加法树改为多数投票机制；2) 在多数投票出现平局的情况下，使用备用的选择器选择预测的出口；3) 将部分更新改为全部更新；4) 为更有效地利用本地相关信息，将全部使用全局历史信息改为包含一个使用本地历史信息的成员。

然而，实验结果表明，使用该映射策略实现的O-GEHL块预测器性能并不好。主要原因有两点：1) 多数投票代替加法树，使得预测结果在平局情况下不能直接得到，而需要进一步判断；2) 由于选择器的低性能导致了多数投票平局情况下O-GEHL预测器性能的降低。从分析可知，如果能找到更好的将O-GEHL预测技术映射到块预测器上的方法，那么块预测器性能是能够获得提高的。

2 实验环境与评价方法

本文所有的数据和分析都基于下面描述的实验环境和评价方法。

2.1 仿真环境

本文采用M5_EDGE[11]作为仿真工具，其基于M5的乱序模型构建的、周期精确的EDGE体系结构研究框架。本文所使用的基准处理器配置参数如表1所示。

表1 基准处理器的配置

取指	每周期16条指令(一个Cache行)
映射策略	1周期延迟的静态映射
分派开销	1周期
窗口大小	1 K指令（8个TRIPS的甚块）
操作数网络延迟	0延迟
执行层	发射的峰值是每周期16条指令
谓词化	完美的
递交延迟	1周期
Cache (高速缓存)	全命中
内存访问顺序	完美的

本文采用的EDGE指令集是具有较成熟的工具链的TRIPS指令集。评价中，使用SPEC CPU2000中能被TRIPS工具链正确编译的11个整型程序作为测试程序，并使用了ref输入集。为将仿真时间控制在可接受的范围，仿真时采用Simpoint仿真方法[17]。

2.2 性能衡量指标

MPKI(mispredicts per kilo instructions，每千条指令误预测数)能较准确地反映出预测器性能，许多文献都使用MPKI对预测器性能进行评估[7,5]。MPKI越低，表明误预测数越少，性能越好。因此，在性能分析过程中，本文使用MPKI作为评判预测器有效性的标准。

3 无选择器的OGEHL出口预测

对现有的O-GEHL块预测器进行的性能分析表明，当多数投票机制出现平局时，使用备用的选择器进行预测以得到最后的结果，该方法的性能均令人失望。主要原因在平局时，选择器的低效导致了块预测器整体准确预测率的降低[4]。

为此，本文针对原文中的预测器，统计了SPEC2000中11个整型测试程序在块预测器大小分别为16～128 KB时的误预测来源，并将其分为选择器导致和非选择器导致两大部分，如图2所示。从图中可以发现，除个别总体误预测次数较少的测试程序(如Perlbmk)外，大部分测试程序中选择器的误预测在整个出口误预测中占有相当的比例，即选择器

导致平均大约45%的出口误预测。如果可以将这部分误预测减少，将使预测器的整体性能得到较大的提高。鉴于该情况，本文提出了一种不使用选择器的策略。

本文策略的具体实施方案是：在每次更新时均检查正确出口结果与每个成员预测表的预测结果是否匹配，并进行记录；在预测时仍然使用多数投票的机制，即如果某个出口的值(置信度)超过了设定的阈值，块出口预测器会将该出口号作为最后的预测结果；否则，会通过查找记录，使用预测正确次数最多的那个预测表的预测结果作为最后预测的出口。如算法1所示。

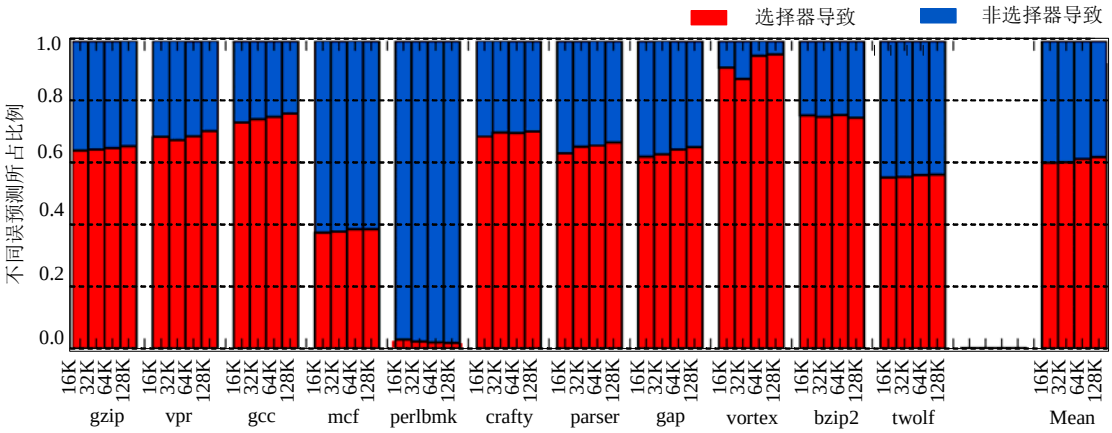

图2　SPEC2000中11个整型测试程序出口误预测数的分解示意图(纵坐标是MPKI归一化后的结果)

