同时多线程技术

刘权胜， 杨洪斌， 吴 悦
(上海大学 计算机工程与科学学院，上海 200072)

摘 要：同时多线程技术结合了超标量处理器与多线程处理器两者的优点，通过增加很少的硬件资源，把一个物理核映射为多个逻辑核，成为一种研制高性能处理器的重要途径。重点介绍了同时多线程处理器出现的原因、优点、基本组成结构、当前的研究成果及影响，并探讨了当前同时多线程技术的发展趋势。

关键词：模型；处理器；多线程；同时多线程；同时多线程处理器
中图法分类号：TP303 **文献标识码**：A **文章编号**：1000-7024 (2008) 04-0963-05

Simultaneous multithreading technology

LIU Quan-sheng, YANG Hong-bin, WU Yue
(School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China)

Abstract: Simultaneous multithreading technology is an important method manufacturing high performance processor which makes use of the advantage of superscalar and multithreading processor, and maps one physical core into multiple logic cores by adding a little hardware resources. The reason, advantage, architecture, status, and trend of the recent simultaneous multithreading technology are presented.

Key words: model; processor; multithreading; simultaneous multithreading; simultaneous multithreading processor

0 引 言

从1971年的Intel 4004第1块微处理器芯片诞生以来，微处理器芯片按摩尔定律的速度不断的更新换代，主频从1 MHZ到1 GHZ，晶体管从几千到上亿。为了提高微处理器的性能，流水线的深度不断增加，但是研究表明处理器的指令级并行的实际极限超出了现代处理器可以获得的能力[1]。随着出现了超标量流水线、超流水线、超长指令处理器、细粒度多线程、粗粒度多线程的多线程处理器、单芯片多处理器[1-5]。根据当前的发展趋势与国外微处理器的发展战略，在不久的将来将会出现8核、16核、32核的微处理器。

超标量流水主要是通过硬件来实现发射多条指令，提高指令的并行度，如图1(a)所示。图1水平方向代表4个功能部件或者发射槽，垂直方向代表不同的时钟周期，在某个时钟周期4个功能部件或者发射槽完全空闲时称为垂直浪费，有部分空闲时称为水平浪费，A、B、C、D代表不同的线程，空白代表在当前时钟周期空闲。超长指令主要是通过编译来提高并行度，每次发射一条长指令，但是有时由于指令的相关性会造成很多的空操作，造成大量的水平浪费。细粒度多线程主要是通过在每个时钟周期多个线程进行上下文切换，这样可以减少垂直浪费，但是在提高总的吞吐率而牺牲了单个线程的性能，如图1(b)所示。粗粒度多线程主要是通过在线程遇到长延迟操作时线程之间进行上下文切换，如在执行过程中出现Cache失效，多个线程在处理器中交叉执行，从而提高了处理器执行资源的利用率，有效的隐藏了很大一部分Cache失效延迟，否则不发生上下文切换，继续执行当前的线程，与细粒度多线程类似对减少垂直浪费取到了良好的效果，如图1(c)所示。单芯片多核处理器是在一个芯片上集成多个物理核，从而采用更复杂与更占面积的体系结构，例如采用乱序执行机制，高准确的分支预测部件，甚至采用大小可变的动态二级Cache，如图1(d)所示。

在1995年由美国Washington大学的Tullsen, Eggers和Levy等人提出了同时多线程处理器的概念，如图1(e)所示。允许多个线程在共享的执行资源中以细粒度的方式任意的动态交叉执行指令，以尽量提高处理器执行资源的利用效率。研究人员提出的同时多线程技术(simultaneous multithreading, SMT)能够同时减少水平浪费与垂直浪费。这样能够更好的通过指令级并行 (instruction-level parallelism, ILP) 与线程级并行性(thread-level parallelism, TLP)提高处理器的效率。Intel Hyper-Threading技术是将SMT技术应用于基于Pentium 4的Xeon处理器[6-7]。最近也有研究人员提出了隐式多线程 (implicit multithreading, IMT)[8]的处理器的方向，我们目前的多线程处理器和多处理机系统都是显式，需要人来开发多线程，或者需要专门的编译器，更差的情况时会牺牲某些重要线程的性能，所以

