CN81-1166/TP
ISSN 1007-130X

计算机工程与科学
COMPUTER ENGINEERING & SCIENCE

2001年第23卷第1期
Vol. 23, No. 1, 2001

文章编号：1007-130X(2001)01-0039-05

Trace Cache 及 Trace 处理器技术*
Trace Cache and Trace Processor Technology

杜贵然，罗金平，徐　明，胡瀚涛，周兴铭
DU Gui-ran, LUO Jin-ping, XU Ming, HU Han-tao, ZHOU Xing-ming
（国防科技大学计算机学院，湖南 长沙　410073）
(School of Computer Science, National University of Defense Technology, Changsha 410073, China)

摘　要：Trace Cache 和 Trace 处理器着力解决取指令的带宽，是一种颇具潜力的技术。本文在介绍 Trace Cache 技术的基础上，结合 ILP 研究的现状，提出了未来 Trace 相关技术的研究方向。

Abstract: Trace Cache and trace processor can achieve high bandwidth of instruction fetch by exploiting instruction reuse. This paper first introduces the trace cache and trace processor structures together with the current ILP study, gives out the trends of trace-processing technology.

关键词：踪迹缓存；踪迹处理器；体系结构；指令级并行

Key words: trace Cache; trace processor; architecture; ILP

中图分类号：TP303　　**文献标识码**：A

1　引言

随着半导体工艺技术的不断革新，CMOS 工艺的特征尺寸日渐缩小，处理器主频迅速提高；存储系统的发展相对滞后，微处理器执行单元与存储层次之间的壁垒越来越大。这一切对微处理器的结构设计提出了新的挑战。

CMOS 技术的发展呈现脱离线性伸缩区的趋势，延迟和互连已成为设计微处理器的首要问题。这种发展趋势使得相同面积的芯片上有了更多可用的晶体管，改变了晶体管数目受限条件下的体系结构设计状况。

为了充分利用半导体技术的成就，满足对处理器性能的更高要求，各大处理器制造商和各大学竞相提出了各自的解决方案。

当前商用的主流超标量微处理器如 Alpha、PA-RISC、PowerPC、UltraSPARC、MIPS 等，为了达到更高性能，在体系结构上进行了以下几方面的改进：(1) 开发指令级并行性、线程级并行性、处理器级并行性；(2) 提高主频；(3) 增大 Cache 的容量和层次等。

大学的研究方案与商业处理器的略有不同，不太考虑产品的兼容性。他们提出的概念模型有：单芯片多处理器(SCMP)、同时多线程(SMT)、向量 IRAM 处理器、宽发射超标量处理器、RAW 结构及 Trace 处理器等，其中一些模型大胆使用了 Trace Cache 结构。Wisconsin-Madison 大学和 Indian 科学院基于 Trace Cache 结构分别提出了 Trace 处理器的概念。

2　Trace Cache 技术

Trace 是指程序执行的动态指令流，是局部

* 收稿日期：2000-01-19；修订日期：2000-05-01
基金项目："九五"国防预研项目资助
作者简介：杜贵然(1968—)，男，博士生，研究方向为先进计算机系统结构、微处理器、指令级并行性和光互连；罗金平，博士生，研究方向为计算机系统结构、光互连和指令级并行性；徐明，副教授；周兴铭，中国科学院院士，教授，博士生导师。
通讯地址：410073 湖南省长沙市国防科技大学计算机学院博士生队；Tel：(0731)4515284(H)
Address: Doctoral Brigade, School of Computer and Science, National University of Defense Technology, Changsha, Hunan 410073, P. R. China

的指令轨迹。Trace Cache 捕获并记录指令执行的动态序列，是一种特殊结构的 Cache。它记录指令执行的逻辑序列，而非指令的存储顺序，它将物理上不连续但逻辑上连续的指令保存在一个 Cache 块中。Trace Cache 的基本构成单元是 Trace，由于 Cache 块的长度限制，这里的每条 Trace 最多只记录 n 条指令，或者是 m 个基本块（两条控制转移指令间的指令）。它的起点（Trace 入口）可以是动态指令流的任意点。n 是 Cache 行的长度，m 是分支预测的最大输出。这样，一条 Trace 可由起始地址和最多 m－1 个分支输出来唯一确定。

2.1 Trace Cache 的工作原理

当前主流的超标量技术一般都采用了分支预测和 Cache 结构，但分支预测与指令 Cache 是两个相互独立的部件。事实上，指令的执行与分支预测是密不可分的，Trace Cache 技术把两者有机地结合在一起。