算法1　无选择器的基于O-GEHL预测技术的出口预测算法。

输入：全局/本地历史(*H*)，块地址(Baddr)，成员预测器预测正确次数(*U*)

输出：块出口(PredExit)

```
if exit0_majVote > threshold then
   PredExit = exit0;
end
……
else if exit7_majVote > threshold then
   PredExit = exit7;
end
else
   second_pred = RightPredMost(U);
   switch second_pred do
      case g4
         PredExit = GlobalPred4(H, BAddr);
      End
……
      case g1
         PredExit = GlobalPred1(H, BAddr);
      end
      case local
         PredExit = LocalPred(H, BAddr);
      end
      case bimodal
         PredExit = BimodalPred(H, BAddr);
      end
   end
end
```

算法描述中，second_pred是备选成员；RightPredMost是挑选预测正确次数最多的预测表；GlobalPred4是历史位最长的全局二级预测表，依此类推；LocalPred是本地二级预测表；BimodalPred是双峰预测表。如果有预测表的正确预测次数相等，则默认选择历史更长的预测表的输出作为最终结果。第5节将对文本预测器的性能进行评估。

4　利用O-GEHL二值预测的出口预测

从第1.2节的分析可得，用多数投票机制代替加法树可能是导致O-GEHL块预测器性能差的主因。因此本文提出了第二种对O-GEHL块预测器的改进策略。该改进策略的块预测器将利用传统的O-GEHL二值预测器完成对出口的预测。简单地说：1) 二值预测器在某个出口预测跳转意味着该出口很有可能是最后预测的出口；2) 对于单个的二值预测器而言，历史信息中的其他出口可以记为0，表明不在特定的出口跳转；反之可记为1，表明在该特定的出口跳转。鉴于指令块存在多个出口(TRIPS指令集规定每个甚块最多8个出口)，该策略存在两种具体形式。

4.1　对8个出口分别使用O-GEHL二值预测的块预测器

第一种策略为对8个出口分别使用传统的O-GEHL二值分支预测器，如图3所示。如果有多个

出口预测跳转，选择加法树结果最大的那个出口作为最后预测的结果(意味着更大的置信度)；如果这8个分支预测器都预测不跳转，再使用备用的选择器从8个出口中选择一个作为最后的出口。具体细节如算法2所示，其中exit 0_pred～exit 7_pred是二值预测的结果；OGEHLBin是一个O-GEHL二值预测器；MaxThreshold用于比较哪个出口对应二值预测器中加法树的值最大；BackupPred是最后的选择器(使用全局历史信息的二级预测器)。

如果出口0～出口7的加法树出现相同的计算结果，将倾向于使用出口ID更小的预测结果。该优先机制是根据所有动态甚块最后提交的出口ID统计结果制定的，具体的比例分布如图4所示。出口ID越小，其所占的整体比例越大，说明大多数甚块倾向于从出口ID较小的出口跳转。

