The Microarchitecture of the Pentium 4 Processor
论文翻译

2014211304 班

D 组

史文翰 2014211218

林宇辰 2014211223

王剑督 2014211228

崔嘉伟 2014211233

杨　莹 2014211238

徐丹雅 2014211243

郝绍明 2014210123

分工情况

史文翰：ABSTRACT/INTRODUCTION/OVERVIEW OF THE NETBURST? MICROARCHITECTURE

林宇辰：CLOCK RATES

王剑督：NETBURST MICROARCHITECTURE 到 Out-of-Order Execution Logic（page6） 之前

崔嘉伟：Out-of-Order Execution Logic 到 page7 末尾

杨　莹:Integer and Floating-Point Execution Units (page8) 到 Store-to-Load Forwarding（page9）之前

徐丹雅：Store-to-Load Forwarding（page9）到 Memory Subsystem（page11）之前

郝绍明：剩余部分

概述

这篇论文描述了 Intel 新的旗舰处理器——奔腾 4 处理器的 Intel NetBurst 微架构。这个微架构是以奔腾 4 为起点的处理器家族的基础，奔腾 4 处理器为许多关键应用领域提供了实质上的性能支持，这使得终端用户可以真正地体会到它与其他处理器的不同。

在这片论文里我们描述了 NetBurst 微架构的主要特点和功能，我们展示了器械的前端，包括了它新的被称为“执行追踪缓存”的指令缓存。我们也描述了乱序执行引擎，包括使用在 3GHz 下的双输出 ALU 的延迟极低的特性。我们也讨论了存储子系统，包括了有着极低延迟的一级缓存，访问它只需要两个时钟周期。随后，我们讨论了一些关键特性，这些关键特性使得奔腾 4 处理器拥有出色的浮点处理能力和多媒体性能。我们提供了奔腾 4 的一些关键性能参数，用以对照奔腾 3 处理器。

介绍

奔腾 4 处理器是 Intel 新的旗舰微内核，它最初在 2000 年 11 月被设计为具有 1.5GHz 的主频。他有着新一代 Intel NetBurst 微架构，这种架构有着极高频率的时钟和世界一流的性能。它包括了一些新的特性和创新，这使得奔腾 4 处理器在工业界的性能会在几年之内处于领先的地位。这篇论文提供了对 NetBurst 微架构的特性和功能的深入剖析。

奔腾 4 的设计宗旨是让用户切实感受到它的性能强大。例如，它在网络音频、流视频、图像处理、视频内容构造、语音识别、3D 应用、游戏、多媒体、多任务处理等方面提供了更好的用户体验。奔腾 4 处理器允许实时编码 MPEG2 视频和几乎实时编码 MPEG4 视频，允许高效的视频编辑和视频会议。他在 3D 游戏应用方面（如《雷神之锤 3》）具有世界一流的性能，使得 3D 应用的现实感和虚拟场景达到了新的高度。

奔腾 4 处理器拥有基于 Intel0.18 微米 CMOS 工艺的 4 千 2 百万多个晶体管，拥有六层互联铝架构层。它的金属模具仅为 217mm², 在 1.5GHz 的工作频率下，它的功率仅为 55W。它具有 3.2GB/s 的系统总线，为那些如今和未来对于数据速率要求高的应用提供了高带宽。它加入了 144 个新的 128 位 SIMD 指令，被称为 SSE2，它可以提高多媒体、内容构造、科学或工程应用的性能。

NetBurst 微架构概述

一个高速处理器需要平衡和协调许多微内核特性，这些特性影响处理器的尺寸、设计效率和可用性。图 1 展现了奔腾 4 处理器的 NetBurst 架构的基础结构。如图所示，主要有四个部分：顺序前端、乱序执行引擎、整数及浮点执行单元、存储子系统。

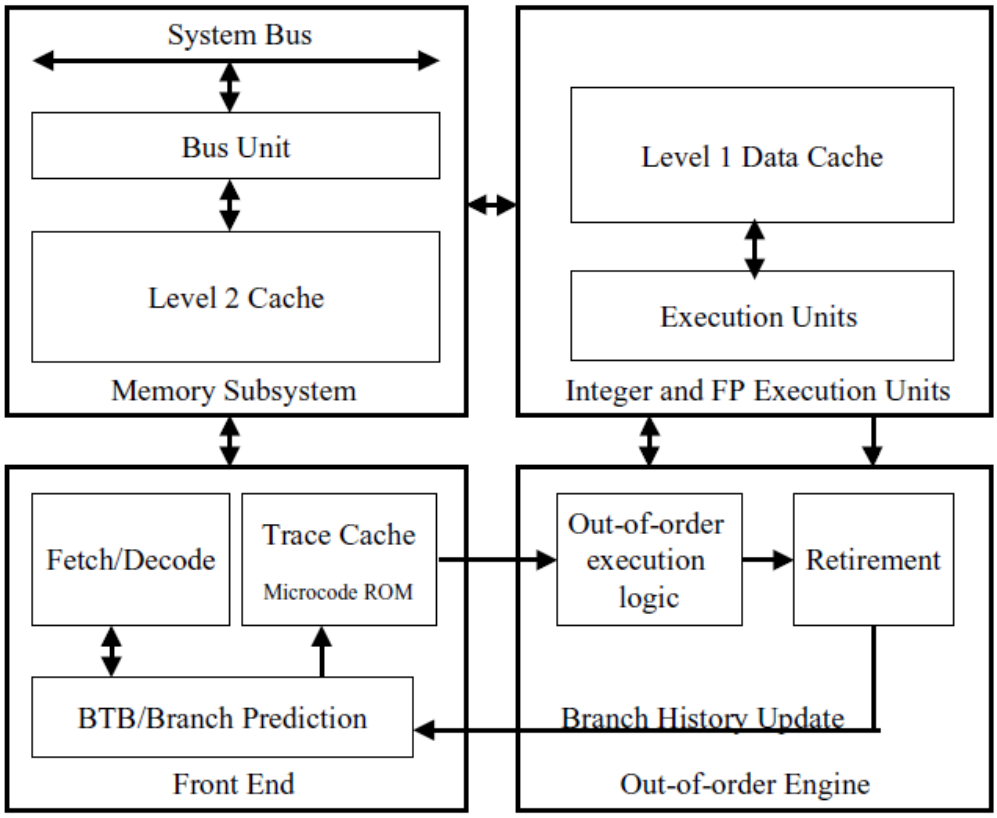

Figure 1: Basic block diagram

顺序前端

顺序前端是功能部件的一部分，该功能部件负责抓取程序中接下来将要执行的指令，并把它们保存在之后将要被使用的管道中。顺序前端的任务是提供一个高带宽的被解码的指令流，并使它们流向乱序执行单元，乱序执行单元中，这些指令才会真正地被执行。前端拥有高准确率的分支预测逻辑，它通过之前执行过的程序来预测接下来的程序的走向。从前端的分支预测逻辑被预测出的指令地址被用来抓取在 L2 缓存的指令字节。之后，这些 IA-32 指令字节将被解码成被称为微指令的基本操作单元，这些微指令会被执行单元执行。