收稿日期：2007-03-05 **E-mail**：post1117@163.com
基金项目：上海—应用材料研究与发展基金项目 (06SA18)。
作者简介：刘权胜 (1981—)，男，江西宜春人，硕士研究生，研究方向为计算机系统结构、ASIC芯片设计；杨洪斌 (1962—)，男，博士研究生，副教授，研究方向为计算机系统结构、ASIC芯片设计、SOC；吴悦 (1960—)，女，博士，教授，研究方向为计算机辅助设计和EDA技术。

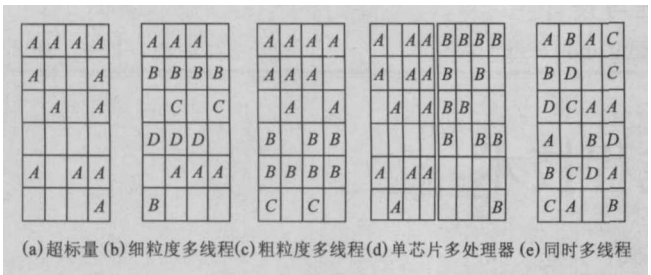

图 1　功能部件或者发射槽的利用效率情况

也就出现了隐式多线程的概念。但是要实现隐式多线程技术需要解决几个难题，即化解控制相关，化解寄存器数据相关，化解存储器数据相关。因此，隐式多线程技术还处在研究阶段。

1　同时多线程处理器

1.1　同时多线程处理器的结构模型提出

同时多线程是一种新的处理器体系结构技术，它允许在一个时钟周期内发射多个线程的多条指令执行，同时利用程序的 TLP 和 ILP 来消除水平浪费和垂直浪费，提高处理器发射槽及功能部件的利用率。SMT 结合了超标量和多线程处理器的特点，可以同时减少水平和垂直浪费，SMT在这两个方面提高了处理器的总体性能。Tullsen、Eggers 和 Levy 等人在超标量的基础上做了少量的改进，同时结合多线程处理器的特点，提出了 SMT 结构，如图 2 所示。

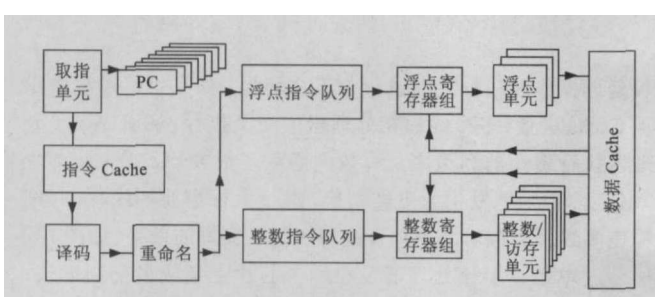

图 2　同时多线程处理器硬件结构

在提出的SMT处理器结构中，有大量的寄存器文件读写，对流水线增加了一个读周期和一个写周期，如图 3 所示。

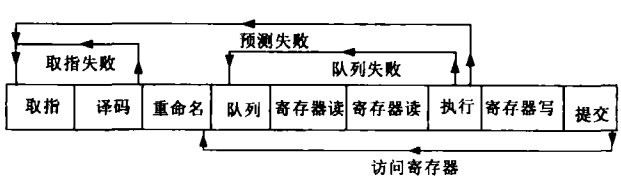

图 3　SMT 流水

1.2　Intel Xeon SMT Processor

Intel Xeon SMT Processor 是现在已经成功的一款处理器，它的成功证明了 SMT 的优越性，Intel Xeon SMT Processor 的性能相对于 Pentium III 要提高 65%，面积仅增加 5%，它的流水线结构，如图 4 所示。Intel Xeon SMT Processor 有 3 种工作模式：ST0-Mode 或者 ST1-Mode 的单处理器，两个处理器工作模式及 Lower-Power-Mode。当其中的一个逻辑物理核执行了 HALT 指令，停止本身逻辑物理核的执行，另一个逻辑物理核处于正常工作状态。如果两个逻辑物理核都执行了 HALT 指令，就