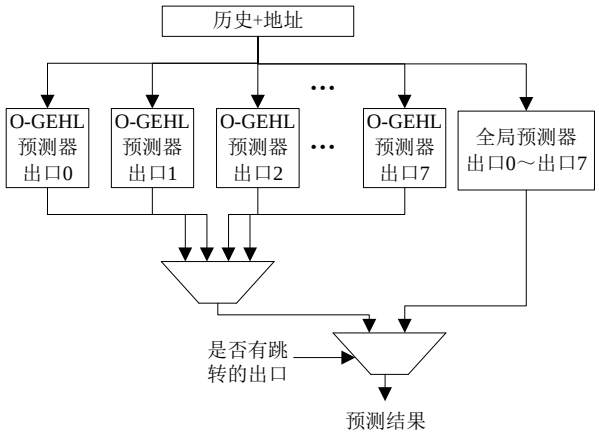

图 3 对 8 个出口分别使用 O-GEHL 二值预测的块预测器出口预测部分示意图

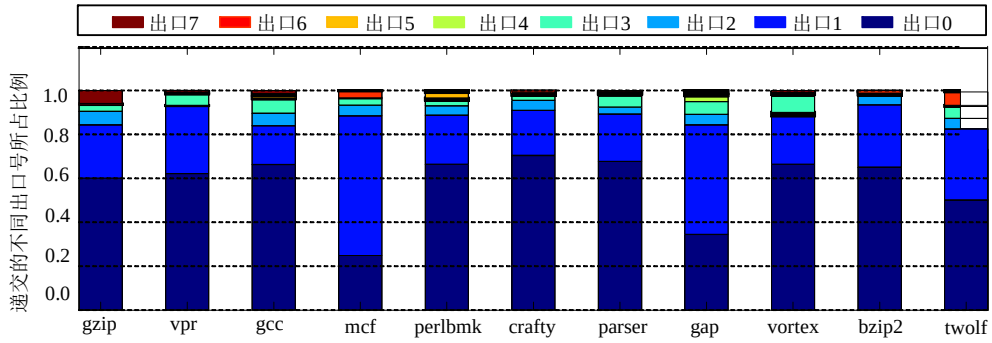

图 4 出口 ID 按比例分布示意图

算法2 对8个出口分别使用O-GEHL二值预测的算法。

输入：全局/本地历史(*H*)，块地址(Baddr)

输出：块出口(PredExit)

```
exit0_pred = OGEHLBin (H, BAddr);
……
exit7_pred = OGEHLBin (H, BAddr);
if (exit0_pred … or exit7_pred) then
   PredExit = MaxSum (exit0_pred, …, exit7_pred);
end
else
   PredExit = BackupPred(H, BAddr);
end
```

4.2 对前4个出口使用O-GEHL二值预测的块预测器

图 4 说明绝大部分甚块倾向于从出口 ID 较小的出口跳转，因此，对每个出口都分别使用传统的 O-GEHL 分支预测器会造成资源浪费。根据图 5 给出的分布，出口 ID 小于 4 的出口占据了绝大部分，所以，本文尝试了只对前 4 个出口使用传统的 O-GEHL 分支预测器的方案，如图 5 所示。

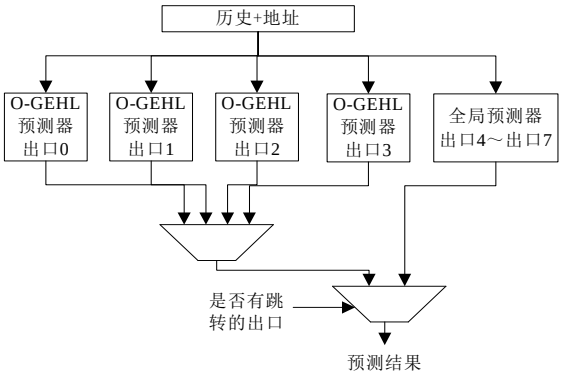

图 5 对前 4 个出口使用 O-GEHLc 二值预测的块预测器出口预测部分示意图

该策略的具体描述可见算法 3，其中大部分名词意义与算法 2 相同，只有 BackupPred4to7 是指在后 4 个出口中进行的选择。同样的，该策略在工作中，选择加法树结果最大的跳转出口为最后预测的结果；如果前 4 个出口均不跳转，就通过一个全局二级预测器从出口 4～出口 7 中选择最后预测结果。

算法3 对前4个出口使用O-GEHL二值预测的

出口预测算法。
输入：全局/本地历史(*H*)，块地址(Baddr)
输出：块出口(PredExit)

```
exit0_pred = OGEHLBin (H, BAddr);
……
exit3_pred = OGEHLBin (H, BAddr);

if (exit0_pred… or exit3_pred) then
   PredExit = MaxSum(exit0_pred, …, exit3_pred);
end
else
   PredExit = BackupPred4to7 (H, BAddr);
end
```

5　实验结果与评价

在进行性能评估时，本文首先复现了文献[4]中基于O-GEHL技术的块预测器，具体配置与文献中一致。本文评估了资源大小16 KB~1 MB时的情况，这也是当前微处理器中，能够为预测器分配的合理资源范围[7,15]。在资源更大或者更小的情况下对预测器性能进行评估的意义不大。

5.1　无选择器的O-GEHL块预测器的性能分析

本小节对第3节中提出的无选择器O-GEHL块预测器进行性能分析，图中纵坐标是11个整型测试程序MPKI的平均值，比较结果如图6所示。从图中可以看出，相比原文中的块预测器，无选择器的OGEHL块预测器性能更好。但遗憾的是，它的性能仅仅比原始的O-GEHL块预测器提高了大约0.7%。