NetBurst 微架构有先进的 L1 指令缓存，被称为“执行追踪缓存”。与传统的指令缓存不同，追踪缓存介于指令译码逻辑和执行单元之间。在这个位置，追踪缓存可以存储已经被译码的 IA-32 指令或者微指令。存储已经译码的指令使得 IA-32 译码过程从执行循环中被剔除。典型的指令被被译码一次，被存储在追踪缓存，之后被反复使用，就像在之前机器上的普通指令缓存一样。IA-32 指令译码器仅在追踪缓存没有被命中，并需要去读取 L2 缓存以获得新的 IA-32 指令字节码时才被使用。

乱序执行逻辑

乱序执行引擎是指令准备被执行的地方。乱序执行逻辑拥有几个缓冲区，这些缓冲区被用来平滑和重排指令流，使得指令流在流过队列并被执行时得到优化。为了使指令在输入的操作数就绪后快速执行，指令的重排现象会经常发生。这种乱序执行使得程序中的指令允许那些被延迟的指令在它们的前后被执行，只要被延迟的指令不依赖于这些程序中的指令。乱序执行允许像 ALU 或者缓存这样的功能部件保持尽可能的忙碌，去执行那些准备就绪的独

立指令。

退化逻辑负责重排指令，使那些乱序发射的指令还原回原顺序。这种退化逻辑可以从执行单元得到被执行的指令的完整的状态信息，并处理执行结果，以便获得与程序逻辑相对应的正确的结构状态。奔腾 4 处理器可以在一个时钟周期内还原 3 个微指令的顺序。这种退化逻辑保证了如果异常发生，那么异常只能是由最老的、没有被退化的指令操作引起的。这种逻辑也向前端的分值预测逻辑提供了分支预测信息，使它们拥有最新的、被证明可靠的分支历史信息。

整数和浮点执行单元

执行单元是指令真正被执行的地方。这个部分包括了寄存器堆，存储了整数和浮点数操作数，这些操作数需要被指令执行。执行单元包括了负责计算结果的几种整数和浮点执行单元和 L1 缓存，L1 缓存在大多数的 Load 和 Store 指令中会被用到。

存储子系统

图 1 也表示出了存储子系统。它包括了 L2 缓存和系统总线。L2 缓存用于存储指令和数据，它们无法被执行追踪缓存和 L1 缓存所命中。外部系统总线与后部的第二层缓存相连，被用来在 L2 缓存未命中时访问内存，同时也被用来做系统级的 IO。

时钟频率

处理器的微体系结构可以被分成不同的层级。管线架构的层级是一种微型架构。对于一个特定处理器，在硅加工工艺给定不变的前提下，它的最终工作频率取决于管线架构的作用程度。在设计一个新的处理器时，核心决策的目标便是设计它的操作频率。目标的工作频率决定了在设计中每条管线中有多少个逻辑门。这进一步地帮助人们确定了该机器有多少种工作状态。

为了设计出更高的时钟频率需要作出许多折中权衡。欲达到更高的时钟频率需要更深的管线架构，所以相同时钟频率的工作效率会随之下降。流水线变长会使得工作更繁多、花费的时钟周期更多，例如缺页中断和不可预测分歧的出现，但通常我们为了补偿变低的单个时钟工作效率，允许设计并运行在一个更高的时钟频率上。比如说，频率翻升 0.5 倍可能虽然仅仅会使得整个网络的表现改良 0.3 倍，但是这个频率的增加仍然会显著改善整体表现。设计一个高的频率十分依赖于电路设计技术、设计方法论、设计工具、硅加工工艺、能源和热力学的限制等等。在高频率的工作中，时钟会发生倾斜和抖动，时钟周期中的锁存器延时比例会更高，理论上减小了可用时钟周期的比例。深层的管线架构使得机器更加复杂，需要有更大的缓冲区来承载这些长流水线。

处理器工作频率的历史发展趋势

图 2 展示了最近六款因特尔处理器时钟的相对频率。竖轴代表了相对频率的大小，横轴区分出了六个彼此型号存在相对差异的六款处理器。

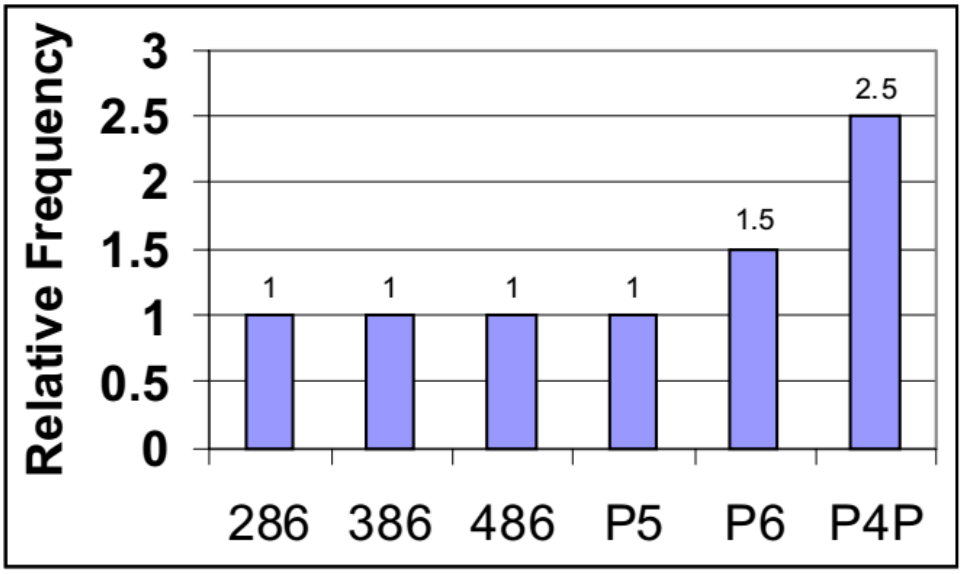

图 2：处理器时钟的相对频率

由图 2 可知，286、Intel386™，Intel486™和奔腾 5 处理器拥有相似的管线深度，也就是说这四款处理器如果以相同的硅加工工艺产出会以相似的时钟频率工作。他们的每个时钟周期都包含相似个数的逻辑门。P6 微架构处理器延伸了管线的深度，每条管线的状态表达需要的逻辑门数量更少，从而非常显著的表现出了极高的效率。与之前的处理器相比，P6 微处理器大约使管线阶段的数量翻倍，在时间上，实现了同等科技手段下，1.5 倍效率的提升。