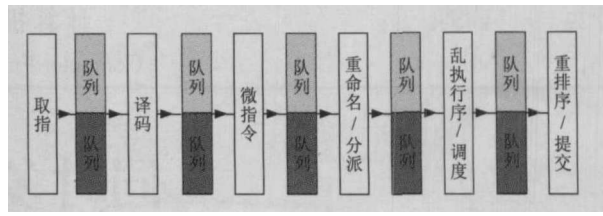

图 4　Intel Xeon SMT Processor 流水线

会转到 Lower-Power-Mode。第 3 种就是通常的超线程模式，即两个逻辑物理核的均在工作的状态[7]。

实际上 Intel Pentium 4 采用的是混合形式的多线程，与 Washington 大学的 Tullsen、Eggers 和 Levy 等人提出的实现方案有本质的不同，因为 Intel Pentium 4 只有在发射，执行，访存段真正的实现多个线程以任意的方式交叉执行。因此没有实现真正意义上的同时多线程技术。而是采用了细粒度与粗粒度相结合的技术。因此，Intel Xeon SMT Processor 的性能还没有达到最大程度的开发。并且在线程切换的情况下，会损失某些重要性能的线程，最终可能影响整个处理器的性能。

1.3　同时多线程处理器的关键组成部分

SMT 处理器粗略的由前端和后端两部分组成。前端部分主要由取指部件，存储层次和分支预测器组成。后端部分由余下部分组成，如图 2 所示。

1.3.1　取指部件

取指部件在设计的过程中所采取的策略对处理器的性能有至关重要的影响。Tullsen、Eggers 和 Emer 等人对下面 5 种取指策略进行了评估：ICOUNT 指优先从在译码，重命名和指令队列中具有最少指令数目的某个线程中取指；BRCOUNT 指优先从在译码，重命名和指令队列中具有最少分支指令数目的线程中取指；MISSCOUNT 指优先从具有最少正在进行的，未完成数据 Cache 不命中操作的线程中取指；IQPOSN 指位置最靠近定点或浮点队列头部的指令所属的线程优先级最低取指；RR 指用 Round-Robin 的方式从不同的线程中周期性取指，以上 5 种取指策略的结果效果表明 ICOUNT 是最好的一种策略。随着研究工作的方法不断改进，不断出现新的方法：①何立强等提出了一种具有 QOS 特性的 SMT 取指策略，该策略的核心思想是利用线程的优先级和流速来同时控制线程的取指过程，从而满足线程在执行速度上的 QoS 需求，与传统的基于纯优先级的取指策略相比，该策略不但具有 QoS 特性，同时还可以更加有效地分配取指带宽，从而能获得更高的处理器性能。模拟实验的结果表明，该策略在提供 QoS 支持的基础上，可以在传统的基于优先级的取指策略 ICOUNT 的基础上提高 15% 的系统性能[9]；②也有研究者从复杂性的角度提出了 Gskew+FTB、Stream Predictor[10]技术，在复杂方面做出了改进，允许用 8 路在一个线程中取指，降低取指单元部件的复杂性，提高取指策略的使用性，结果表明取得的性能优于 ICOUNT 策略；③Fetch Prioritizing 和 Fetch Gating 策略，Fetch Prioritizing 主要是根据每个线程有还没确定的分支数目来给线程定义优先级，Fetch Gating 是阻止那些没确定的分支数目超过一个限制的线程，从而使流水线中尽可能少错误路径方向的指令，也取的了良好的性能，比 ICOUNT 提高了 14.9%[11]。发射策略在 SMT 中也是一个需要考虑的方面。但是，根据 Tullsen 等人的研究结果表明发射策略在 SMT 中的中的影响不明显。