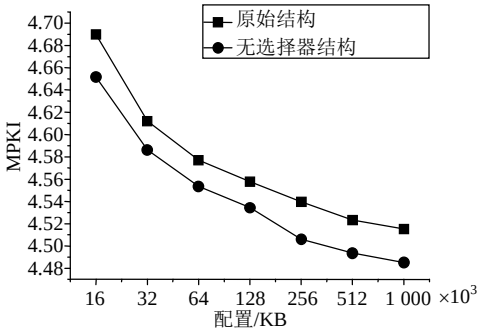

图6　文献[4]中基于O-GEHL预测技术的块预测器与无选择器的O-GEHL块预测器的性能(平均MPKI)比较

造成该策略比原有的策略性能略好的原因主要有两点：1) 使用正确预测次数而不是由选择器打破平局，该方法简单而有效；2）该策略在预测器相同配置的情况下，可以将选择器占用的资源节省下来，分配给成员预测器。由于成员预测器在整个预测过程对预测的准确性起着很关键的作用，因此分配更多的资源给成员预测器可以有效地提高预测准确率。性能提高不多可以理解为，该预测算法的本质与原始的预测算法是相同的。首先均使用多数投票法，然后使用一种备选机制打破多数投票造成的平局。因此多数投票法较低的准确率是制约采用该算法的预测器性能的主要因素。

5.2　利用O-GEHL二值预测的块预测器的性能分析

本小节对第4节中提出的利用O-GEHL二值预测能力的块预测器进行性能分析。除了使用8个、4个二值O-GEHL预测器，本文还对其他可能进行了探索，分别尝试使用5个和3个二值O-GEHL预测器。结果表明，性能没有使用4个二值O-GEHL预测器的好。

图7展示了文献[4]的O-GEHL块预测器与使用8个O-GEHL二值预测器和使用4个O-GEHL二值预测器的块预测器的性能比较，纵坐标为11个整型测试程序MPKI的平均值。图中，在较低配置下，使用4个二值O-GEHL预测器的性能要更优，而当配置超过128 KB时，情况正好相反。这种现象可以从这两种预测器的基本结构和出口特性的角度来解释。在相同配置的情况下，当使用8个二值O-GEHL预测器时，分配给每个二值分支预测器的资源要更少。因而在资源较少的情况下，由于每个二值分支预测器的性能受到资源的限制，整体的性能不如使用4个二值O-GEHL预测器的块预测器。

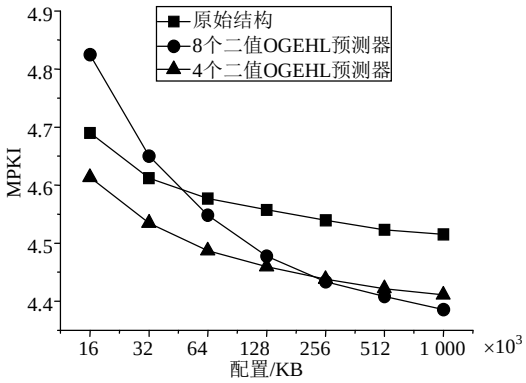

图7　使用O-GEHL二值预测器的块预测器与文献[4]中基于O-GEHL预测技术的块预测器的性能(平均MPKI)比较

另一方面，虽然对指令块出口的分布统计说明前四个出口占所有出口的绝大部分，但是对于部分程序(如gcc、mcf等)，出口4～出口7也占有相当的比例，因此，对每个出口都使用二值分支预测器可以使对这些程序的预测更准确。因此在资源较多的情况下，使用8个二值O-GEHL预测器可以使性能得到提高。

在原始O-GEHL块预测器与使用8个O-GEHL二值预测器的块预测器的性能比较中可以发现，在较

小的相同配置下，原文中的结构占优，但当资源配置超过32 KB时，改进后的块预测器性能更好，而且随着资源的增多，性能提高更多。在图中所显示的配置范围内，该改进的块预测器在最好的情况下可以将原始的MPKI提高大约3%。

在原始的O-GEHL块预测器与使用4个O-GEHL二值预测器的块预测器的性能比较中可以发现，在给出的配置范围内，改进的块预测器的性能总是好于原始块预测器的性能，其性能平均提高约2%。

6 结 论

在对现有的基于O-GEHL预测技术的块预测器的研究中本文发现其存在改进空间，因此本文提出了两种改进的基于O-GEHL预测技术的块出口预测器：一种是无选择器的出口预测方案；一种是直接利用O-GEHL二值预测的出口预测方案。