Trace Cache 的工作过程是：当首次遇到一条指令轨迹（Trace）时，就在 Trace Cache 中为之分配一个 Cache 行。如果程序继续执行中再一次遇到一条已缓存的 Trace，便直接从 Trace Cache 中提取，而无需从指令 Cache 中分块调入。

Trace Cache 的核心思想是将指令流（物理上）不连续的块转变为连续的块，从而加快指令的预取，为超标量的指令译码提供更大的指令流量，减少存储层次的带宽压力，加快译码的速度。图 1 是 Trace Cache 的工作原理示意图。

从图中可以看出，当处理器有指令请求时，地址信息同时发送到指令 Cache 和 Trace Cache。指令 Cache 如命中会取出一个 Cache 块，作二选一选择器的一路输入；Trace Cache 接收到地址后，结合分支指令的预测：若命中，也从 Trace Cache 中取出一条 Trace，作二选一选择器的另一路输入。选择器从两个输入中选择其一提供给译码单元，选择器的输出也作为 Trace Cache 填充单元的输入。填充单元的主要功能就是将处理器执行过的指令经组织后送到 Trace Cache 缓存。

由于 Trace Cache 是在指令 Cache 的基础上结合了分支预测，故比单纯的指令 Cache 复杂，

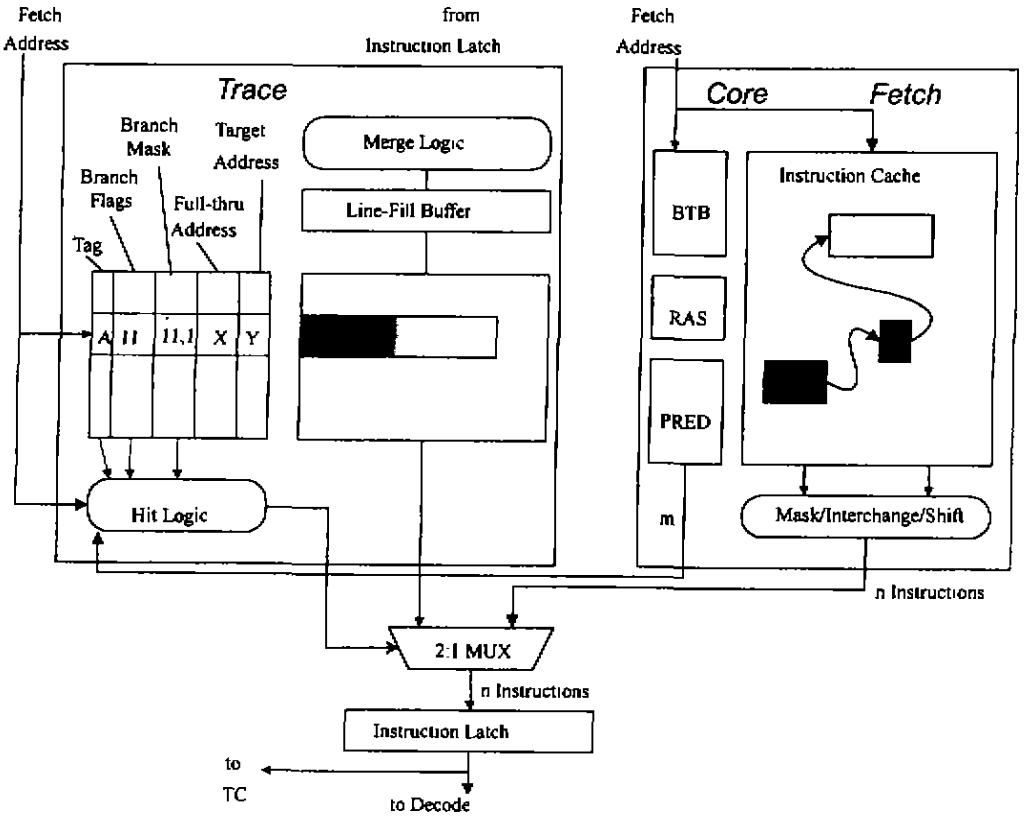

图 1 Trace Cache 工作原理示意图

功能更强大。因此，在实现了 Trace Cache 的结构上就有两种选择：以指令 Cache 为中心或以 Trace Cache 为中心。前者以指令 Cache 为主，Trace Cache 是指令Cache 的支撑；后者在结构上突破了 Cache 的现有结构，把指令 Cache 作为 Trace Cache 的后备。