NetBurst 微架构旨在拥有比以往更深的管线(是 P6 微架构的两倍)，它每个时钟周期中参与工作的逻辑门更少，使得它的时钟频率拥有足够资本领跑业界。跟 P6 家族的处理器相比，奔腾 4 处理器致力于在同等加工手段下将主时钟频率提高至原来的 1.6 倍。这使得它在相同的硅加工工艺下比 P6 家族处理器的操作更加迅捷。它在 2000 年 11 月的发布引言中声明处理器的工作频率是奔腾 3 的 1.5 倍。随着时间的推移，奔腾 4 处理器的设计愈发成熟，频率差距会越来越大。

奔腾 4 处理器的不同组件其实是以不同的时钟频率工作的。每个逻辑节段的工作频率被准确地设置为其实现功能所需的精准值。最高频率的节段的设置值等于经过算术逻辑单元操作环的速度临界值，它适用于一个完整程序的大多数指令。芯片其余的大部分组件大都以 3GHz 频率的一半来运行，因为这样做可以减小设计的难度。芯片的少部分节段以最高频率的四分之一来运行，目的同样是为了简化设计难度。为了匹配系统总线的需求，逻辑总线的工作频率为 100MHz。

图 3 是用来彰显管线差异的实例，它分别在 P6 与奔腾 4 处理器上选择了一条主要流水线，即分支误预测流水线。这条流水线包含了一个处理器“从一个方向错误、与流水线起始时预测的、上次读取硬盘地址不同的”分支恢复需要花费的周期数。如图所示，奔腾 4 处理器拥有 20 阶的分支误预测流水线，而 P6 微架构只有 10 阶的分支误预测流水线。将这些管线分解成小段后，管线的每个阶都会做更少的工作，从而时钟频率可以提高很多。

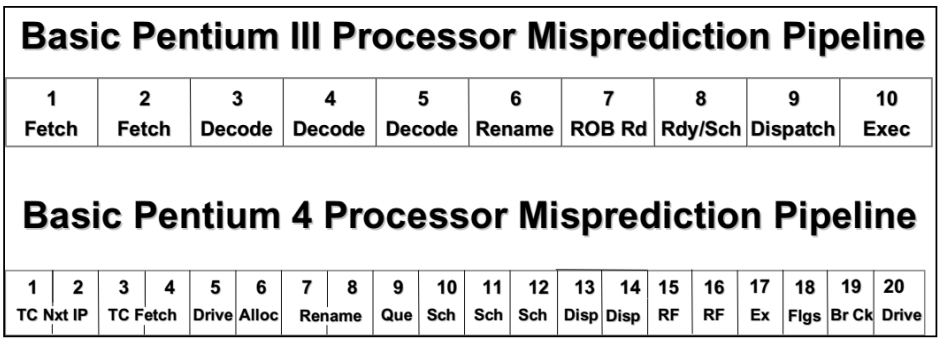

图 3：分支误预测管线

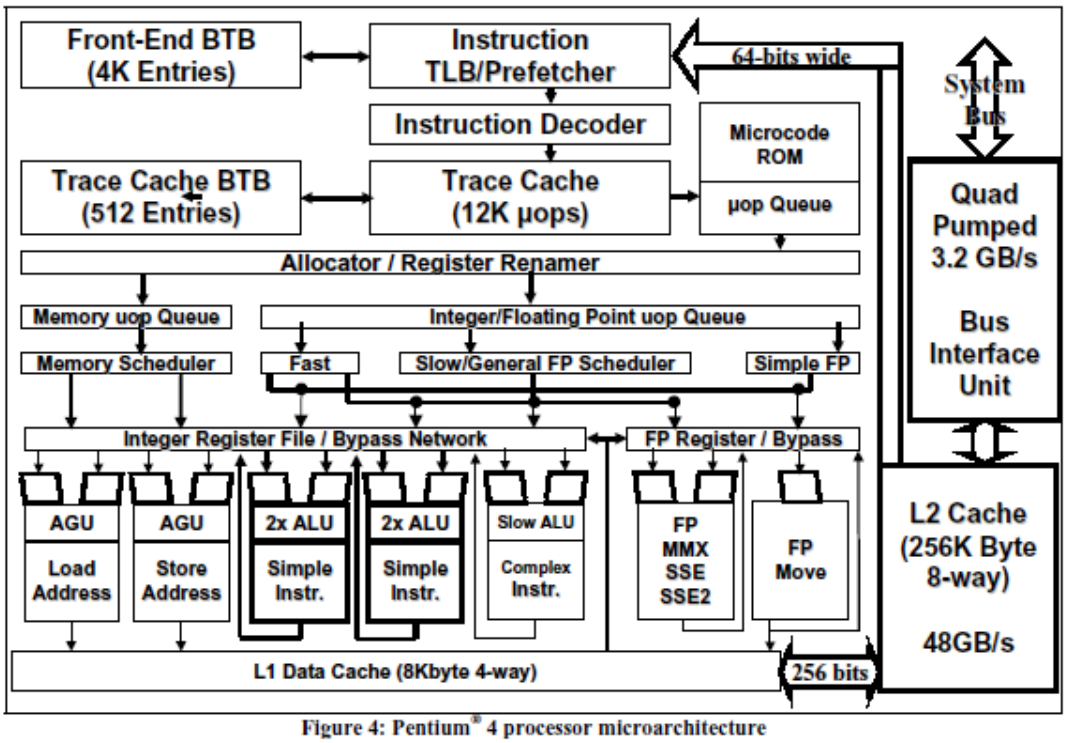

Figure 4: Pentium® 4 processor microarchitecture

NetBurst 微体系结构

图 4 显示了一个更详细的奔腾 4 处理器的 NetBurst 微体系结构框图。图的左上角显示了机器的前端。图的中间说明了无序缓冲逻辑，图的底部显示了整数和浮点执行单元和 L1 数据缓存。图的右边是内存子系统。

前端

奔腾 4 处理器的前端由几个单元组成，如图 4 的上部所示。它包含指令 TLB（ITLB），前端分支预测器（标记为前端 BTB），IA-32 指令译码器，追踪缓存，和微码 ROM.