本文的实验结果表明在相同配置情况下，在资源从16 KB～1 MB的范围内，采用无选择器的出口预测方案的块预测器比之前基于O-GEHL预测技术的块预测器性能平均提高了0.7%；采用8个传统O-GEHL预测器的块预测器在最高配置(1MB)的情况下性能比原来提高了3%；对前4个出口采用传统O-GEHL预测器的块预测器则将性能平均提高了约2%。同时，后两种方案还有随着配置增加性能更好的趋势。

尽管本文只针对EDGE体系结构进行了研究，但本文所提出的两种改进策略同样可以扩展到其他基于甚块的、需要“*N*选1”预测器的体系结构中。

参 考 文 献

[1] BURGER D, KECKLER S W, Mckinley K S, et al. Scaling to the end of silicon with EDGE architecture[J]. IEEE Computer, 2004, 37(7): 44-55.

[2] CHANGKYU K, SETHUMADHAVAN S, GULATI D, et al. Composable lightweight processors[C]//Microarchitecture the 40th Annual ACM/IEEE International Symposium on. NW Washington, DC USA: IEEE Computer Society, 2007: 381-394.

[3] MALHLKE S, LIN D, CHEN W, et al. Effective compiler support for predicated execution using the hyperblock[C]// Microarchitecture the 25th Annual International Symposium on. Portland, Oregon USA: IEEE Computer Society, 1992: 45-54.

[4] RANGANATHAN N. Control flow speculation for distributed architectures[D]. Austin, TX USA: Department of Computer Sciences, University of Texas at Austin, 2009.

[5] RANGANATHAN N, BURGER D, KECKLER S W. Analysis of the TRIPS prototype block predictor[C]// Performance Analysis of Systems and Software IEEE International Symposium on. Boston, MA USA, 2009: 195-206.

[6] RANGANATHAN N, NAGARAJAN R, JIMENEZ D, et al. Combining hyperblocks and exit prediction to increase front-end bandwidth and performance[EB/OL]. [2011-06-28]. http://www.taco.cs.utsa.edu/pdfs/exit.pdf.

[7] SEZNEC A. Analysis of the O-GEometric history length branch predictor[C]//Computer Architecture the 32nd Annual International Symposium on. Madison, Wisconsin USA, 2005: 394-405.

[8] SANKARALINGAM K, NAGARAJAN R, MCDONALD R, et al. Distributed microarchitectural protocols in the Trips prototype processor[C]//Microarchitecture the 39th Annual ACM/IEEE International Symposium on. Orlando, FL USA: IEEE Computer Society, 2006: 480-491.

[9]S MITH A, GIBSON J, MAHER B, et al. Compiling for EDGE architectures[C]//Code Generation and Optimization International Symposium on. NW Washington, DC USA: IEEE Computer Society, 2006: 185-195.

[10] JACOBSON Q, BENNETT S, SHARMA N, et al. Control flow speculation in multiscalar processors[C]//International Symposium on High Performance Computer Architecture. San Antonio, Texas USA: [s.n.] 1997: 218-229.

[11] GOU P, LI Q, JIN Y, et al. M5 based edge architecture modeling[C]//Computer Design (ICCD) IEEE International Conference on. Netherlands: Amsterdam, 2010: 289- 296.

[12] HAO E, CHANG P, EVERS M, et al. Increasing the instruction fetch rate via block-structured instruction set architectures[C]//The 29th Annual IEEE/ ACM International Symposium on Microarchitecture. Paris, France: IEEE Computer Society, 1996: 191-200.

[13] JACOBSON Q, ROTENBERG E, SMITH J. Path-based nest trace prediction[C]//The 30th Annual IEEE/ACM International Symposium on Microarchitecture. Research Triangle Park, NC USA: IEEE Computer Society, 1997: 14-23.

[14] PATEL S, LUMETTA S. A hardware framework for dynamic optimization[J]. IEEE Transactions on Computers, 2001, 50(6): 590-608.

[15] SEZNEC A, MICHAUD P. A case for (partially) TAgged GEometric history length branch prediction[J/OL] [2011-06-28]. http://www.jilp.org/volb/v8paper1.pdf.

[16] SEZNEC A. Looking for limits in branch prediction with the GTL predictor[J/OL]. [2011-06-28]. http://www.irisafr/caps/people/seznec/GTL.pdf.

[17] SHERWOOD T, PERELMAN E, CALDER B. Basic block distribution analysis to find periodic behavior and simulation points in applications [C]//International Conference on Parallel Architectures and Compilation Techniques. Barcelona, Spain: [s.n.], 2001: 3-14.

编 辑 张 俊