Rotenberg 的研究表明，经 IBS(Instruction Benchmark Suite)和 SPEC92 整数测试程序的测试，增加 4KB 容量的 Trace Cache 比传统的顺序取指要快 28%。

2.2 下一条 Trace 预测技术

下一条 Trace 预测技术是以 Trace 为基本单元，在 Trace 执行中记录其历史信息，用 Trace 的首条指令地址和对 Trace 中分支的预测结果作为索引来预测下一条被执行的 Trace。它是基于路径的高层控制流预测，能单周期预测多个分支。而在无 Trace Cache 的结构中，为了满足超标量处理器对宽指令发射的要求，提出了基于路径的单周期多分支预测，以达到对指令的预取要求。

分支预测的某些技术可用于下一条 Trace 预测。如用最近最常用的 Trace 构造一个小索引表，做成复合预测器，并针对调用指令，做一个返回历史栈，当执行对应的返回指令时，恢复相应的状态（历史），从而提高预测的精度。Jacobson 的研究证明，采用下一条 Trace 预测技术可减少 26%的错误预测。

2.3 Trace Cache 与指令 Cache 的区别

总的来说，这两种结构都是向处理单元提供指令，其区别在于：

（1）结构不同。两种结构的 Cache 单元和单元标识不同。

（2）功用上有主次关系。指令 Cache 是以取指为主要目标，而 Trace Cache 用来提高前者的命中率并减小存储带宽。不过，最新的 Trace Cache 思想已经突破了这点，已经取代了指令 Cache 的功能。

（3）实现的速度不同。由于功用不同，Trace Cache 和指令 Cache 的实现不要求相同的指标，处在主要位置的要快一些，容量小一些；处在次要位置的容量可以大一些，速度可以慢一些。

（4）处理呈依赖关系，Trace Cache 依赖于指令 Cache。

2.4 Trace Cache 的实例

在 Microprocessor Forum'99 会议上，Fujitsu 的子公司 HAL 公布了 SPARC64 V 的结构。图 2 是 SPARC64 V 的功能块示意图，其设计采用了 Trace Cache。从 I-Cache 中取出的指令经预译码后打包，填入容量为 1024 项的四路组关联 Cache（即 Trace Cache）中，每个包最多包括两个基本块。一级 I-Cache 只是作为 Trace Cache 的后备。PC 要求的指令块并行地从 Trace Cache 和 I-Cache 中读出。另设 Alternate Path Buffering (APB)机制保存来自非预测分支地址的第一个包，减少错误预测的代价。

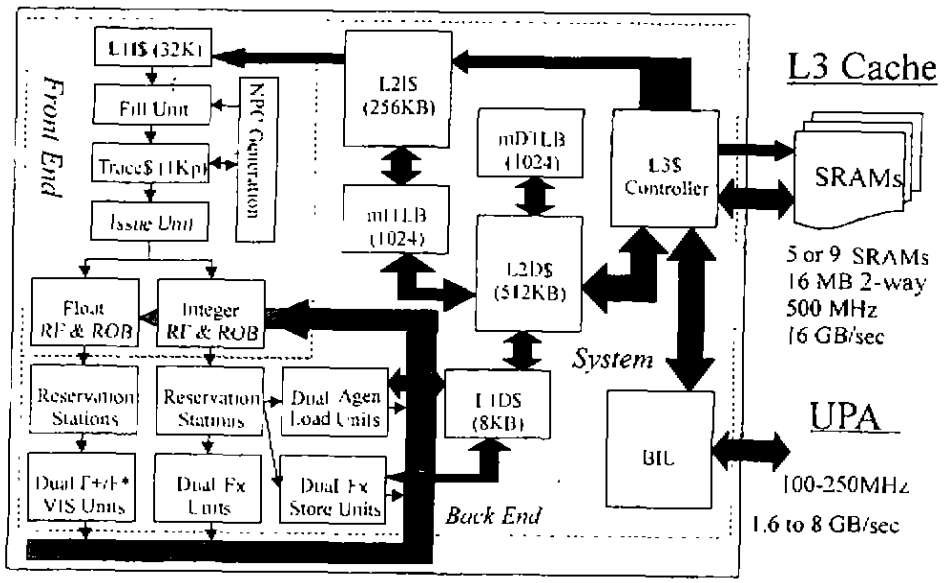

图 2 HAL 公司 SPARC64 V 结构示意图

3 Trace 处理器

下一代高档微处理器（超标量）无一例外都增加了每个时钟周期的指令发射数目，一般都达到了八路定点指令发射。多路发射带来的设计复杂性使之不能完全发挥处理单元的处理能力，不能从顺序执行的程序中提取更多的可并行执行的指令。超标量的结构局限导致了这种结果。