1.3.2 分支部分

流水线机器只有在流水模式下才能达到最大的吞吐率。因此，分支部分在以往的处理器设计中起到了关键的作用。目前，常用的分支技术有Gshare、Gselect、2-bit分支预测、Trace Cache技术。根据Hily与Seznec对前面3种分支策略的研究结果表明Gshare的精度最高，相反2-bit分支表现最差。但是，Matt Ramsay、Chris Feucht等人针对这几种策略在同时多线程处理器的条件下进行了性能分析，结果表明分支采用的策略准确性对同时多线程处理器的影响不大[12]。目前，Trace Cache是比较先进的技术，在每个周期能够准确的地处理多条跳转分支指令，包括预测分支方向和目标地址，取指，对齐，合并多条分支的目标地址等一系列工作。Trace Cache是一种基于历史信息的取指机制，它将指令的动态执行轨迹存储在一个Cache中，通过指令地址和分支结果进行访问。因此，Trace Cache中重排序的基本块接近于动态执行的顺序，并且取指部件很少在Trace Cache不连续的位置取指，这样解决了在Cache中的长的指令序列不相邻的问题。传统的技术在这方面限制了处理器的性能，用这种技术跟踪动态的指令流，使得不相邻呈现了相邻的效果，通过指令控制流和指令的提供，取得了低延迟高效的取指带宽[1]。Intel Xeon SMT Processor中就是采用了这种先进的技术。当前，也有研究人员提出了神经网络及与其它算法相结合的预测器，主要是用来解决当前预测器的精确性的极限问题。

1.3.3 存储层次

存储层次的设计是直接关系到访存指令的效率，以致决定到整个处理器的性能。在简单的情况下，存储层次中Cache的大小会对处理器的性能产生一定的影响，片上的两级Cache在大多数的工作量时性能略微高于单级Cache的情况。Tullsen，Eggers和Levy在以四发射八线程及L2Cache与L3Cache共享为前提条件下，研究了Cache的性能。线程数越少，指令与数据分别采用多线程共享Cache方式性能优于线程采用单独的方式；相反线程数越多，指令与数据Cache用每个线程单独分开的方式的性能更佳。而平均的情况下，指令Cache采用单独方式与数据Cache采用共享方式微处理器能够获得最优的性能。Hily与Seznec研究表明L2Cache是SMT中的一个关键部分，如果忽视L2Cache将会导致我们对机器性能的过高估计，甚至会得出错误的结论。L1Cache的相连度越高，分块越小，处理器的性能发挥的越好。同时，处理器支持的线程数取决于Cache的大小否则处理器的性能会下降。Baboescu与Tullsen还提出了甚至以牺牲访问第一级Cache的时间为代价，提高Cache的相联度，SMT的性能也能够到达提高的效果，并且用一种新的替换策略Promote-K，能够有效的减小L1与L2Cache之间的总线使用率，从而有效的提高整体性能。Cache越小，离处理器越近，速度就越快[6]。在设计Cache的级大小等方面的因素很重要，否则会形成影响处理器性能的瓶颈。

1.3.4 寄存器堆.寄存器重用

寄存器重用也称为寄存器回收，包含两种不同的形式：静态和动态[13]。静态编译器重用是一种编译器的优化，主要由两个部分组成：代码生成和寄成器分配。代码生成生成面向机器的指令，寄成器分配是将无限的寄存器符号资源映射到有限的，固定的体系结构寄成器上。动态是指循环中的指令重复执行导致的重用。重命名寄存器是现代处理器的关键部分，通过寄存器消除假相关。解决有限资源的方面：①针对提高SMT的寄存器文件的使用效率，为了体系结构和重命名器的需要，使得不同线程之间的寄存器全局共享，主要表现在两种情况：操作系统能够把处于空闲线程的上下文分配给其它的线程使用；处理器通过编译器而不需等到寄存器的重新定义就进行重新使用[14]；②提出了一种降低访问寄存器时间的方法，当寄存器里的数据的有效位数较少时，把它存储在映射表中，同时释放所占的寄存器资源。在4-wide模型中性能平均提高7.8%，最高达到12.8%；8-wide模型中性能平均提高14.8%，最高达到19.2%[15]；③提出了一种消除线程之间相互影响寄存器重命名性能的方法，这种方法通过跟踪性能的变化与每个线程动态需要的重命名寄存器的数目。TSRR (thread-sensitive register renaming)具有几个方面的优点：每个线程都有发挥自己潜能的机会；每个线程都能够分配到合适的重命名寄存器数目，消除线程之间不利的相互冲突；性能最差的线程不会影响其它的线程，也不会导致其处于“饥饿”状态[16]。