追踪缓存

追踪缓存是奔腾 4 处理器的初级或 1 级（L1）指令缓存，每个时钟周期给无序执行逻辑提供高达三条内部指令。程序中的大部分指令是从追踪缓存中提取并执行的。只有当有一个追踪缓存丢失时 NetBurst 微体系结构才会从第 2 级（L2）缓存中读取和解码指令。这种情况发生的频率和前面的处理器错过了其 L1 指令缓存的频率一样高。追踪缓存有着容纳 12 条

内部指令的容量。它有一个类似为 8K 到 16K 字节的常规指令高速缓存的命中率。

IA-32 指令的解码过程很繁琐。指令有可变数目的字节，并且有许多不同的选项。指令译码逻辑需要解决这些问题，把这些复杂的指令转化为简单的机器知道如何执行的内部指令。这个解码是特别困难，因为要尝试在 Pentium 4 处理器的高时钟频率下运行时，每个时钟周期解码多个 IA-32 指令。能够在每个时钟周期解码多条指令的高带宽 IA-32 解码器，需要几个管道阶段来完成工作。当一个分支预测错误，如果该机器不必重新解码在修正的分支目标位置恢复执行所需的 IA-32 指令，恢复时间要短得多。通过缓存先前在追踪缓存中解码的内部指令，NetBurst 微体系结构的指令解码器绕过大部分时间从而降低预测错误潜伏期和允许解码器被简化：它每时钟周期只需要解码一条 IA-32 指令。

追踪缓存从 IA-32 指令解码器获取已经解码的内部指令并进行汇编或者将它们构建为内部指令程序有序的序列的执行过程称为痕迹。它为每条痕迹线分配六条内部指令。在一个单一的痕迹中可以有许多迹线。这些痕迹包括操作顺序下的 IA-32 程序执行路径运行的预测。这允许将一个分支的目标包含在同一个分支中。这允许分支的目标被包括在与分支本身相同的跟踪高速缓存行中，即使分支及其目标指令在程序中有成千上万的字节。

传统的指令缓存通常提供很多指令，包括一个分支指令，但在那个时钟周期后就一个指令都没有了。如果分支是高速缓存行中的第一条指令，则只有单分支指令被传送到时钟周期。传统的指令高速缓存也经常增加一个时钟延迟到达所采取的分支的目标，因为延迟通过分支预测器然后访问指令缓存中的新位置。追踪缓存为适合追踪缓存的程序避免了这两个方面的指令传递延迟。

追踪缓存有自己的分支预测器，指示在跟踪缓存中需要取下指令的位置。这个追踪缓存（图 4 的标记痕迹 BTB）小于前端的预测，因为它的主要目的是预测目前在追踪缓存中的程序子集中的分支。分支预测逻辑包括 16 个返回地址堆栈，以便有效地预测返回地址，因为相同的过程通常从多个不同的调用站点调用。跟踪缓存 BTB，以及前端 BTB，使用先进的分支预测算法，与 P6 微架构中的预测值相比，将分支错误预测率降低了约 1/3。

微码 ROM

在追踪缓存附近是微码 ROM。这个 ROM 是用于复杂的 IA-32 指令，如串动，针对故障和中断处理。当遇到复杂的指令时，追踪缓存跳进微码 ROM，然后 ROM 发出完成操作所需的内部指令。微码 ROM 完成当前 IA-32 指令排序操作后，机器的前端从跟踪缓存中恢复提取内部指令。

来自追踪缓存和微码 ROM 的内部指令缓存在一个简单的，有序的内部指令队列中，这个指令队列帮助内部指令顺畅流向无序执行引擎。

ITLB 和前端 BTB

IA-32 指令 ITLB 和前端 BTB，如图 4 的上部所示，在机器丢失追踪缓存时引导前端。ITLB 将给定的线性指令指针地址转换为访问 L2 高速缓存所需的物理地址。该所还执行页级保护检查。ITLB 还执行页面级保护检查。

预取与前端 BTB 有关的逻辑的硬件指令从预计将要执行的 L2 缓存中提取 IA-32 指令字节。预取逻辑尝试保持指令解码器馈送程序需要执行的下一个 IA-32 指令。这一指令预取是通过分支预测逻辑引导（分支历史表和分支目标缓存在这里列为前端 BTB）来知道下一步需要获取什么。分支预测允许处理器在以前的分支结果确定之前就开始获取和执行指令。前端分支预测器相当大（有 4K 分支目标项），可以捕获程序的大部分分支历史信息。如果一个分支在 BTB 中找不到，分支预测硬件将基于分支位移的方向静态地预测分支的结果（向前或向后）。向后的分支被假定为被接收，向前的分支被假定为不被接收。

IA-32 指令译码器

指令译码器从 64 位二级缓存一次性接收 IA-32 指令字节，将它们解码成原函数，也成

为内部指令，这些指令是机器能够执行的。这一指令解码器的最大解码速率可以达到在每一个时钟周期解码一个 IA-32 指令。许多 IA-32 指令被转换成一个单一的指令，其他 IA-32 指令则需要好几个指令来完成全部操作。如果完成一个 IA-32 指令需要超过四条内部指令，解码器将机器发送到微码 ROM 来完成指令。大多数指令不需要跳转到微码 ROM 来完成。串动是一个包含多条内部指令的指令的例子，串动可能包含上千条内部指令。