在 Trace Cache 的基础上，由 Eric Rotenberg、James E. Smith(Wisconsin-Madison 大学)及 Sriram Vajapeyam(Indian 科学院)等人又分别提出了 Trace 处理器的概念。Trace 处理器将程序的执行看作是对 Trace 的执行。它不同于传统的处理器以指令为基本的执行单元。Trace 处理器围绕 Trace 工作。一条 Trace 可有一定数目的控制转移指令，也即隐含了一定数目的控制预测，这就使预测不仅仅针对单个的控制转移指令，而是以 Trace 为单元，从而有效地忽略了一条 Trace 包含的内在控制流程。此外，执行一条 Trace 时将其使用的寄存器分为输入寄存器、局部寄存器和输出寄存器，便于将寄存器文件作层次划分，进而减少了处理单元之间的数据传输。图 3 是 Rotenberg 等人提出的 Trace processor 结构设想。

Rotenberg 提出的 Trace processor 由处理单元(PE)构成，每个 PE 是一个小规模的超标量处理器。PE 包括能容纳一条完整 Trace 的指令缓冲空间、多路专用的功能单元、保存局部结果的专用局部寄存器文件、全局寄存器文件的一个拷贝。处理器前端的预处理以 Trace 为中心。

Vajapeyam 的设计更进一步，在 Trace processor 中加入了向量化方式，处理 Trace 的层次更为突出。使重复比较显著的 Trace（如循环）像向量处理机处理向量运算一样运行。该处理器模型有三个层次：底层处理单条指令，中间层处理多条指令构成的基本块（或多个基本块构成的 Trace），最上一层处理多个 Trace 构成的向量化单元，每个 PE 基本上是超标量处理器。为了适应向量化的处理，更强调把 Trace 分布到各个 PE，不同于前一结构的 Trace 所采用的集中式结构。

这两个 Trace 处理器模型最底层的结构大致一样，都采用了当前主流的超标量结构，二进制兼容是其主要设计思想。Vajapeyam 宣称他的模型可以比当前的超标量结构在速度上快 2～3 倍，能这样快速运行的程序大概也是循环为主的计算。

4 Trace 处理器的研究方向

Trace 记录指令执行的动态顺序，隐含对分支指令的预测。Trace Cache 保存指令的执行轨迹，能单周期提供多个指令基本块，较好地解决了超标量处理器对取指令带宽的要求。Trace 处

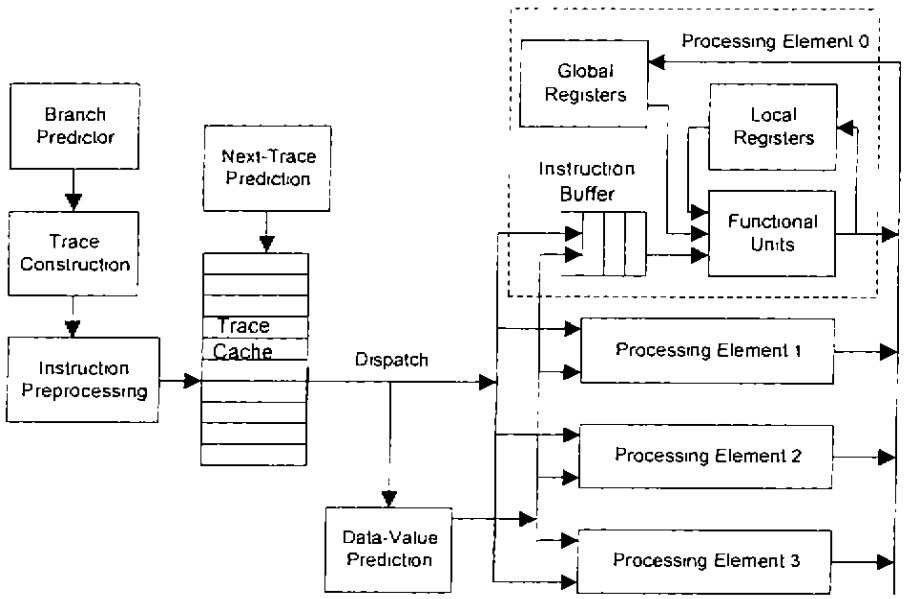

图 3 Rotenberg 等设计的 Trace processor

理器以 Trace 为基本执行单元，降低了指令和数据的相关性冲突，充分利用了多处理单元的执行效率，加快了指令的处理速度，是一种很有前途的技术。研究表明，采用 Trace Cache 技术能提高处理速度达 26%，预测采用 Trace 处理器技术能提高性能达 2～3 倍。通过对现有成果的分析和研究，我们提出如下研究方向：