寄存器文件在处理器中是个关键部分，一般情况下有两种实现方式：①在多个线程时，每个线程有自己的寄存器文件(整数和浮点数)，但是还需要重命名逻辑是映射实现。例如，Tullsen、Eggers和Levy等人提出的同时多线程结构。在8个线程同时执行时需要8*32个寄存器还需要100个重命名寄存器；②体系结构寄存器和重命名寄存器集成一个寄存器文件。比如，IBM RS/6000中采用了这种集中寄存器文件方法。重命名逻辑是通过Register Alias Table(RAT)来确定体系结构寄存器与重命名寄存器的关系，也就是实现寄存器与重命名寄存器的映射的桥梁[7]。

1.3.5 队列资源使用策略

队列资源在处理器中采用的策略有多种。在Xeon SMT Processor中它的资源分为完全共享，分割，重复，入口标记4种情况。完全共享部分就是多个逻辑核完全共享的资源；分割部分就是在多个线程工作模式时逻辑核平均分配某个总资源，防止“活锁现象“，否则工作在单线程工作模式时，这部分资源重新组合成一个资源；重复部分每个逻辑核都有自己的单独资源部分；入口标记是部分竞争共享资源，每个逻辑处理器有自己的资源，并用各自的ID确定[17]。不论处理器处于哪种工作模式，通用寄存器、状态寄存器、中断管理器等资源，每个逻辑处理器都进行了重复设置。在设计处理器时达到3个目标[6]：①采用Hyper-Threading技术时增加的面积尽可能小；②保证当其中一个逻辑处理器停顿时，另外的一个逻辑处理器仍然会正常执行；③保证只有单个线程工作时的性能达到非采用Hyper-Threading技术获得的效果。

Uop Queue[6-7,18]中的微指令是来自Trace Cache或者Microcode ROM。Microcode ROM主要是用来存放部分比较复杂指令预先已经译为的简单RISC指令。同时队列的长度也不是越长就越好，而是一定的长度就能满足处理器的性能，否则会造成资源浪费。在实际的队列的资源工作方式主要取决于处理器的工作模式、资源的复杂性和大小、潜在的“死锁”和“活锁”因素及其它的可能性[6]。

1.3.6 指令译码、调度、乱序执行、重排序、功能单元

随着处理器的发展，虽然大多数的处理器在译码段的译码方法比较相似，但是也有相应的变化，最主要的是因为指令系统不同，实现的复杂程度不尽相同。也可能是采用了部分高效的辅助设计部件提高译码效率。在译码阶段还用了Micro-code ROM的辅助进行复杂指令的译码，并且两个线程轮流译码[7]。指令在后端乱序执行时是一个动态的过程，即指令在调度，分派到不同的功能单元。为了进行指令的硬件动态调度，先后提出了两种著名的算法：SCOREBOARD与TOMASUL。SCOREBOARD算法有两个缺点：①在处理RAW相关时没有采用内部前推技术；②在处理WAR相关时没有使用寄存器重命名技术。TOMASULO算法实现了内部数据前推技术和使用预约站实现了寄存器动态重命名技术，这种算法应用在IBM的大型机360/91。第2种方法解决了假相关，但是真相关还仍然存在。近来有研究者研究了载入数据的预测，他们发现同一条静态Load指令载入的是可以预测的，其它的许多指令也是可以预测的，开始了解决真相关问题。最近有提出了动态指令重用的技术思想，动态指令重用技术试图追踪冗余计算，当探测到冗余计算时，先前的计算结果直接被使用而不进行计算操作。这种技术不是推测性的，因此无需进行验证工作。研究表明，当冗余计算技术用于功能性语言书写的程序中时，可以获得很高的性能，最新的研究也证明了这一点，这一领域目前是比较热门。

同时多个线程在功能单元中乱序执行，后来的指令可能会比先发射的指令先执行完，但是在提交的时候还是按有序提交的，即根据每条微指令的重排序缓冲物理序号进行提交，改变机器的状态。功能单元的设计也是微处理器设计的一个关键部分，一般情况下功能单元主要是根据各种指令流出现的百分比、实现的难易程度及处理器的特点来确定功能单元的具体情况，如功能单元由AGU(LoadAddress)、AGU(StoreAddress)、4x ALU (SimpleInstr.) Slow ALU (ComplexIInsttrr.)、FPMMX-SSESSE2、FPMove组成[18]。但是在特殊的情况下，为了进一步提高微处理器的性能，甚至某些功能部件需要重复的设置。