无序执行逻辑

无序执行引擎包括分配、重命名、和调度功能。这个部分的功能是重新安排指令使机器能在输入操作数后能够尽快准备执行。

处理器在每个时钟周期试图寻找并执行尽可能多的指令。无序执行引擎在每个时钟周期将会执行尽可能多的已经准备好的指令，即使他们不是按照原始程序的顺序。通过即时地分析程序中的大量指令，无需执行引擎通常能找到许多已经准备执行且独立的指令来开始执行。NetBurst(微体系结构)比 P6(微体系机构)有更大的缓冲区来使引擎进行这些操作。它能够一次处理 126 条指令并向机器中分配 48 条负载和 24 条存储。

分配器

无序执行引擎有数个缓冲区来执行重新排序，追踪和排序操作。分配器逻辑根据每个微指令的执行需要来分配许多关键的机器缓冲区。如果在这个时钟周期内分配器的三条为指令中的一个无法利用一个资源(比如寄存器文件接口)，分配器会关掉这部分机器。当这些资源能够利用后，分配器会把它们分配到请求微指令并把需要这些资源的微指令放到流水线中执行。分配器分配一个重新排序缓冲区(ROB)接口，功能是追踪这 126 个同时在机器中执行的微指令的其中一条微指令的执行完成状态。分配器也会为微指令的结果数据值分配一个接口(128 个整数或浮点数寄存器之一)，并且可能分配一个加载或存储缓冲区用来追踪流水线中的 48 条负载和 24 条存储中的一个。此外，分配器在指令调度之前会为这两个微指令队列之一分配一个接口。

寄存器重命名

寄存器重命名逻辑重命名 IA-32 寄存器(比如把 EAX 电子自动交换机命名为 the processors 128-entry physical register file)。这使得小的，8 入口的，基于架构的 IA-32 寄存器文件能被动态扩展到使用 128 物理寄存器的奔腾 4 处理器。重命名处理去除了由多重指令产生的同步但唯一的寄存器版本(比如 EAX 电子自动交换机)造成的错误冲突。在机器流水线中会同时有许多独特的 EAX 的实例。重命名逻辑会保存每个寄存器的最近版本(比如 EAX 电子自动交换机)在寄存器别名表(RAT)中，这样每一个从流水线上下来的新指令就能知道每个正确且最近的它的输入操作数寄存器的实例在什么位置。

如图 5 所示的 NetBrust 微体系结构与 P6 微体系结构在分配并重命名寄存器上有些许不同。图 5 的左边展示的是 P6 的模式。它把数据结果寄存器和 ROB 接口作为一个单一，广泛的有数据和状态字段的实体来分配。ROB 数据字段是用来存储微指令的数据结果值，ROB 状态字段是用来追踪微指令在机器中执行时状态。这些 ROB 接口被循序地分配并解除配置，通过一个标志它们与接口相对时间的序列号。在执行之后，结果数据从 ROB 数据结果字段拷贝到各自的撤回寄存器文件(RRF)。寄存器别名表(RAT)指向每个架构寄存器(如 EAX)的最近的版本。这个最近的寄存器可能在 ROB 或 RRF 中。图 5 的右边展示的是 NetBurst 微体系结构的配置方案。它独立地分配 ROB 接口和结果数据寄存器文件(RF)接口。

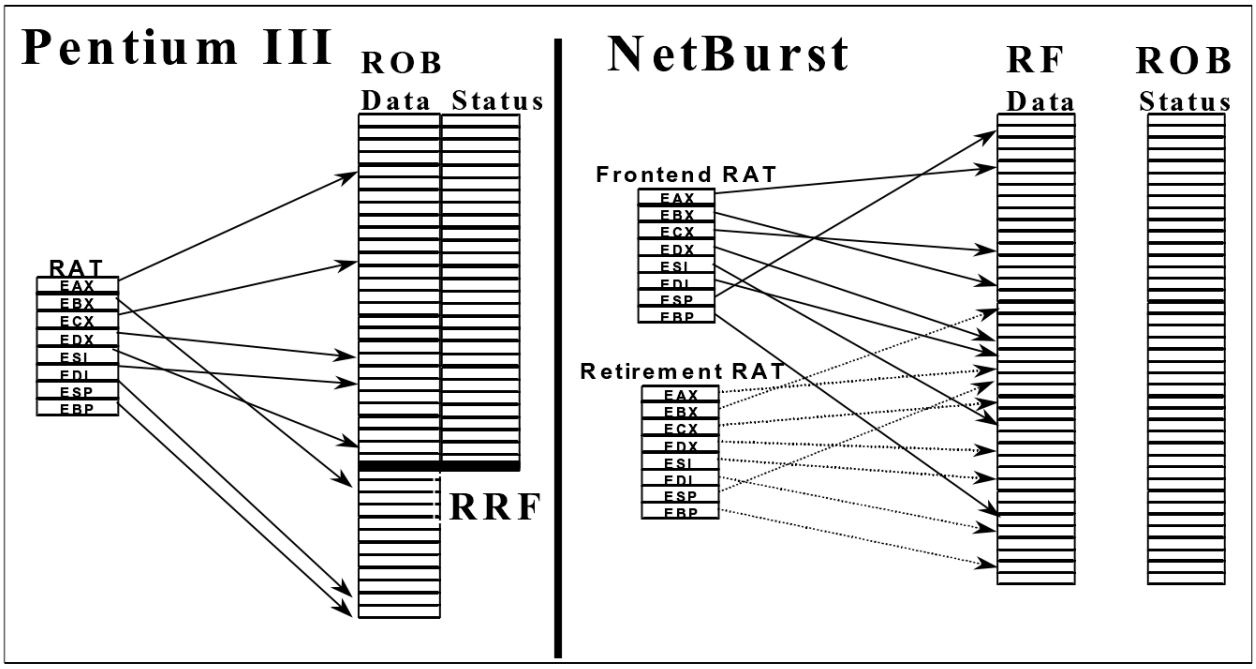

Figure 5: Pentium® III vs. Pentium® 4 processor register allocation

追踪微指令状态的 ROB 条目仅由状态字段组成并被循序地分配和回收。分配给每个微指令的序列号标志了其相对存在时间。与 P6 微体系结构相同，这个序列号指向 ROB 队列中微指令的接口。像不循序地分配 ROB 条目一样，寄存器文件条目被从 128 条目-RF 中的可用的寄存器分配。分配后，实质上没有结果数据值从一个物理结构转移到另一个物理结构中。