(1) Trace 的预处理。要使处理器达到合理的指令执行速度，存储层次必须尽快向执行单元提供指令。对于多 PE 的处理方式，预取更要先行。若考虑处理核心与 Cache 间越来越大的延迟(对时钟周期而言)，预取就更显得必要。Trace 的预构以大 Cache 为基础，可利用预执行框架信息、编译时的提示或运行时的线索等来构造。

(2) Trace Cache 的层次。一级 Cache 的容量总是有限的，要保证处理器能尽快得到 Trace，只一级 Cache 是不够的。若要考虑 Trace Cache 的替换，还可将清除的 Trace 作缓冲，供处理器随时再用。

(3) 基于 Trace 的数据预取技术。基于 Trace 的数据预取不同于基于运行的数据预取，也不同于基于 Cache 失效的数据预取，更容易取得目标，提高命中的准确率和预取效率。

(4) Trace 的相关性预测和处理。这与分支预测、Trace 预测、前瞻技术都有密不可分的联系，有助于提高 Cache 的效率，能提高与 Trace 处理器相关的执行效率。

(5) Trace Cache 的效率问题。由于 Trace Cache 内在的存储空间利用不充分（有冗余和零头），造成 Cache 资源浪费，所以对 Trace Cache 有必要建立新规则和扩充的方式。

(6) Trace 与多线程相结合。Trace 能够有效减少存储设备的带宽要求，加快指令预取的速度，但遇到相关无法消解的情况时，仍不能减少 CPU 的等待周期。多线程正好可以弥补延迟等待的空周期。

5 结束语

Trace Cache 将逻辑上连续但物理上分离的指令块转变为物理上连续的单元存储，从而增大了指令读取的带宽，实现了单周期读取多块的目的；结合“下一条 Trace 预测”技术，能够及时有效地向 CPU 提供所需的指令，实现了分支预测和指令缓存的有机结合。以此为基础提出的 Trace 处理器，是以 Trace 为基本处理单元，根据运行历史记录确定 Trace 的重用，进而提出了向量化处理设想，是当前超标量微处理器在保持兼容前提下的改进结构，模拟结果显示性能有显著提高。

参考文献：

[1] Wiel S P V, Lilja D J. When Caches Aren't Enough: Data Prefetching Techniques[J]. IEEE Computer, July 1997, 23～30

[2] Hsu Wei-Chung, Smith J E. A Performance Study of Instruction Cache Prefetching Methods[J]. IEEE Trans on Computers, 1998, 47(5): 497～508

[3] Jacobson Q, Rotenberg E, Smith J E. Path-Based Next Trace Prediction [A]. 30th Int'l Symp on Microarchitecture[C]. 1997. 14～23

[4] Reinman G, Calder B. Predictive Techniques for Aggressive Load Speculation[A]. 31st Int'l Symp on Microarchitecture [C], 1998. 127～137

[5] Rotenberg E, Bennett S, Smith J E. Trace Caches: A Low Latencey Approach to High Bandwidth Instruction Fetching[A]. 29th Int'l Symp on Microarchitecture, 1996. 24～35

[6] Rotengerg E, Jacobson Q, Smith J E. Trace Processors[A]. 31st Int'l Symp on Microarchitecture[C], 1997. 138～148

[7] Semiconductor Industry Association. The National Technology Roadmap for Semicooductors' 1997[EB/OL]. http://www.sia.org/, 1997

[8] Shebanow M C. SPARC64 V, A High Performance System Processor[A]. Microprocessor Forum' 99[C], 1999

[9] Uht A K, Sindagi V, Somanathan S. Branch Effect Reduction Techniques[J]. IEEE Computer, May 1997, 71～78

[10] Vajapeyam S, Joseph P J, Mitra T. Dynamic Vectorization: A Mechanism for Exploiting Far-Flung ILP in Ordinary Programs[A]. ISCA' 99[C], 1999. 19～27

[11] Vajapeyam S, Mitra T. Improving Superscalar Instruction Dispatch and Issue by Exploiting Dynamic Code Sequences [A]. ISCA' 97[C], 1997. 1～12