2 基于同时多线程处理器的系统

同时多线程处理器是现代处理器发展的一个重要分支，Xeon SMT Processor在以小的面积和低功耗为代价而获得了高性能，但是也带来些影响，比如引起了饥饿现象、公平性，增加了逻辑复杂性、增加了设立精确条件的困难[17]。同时多线程处理器是以多个线程同时执行来提高性能的，即需要并行的原代码才能获得高效的性能，影响主要表现在以下几方面：①编译器。研究人员用Intel®OpenMP C++/Fortran编译器进行了测试，选择了SPEC OMPM2001的一个子集，用Intel® OpenMP C++/Fortran编译器得到测试代码，测试结果表明使用多线程是处理器的性能得到了提高，但是某些情况下由于线程分派、同步、访存等因素也可能会造成性能下降[19]；②操作系统。研究者研究了操作系统对处理器的影响，结果表明同时多线程处理器在用SPECInt95和Apache Web测试时取的了良好的性能，在八发射的情况下，性能要比传统的超标量提高约4.2倍[20]；③多媒体。Inter Xeon SMT Processor对多媒体的性能也产生了影响，主要在视频编码、视频解码、水印检测方面进行了研究，结果显示了多媒体应用程序在多线程处理器上的性能的到了7%~18%的提高[21]；④数据库。研究人员对同时多线程处理器与数据库的关系进行了研究，结果表明在长延迟的情况下比单个线程的要提高到30%~70%的性能，同时多线程处理器在数据库中有很大的潜力[22]；⑤功耗。集成电路的功耗是一个十分重要的课题，结果表明通过提高资源的利用效率和减少无用指令，SMT的性能的到了大幅度的提高，功耗降低，在4个线程的情况下，执行有用指令的所耗的能量比单个线程的情况下获得22%的改进[23]。文献[24]也提出了高性能、低功耗这两个问题是下一代微处理器设计的目标。章隆兵与何立强的综述文献还对预测、作业调度、同步等方面进行了总结。由此，实践表明新一代的处理器的出现，对当前周围的环境提出了新的挑战。

3 结束语

同时多线程技术是一种以超标量处理器与多线程处理器为基础，通过增加很少的部分硬件资源，把一个物理核映射为多个逻辑核，开发线程级并行与指令级并行，实现多个线程的指令资源同时共享，构成高性性能的体系结构的途径。但是到目前为止，关于SMT的性能潜力，惟一确定的证明来自Intel。由于Pentium4的并行有限，它只支持两个线程，只有在发射、执行、访存段真正的实现了SMT，所以性能提高的幅度要比一些文献中描述的要低很多。然而，这些文献中所描述的方法是否可行，目前还不是很清楚。尽管目前研究人员已经充分了解了实现SMT的耗费、复杂性等理论，但是仍然不清楚即使解决了当前遇到影响SMT性能的问题后，SMT的设计能否扩展到超过两个线程的情况，比如，3个、4个线程的情况。另外，在线程切换的时候可能会影响部分重要线程的性能。因此为了解决当前同时多线程存在的问题，可能从以下几个方向来解决：①在原有的基础上进一步开发多个线程的同时执行，使处理器中个功能部件全部实现SMT技术，实现真正的多个线程任意执行；②利用多线程技术与单芯片多核技术结合来弥补当前的不足。因为单芯片多核处理器技术集成多个核很容易导致圆片面积过大，从而带来制造上的一些问题，尤其是在频率很高的情况下更是如此。经验表明，即使单芯片多核处理器的设计频率达到了单处理器的水平，实际也会因为生产的原因带来一些频率的缺陷。恰好相反，同时多线程能用很小的面积就可以换的很高的性能；③研究一种新的同时多线程技术。即提出一种当前的被称为隐式同时多线程的技术。因为这种技术利用程序控制流的特点，将单个线程自动生成多个并行执行的线程。有资料显示，隐式同时多线程技术对传统的线程级并行效果不太理想的应用却取得了良好的加速性能。这种技术不但可以解决当前的问题，并且还可以解决其它的问题。利用这种技术的同时多线程处理器将不在需要执行人为开发或者利用专门的编译器产生的并行多线程。从而可以在很大的程度上提高同时多线程处理器的性能。

参考文献：

[1] 张承义,邓宇,王雷,等.现代处理器设计——超标量处理器基础[M].北京:电子工业出版社,2004.