微指令调度

微指令调度器通过追踪其输入寄存器操作数来确定一个微指令是否已经准备好执行。这是无序执行引擎的核心。微指令调度器使指令在准备好时就能被重新整理并执行,同时仍然保持正确的原始程序依赖关系.NetBurst 微体系结构有两套结构用来协助微指令调度：微指令队列和实际的微指令调度器。有两个微指令队列-一个用于存储操作(加载和储存)，另一个用于非存储操作。这些队列按照严格的 FIFO(first-in，first-out 先进先出)顺序存储微指令，但其他队列能够无序地访问这些队列。这使得动态无序调度窗口比能够使微指令调度器做所有的重新排序工作所需的窗口要大。

图 6 展示的是个别用来为奔腾 4 处理器上的各种执行模块调度不同类型的微指令调度器。这些调度器基于输入寄存器操作数来源的就绪情况和微指令完成他们的操作所需的执行资源来决定微指令是否就绪。

这些调度器与四个不同的调度端口绑定。图 6 中有两个执行单元调度端口分别标记为端口 0 和端口 1。这些端口处理速度很快 :它们可以在每个主处理器时钟周期内分配多达 2 个操作。复合调度器共享这两个分配端口。快的运算器(ALU)调度器可以在每半个主时钟周期进行调度，而其他调度器只能只能在每一个主处理器时钟周期调度一次。它们在复合调度器有准备就绪的操作时立即竞争分配端口。还有一个加载和存储调度端口可以在每个时钟周期分配一个就绪的加载和存储。总的来说,每个主要的时钟周期这些微指令调度端口可以被分派到 6 个微指令。这个调度带宽超过前端带宽和撤回带宽，3 条微指令每个时钟周期，使得峰值可以大于 3 条微指令每个时钟周期并且能够灵活地分配微指令到不同调度端口。图 6 展示了这种能在每个时钟周期分配到每个端口的操作。

整数和浮点数执行单元

执行单元是指令实际执行的位置。为了全面优化性能，执行单元被设计为能尽可能快地处理大多数常见的事件。在 NetBurst 微架构中有几个不同的执行单元。用来执行整数运算的单元包括低延迟整数 ALU，复杂整数指令单元，加载和存储地址生成单元和 L1 数据高速缓存。

浮点数（x87），MMX，SSE（Streaming SIMD Extension）和 SSE2（Streaming SIMD Extension 2）在两个浮点执行块中被执行。MMX 指令是在 8，16 或 32 位操作数上运算的 64 位压缩整数 SIMD 运算。SSE 指令是 128 位压缩 IEEE 单精度浮点数运算。奔腾 4 处理器加入了被称作 SSE2 的新形式的 128 位 SIMD 指令。SSE2 指令支持 128 位压缩 IEEE 双精度 SIMD 浮点数运算和 128 位压缩整数 SIMD 运算。压缩的整数运算支持 8，16，32 和 64 位操作数。若想了解更多关于这些 SIMD 运算的细节，可以查看 IA-32 Intel Architecture Software Developer's Manual Volume 1: Basic Architecture。

整数和浮点数寄存器堆在调度器和执行单元中间。对于整数和浮点/SSE 运算都分别有一个 128 输入寄存器堆。每个寄存器堆也有一个多时钟分流网络，可以分流或转发还没有被写入寄存器堆中的刚刚运算完成的结果到新的依赖的微指令。由于该设计中非常高的频率，这种多时钟分流网络是必需的。

低延迟整数 ALU

奔腾 4 处理器执行单元的设计是通过尽可能快地处理大多数常见的事件来实现全面优化性能。奔腾 4 处理器能以两倍主频做完全依赖 ALU 的运算。ALU 分流循环是处理器流水线一个关键的闭循环。典型的整数程序中大约有 60%-70%的微指令使用这个关键整数 ALU 循环。在主时钟的 1/2 延迟下执行这些运算有助于提高大多数程序的执行速度。在一个半时钟循环中执行 ALU 运算并不能得到 2 倍的性能增长，但可以改善大多整数应用性能。

这个高速 ALU 内核被做得尽可能小，以便于减小金属覆盖的长度和负载。只有执行频繁的 ALU 运算所需要的必要硬件会被包含于这高速 ALU 当中。对于多数整数程序，那些不频繁使用的功能不会放在这个关键的低延迟 ALU 中，而是放在其他地方。放在其他地方的整数运算硬件，例如，乘法器，移位，标记逻辑和分支进程。

处理器用一个半时钟周期的有效延迟来进行 ALU 运算。如图 7 所示，处理器用一系列的三个快速时钟周期来进行这个运算（快速时钟以 2 倍的主频运行）。在第一个快速时钟周期，低 16 比特被计算并且立刻可以满足下一个依赖的运算时钟周期所需要的低 16 比特。高 16 比特在下一个快速时钟周期被运算，用刚刚由低 16 比特生成的运算结果。这个高 16 比特的运算结果在下一个依赖的运算所需要时会被使用。这种机制叫做交错添加。ALU 标记在第三个快速时钟周期中被执行。交错添加意味着只有一个 16 比特添加并且它的多路输入需要在第一个快速时钟周期中完成。一次需要低 16 位数据，为了在 L1 数据高速缓存用于地址输入时打开入口。

复杂整数运算

简单而又十分频繁的 ALU 运算在上述高速整数 ALU 执行单元中被执行。为了完成更复杂的整数运算，需要用到其他分离的硬件。多数整数移位或循环运算被送到复杂整数分派端口。这些移位运算有四个时钟的延迟。整数乘法和除法运算也有长的延迟。典型的乘法和除

法形式分别有大约 14 和 60 个时钟的延迟。

低延迟 1 级（L1）数据高速缓存

1 级（L1）数据高速缓存是一个 8 千字节的高速缓存，用于整数和浮点/SSE 的加载和存储。它是一个 4 通道的级联调整缓存，每个高速缓存线上有 64 字节。它是一个直接写入式高速缓存，意味着写入它的内容总是被复制到 L2 高速缓存中。每个时钟周期它可以做一次加载和一次存储。

加载操作的延迟是处理器性能的关键方面。 由于指令集中的寄存器数量有限，因此对于具有大量加载和存储的 IA-32 程序尤其如此。 NetBurst 微架构通过由具有中等延迟的大型高带宽二级缓存备份的小型，非常低延迟的 8K 字节缓存来优化最低的总负载访问延迟。对于大多数 IA-32 程序，此配置的一个小型但非常低延迟的 L1 数据高速缓存后跟大型中等延迟 L2 高速缓存，提供较低的负载访问延迟，因此具有比较大，较慢的 L1 缓存更高的性能。 L1 数据缓存以整数负载的 2 时钟负载使用延迟和浮点/ SSE 负载的 6 时钟负载使用延迟进行操作。

奔腾 4 处理器的非常高的时钟速率难以实现 2 个时钟的负载延迟。 该缓存使用新的访问算法来实现非常低的加载访问延迟。 新算法利用了几乎所有的访问命中第一级数据高速缓存和数据 TLB（DTLB）的事实。

在这个高频率下，使用这种深层机器流水线，从负载调度器到执行的时钟距离比负载执行延迟本身要长。 在父负载完成执行之前，微指令调度程序调度相关操作。 在大多数情况下，调度程序假定负载将击中 L1 数据高速缓存。 如果加载丢失了 L1 数据高速缓存，则在运行中将存在依赖操作。 离开调度程序的这些依赖操作将暂时收到不正确的数据。 这是一种数据猜测的形式。 使用称为重播的机制，逻辑跟踪并重新执行使用不正确数据的指令。只有依赖操作才能重放：独立的操作被允许完成。

存储器子系统中任何一个时刻，L1 数据缓存中可能存在至多四个未完成的加载缺失。

存储到负载转发

在无序执行处理器中，不允许存储被提交到永久性机器状态（L1 数据高速缓存等）,直到存储器退休。等待直到退休意味着所有其他前操作已经完成。所有故障,中断,预测分支等必须事先通知,以确保这个存储器安全的执行。随着奔腾 4 处理器管道的深入，需要许多时钟周期才能使一个存储器退休。同时，存储器在退休之前通常需要等待的存储器来完成数据缓存的更新。这台机器的传输途径一次可以拥有高达 24 个存储器。有时他们中的许多已经退休，但还没有将他们的状态提交到 L1 数据缓存中。其他存储器可能已经完成,但是还没有退休,所以他们的结果也没有在 L1 缓存数据中。由于可用的寄存器数量有限，通常负载必须使用其中一个暂存存储的结果，特别是对于 IA-32 程序。为了实现对待处理存储的这种使用，现代无序执行处理器具有一个挂起的存储缓冲区，允许加载在存储已写入 L1 数据高速缓存之前使用挂起的存储结果。这个过程称为存储到负载转发。

为了使这种存储到负载转发过程有效，这个待处理的存储缓冲区被优化，以允许有效和快速地将数据转发到来自待处理存储的依赖负载。奔腾 4 处理器有一个 24 入口存储转发缓冲区与存储器的数量匹配以保证能够立刻被传输。如果负载与仍在存储转发缓冲区中的正在进行的完成的挂起存储相同，则允许转发。负载也必须与待处理存储大小相同，并且具有与存储相同的起始物理地址，以进行转发。这是迄今为止最常见的转发案例。如果一个加载器请求的字节仅部分地与一个挂起的存储器重叠，或者需要一些字节来自多个待处理存储器同

时进行，则不允许进行存储转发。负载必须从缓存中获取其数据，并且在存储已将其状态提交到缓存之前无法完成。

从性能损失方面,这种不允许的存储到负载转发的情况可能是相当昂贵的，如果它经常发生。发生这种情况时，往往会发生在较旧的 P5 内核优化应用程序上，这些应用程序尚未针对现代的无序执行微体系结构进行优化。IA-32 编译器的较新版本删除了大部分或全部这些不良的存储转载案例，但仍然存在许多旧版 P5 优化应用程序和基准测试。这种不利的存储转发案例对于基于 P6 的处理器和其他现代处理器来说是一个很大的性能问题，但是由于奔腾 4 处理器的更深层次的管理，这些情况在性能上更加昂贵。

FP / SSE 执行单元

奔腾 4 处理器的浮点（FP）执行群集是执行浮点，MMX，SSE 和 SSE2 指令的地方。这些指令通常具有宽度为 64 到 128 位的操作数。FP / SSE 寄存器文件有 128 个条目，每个寄存器为 128 位宽。该执行群集具有两个 128 位执行端口，每个时钟周期都可以开始新的操作。一个执行端口用于 128 位通用执行，一个执行端口用于 128 位寄存器到寄存器移动和存储器存储。FP / SSE 引擎也可以在每个时钟周期内完成一个完整的 128 位负载。

在奔腾 4 处理器的开发周期早期，我们有两个完整的 FP / SSE 执行单元，但这花费了大量硬件，并且在大多数 FP / SSE 应用中并没有购买非常多的性能。相反，我们使用 FP / SSE 移动的简单第二端口和 FP / SSE 存储数据原语来优化成本/性能权衡。这种权衡显示出能够以更少的芯片尺寸和功耗成本购买第二个全功能端口的大部分性能。

许多 FP /多媒体应用程序具有相当平衡的乘法和加法。通常，与完全流水线化所有 FP / SSE 执行硬件相比，该机器通常可以保持每两个时钟周期一次乘法和一次加法繁忙的交织。在奔腾 4 处理器中，FP 加法器可以每个时钟周期执行一个扩展精度（EP）加法，一个双精度（DP）加法或两个单精度（SP）加法。这允许它每两个时钟周期完成 128 位 SSE / SSE2 封装的 SP 或 DP 加法阵列。FP 乘法器可以每两个时钟执行一个 EP 倍增，也可以在每个时钟执行一个 DP 乘法或两个 SP 乘法运算。这允许它每两个时钟周期完成一个 128 位 IEEE SSE / SSE2 封装的 SP 或 DP 乘法器，给出单精度峰值 6 GFLOPS 或 1.5GHz 时双精度浮点的 3 GFLOPS。

许多多媒体应用程序交织加法，乘法和打包/解包/随机播放操作。对于 64 位宽的 MMX 或 128 位宽 SSE2 指令的整数 SIMD 操作，可以并行运行三个执行单元。SIMD 整数 ALU 执行硬件可以在每个时钟周期处理 64 个 SIMD 整数位。这允许该单元每两个时钟周期执行新的 128 位 SSE2 压缩整数。单独的随机播放/解包执行单元还可以在每个时钟周期处理 64 个 SIMD 整数位，从而每两个时钟周期执行一次完整的 128 位洗牌/解包操作。MMX / SSE2 SIMD 整数乘法指令使用上述 FP 乘法硬件，也可以每两个时钟周期执行 128 位压缩整数乘法。

FP 分频器执行所有除法，平方根和余数。它基于双泵 SRT 基数-2 算法，每个时钟周期产生两位商（或平方根）。

实现显着更高的浮点和多媒体性能需要的不仅仅是快速执行单元。它需要一个平衡的功能集合。这些程序在其内部循环中经常具有许多长延迟操作。奔腾 4 处理器的非常深的缓冲区（飞行中有 126 个 uop 指令和 48 个 load 指令）允许机器一次检查大部分的程序。无序执行硬件通常在其执行窗口中多次展开这些程序的内部执行循环。这种动态展开允许奔腾 4 处理器通过找到许多独立的指令同时工作来重叠长延时 FP / SSE 和存储器指令。大多数 FP /多媒体应用程序的这个深层窗口比更多的执行单元可以获得更多的性能。

FP /多媒体应用通常需要非常高带宽的存储器子系统。有时，FP 和多媒体应用程序在 L1 数据高速缓存中不能很好地适配，但是适合 L2 高速缓存。为了优化这些应用，奔腾 4 处理

器具有从 L2 数据高速缓存到 L1 数据的高带宽路径。一些 FP /多媒体应用程序从内存流数据 - 没有实际的高速缓存大小将保存数据。他们需要主存的高带宽路径才能表现良好。长 128 字节的 L2 高速缓存行与下面描述的硬件预取器有助于预取应用程序将要需要的数据，从而有效地隐藏长存储器延迟。奔腾 4 处理器的高带宽系统总线允许这种预取来帮助执行引擎充分利用流数据。

存储子系统

奔腾 4 处理器拥有高新能存储子系统，可以处理新型的、高带宽、面向流数据的 3D、视频和内容创建应用。存储子系统包括 L2 缓存和系统总线。L2 缓存存储没有被 L1 命中的数据。系统外部总线被用来在 L2 未被命中时访问内存，也用来做系统级的设备 IO。

L2 指令和数据缓存

L2 缓存是一个 256KB 的缓存，它存储未被踪迹缓存命中的指令和未被 L1 缓存命中的数据。L2 缓存架构为 8 路集合，每一个缓存线路有 128B。这些 128B 大小的线路包括两个 64B 的扇区。如果未命中 L2 缓存，这个未命中信息典型地初始化两个 64B 块访问系统总线，来填充缓存线路的两个部分。L2 缓存是回写缓存，当存储或读取命令未命中时，它分配新的缓存区域。它拥有一个 7 个时钟周期的访问网络。一个新的缓存操作能在每两个处理器时钟周期开始，当工作在 1.5GHz 时，最高可以达到 48GB/s 的带宽。

与 L2 缓存相关联的是一个硬件预取器，它监视数据访问模式并将数据自动地预取到 L2 缓存中。它在实时的数据区域中预留出 256B，并记录缓存未命中的信息来检测并发、独立的数据流，来达到预取数据的目的。预取器也尝试尽可能地减少预取无用的数据，这些无用的数据会占用存储空间，延迟真实需要访问的数据的时间。

400MHz 系统总线

奔腾 4 处理器有一个 3.2GB/s 的系统总线，这高速系统总线是实现流数据存储的关键。它宽 64bit，能以 400MHz 的速度传输数据。它使用一个资源同步协议来保证数据在 100MHz 的总线上传输 400 百万的数据。它有一个分离交互，高度流水化的协议来保证存储子系统重叠处理许多并发的请求，以至真正地在实时系统中实现高速的存储带宽。总线协议的访问宽度也是 64bit。

性能表现

奔腾 4 处理器拥有 SPEC 整数标准的最高性能。SPEC 是衡量整数和浮点数处理能力的工业标准。图 8 显示了奔腾 4 处理器工作在 1.5GHz 和奔腾 3 工作在 1GHz 的不同应用的性能。整型应用的性能提高了 15-20%，FP 和多媒体应用性能提高了 30-70%。对于 P2000 来说，新的 SSE/SSE2 指令集对比与 x87 有了 5%的性能提升。随着编译器能力的不断提升，这些从新指令集中得到的性能提升会不断增加。同时，随着奔腾 4 处理器的主频不断提升（设计不断成熟），所有这些的性能差异会不断加大。

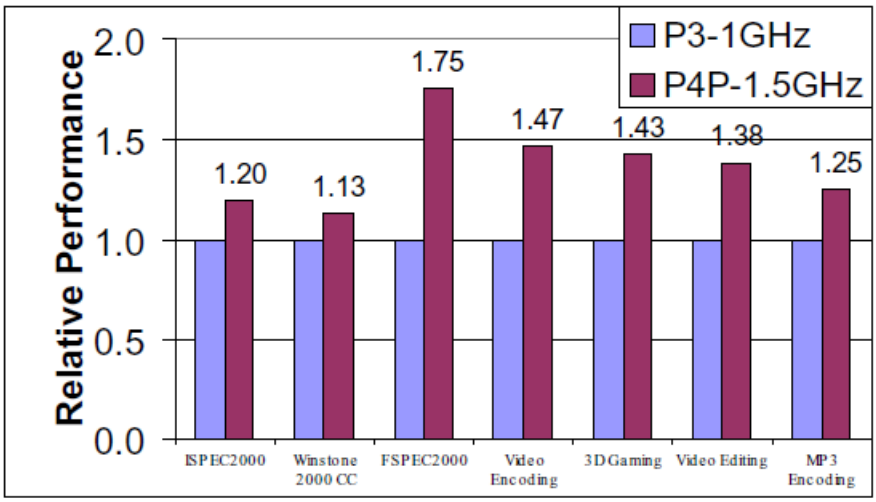

Figure 8: Performance comparison

更多的奔腾 4 处理器的应用性能对比，请访问 http://www.intel.com/procs/perf/pentium4/

结论

奔腾 4 处理器是一个全新的处理器和微处理架构。它是使用新的 Intel NetBurst 微架构的新处理器家族的开始。它被设计为高度流水化的，拥有世界领先的主频。它使用了许多创新的微架构思想，包括踪迹缓存，双时钟周期 ALU，新的低延迟 L1 缓存算法，和一个高带宽的系统总线。它在媒体丰富的环境、3D 应用、工作站应用和内容创建中，表现出了世界一流的性能。