[2] 李学干.计算机系统结构[M].西安:西安电子科技大学出版社,2000.

[3] Ron Kalla,Balaram Sinharoy,Joel M Tendler.Power5 chip: A dual-core multithreaded processor[J].IEEE Micro,2004,24(2):40-47.

[4] Hammond L,Hubbert B,Siu M,et al.The stanford hydra CMP[J]. IEEE Micro,2000,20(2):71-84.

[5] Tullsen D M,Lo J L,Eggers S J,et al.Supporting fine-grained synchronization on a simultaneous multithreading processor[C]. Orlando,FL,USA:Proc of the 5th International Symposium on High Performance Computer Architecture,1999:54-58.

[6] David Koufaty,Deborah T Marr.Hyperthreading technology in the netburst microarchitecture [J]. IEEE Computer Society, 2003,23(2):56-65.

[7] Deborah T Marr,Frank Binns,David L Hill,et al.Hyper-threading technology architecture and microarchitecture[J].Intel Technology Journal,2002,6(1):4-15.

[8] Il Park,Babak Falsafi,Vijaykumar T N.Implicitly-multithreaded processors[C]. New York,NY,USA: Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA'03),2003:39-51.

[9] 何立强,刘志勇.一种具有QoS特性的同时多线程处理器取指策略[J].计算机研究与发展,2006,43(11):1980-1984.

[10] Ayose Falcon, Alex Ramirez, Mateo Valero. A low-complexity, high-performance fetch unit for simultaneous multithreading processors [C]. Spain: Proceedings 10th International Symposium on HPCA-10,2004:244-253.

[11] Kun Luo,Manoj Franklin,Shubhendu S Mukherjee,et al. Boosting SMT performance by speculation control[C]. San Francisco, CA,USA: Proceedings of the 15th International Parallel and Distributed Processing Symposium,2001.

[12] Matt Ramsay,Chris Feucht,MikkoH. Lipasti.Exploring efficient SMT branch predictor design[C]. New York,NY,USA: Workshop on Complexity-Effective Design,Conjunction with ISCA,2003.

[13] 李亚民.计算机组成与系统结构[M].北京:清华大学出版社,2000.

[14] Jack L Lo,Sujay S Parekh,Susan J Eggers,et al.Software-directed register deallocation for simultaneous multithreading processors [J]. IEEE Transactions on Parallel and Distributed System, 1999,10(9):922-933.

[15] Mikko H Lipasti,Brian R Mestan,Erika Gunadi.Physical register inlining[C]. Munich,Germany: Proceedings of the 31st International Symposium on Computer Architecture,2004:325-335.

[16] Yang Hua,Cui Gang,Yang Xiaozong.Eliminating inter- thread in terference in register file for SMT processors [C]. Dalian, China: Proceedings of the 6th International Conference on Parallel and Distributed Computing,Applications and Technologies,2005:40-45.

[17] David Burns.Desktop platforms group, Intel Corp pre-silicon validation of hyper-threading technology[J].Intel Technology Journal,2002,6(1):16-21.

[18] Glenn Hinton,Dave Sager,Mike Upton,et al.The microarchitecture of the Pentium 4 processor[J].Intel Technical Journal,2001,5 (1):1-12.

[19] Xinmin Tian,Aart Bik,Milind Girkar,et al.Intel openMP C++/ Fortran compiler for hyper-threading technology: Implementation and performance [J]. Intel Technology Journal, 2002,6 (1): 36-46.

[20] Joshua A Redstone,Susan J Eggers,Henry M Levy.An analysis of operating system behavior on a simultaneous multithreaded architecture [C]. Boston, MA, USA: Proc of 9th International Conference on Architectural Support for Programming Languages and Operating Systems,2000:245-256.

[21] Yen-Kuang Chen,Matthew Holliman,Eric Debes,et al.Media applications on hyper-threading technology[J].Intel Technology Journal,2002,6(1):47-57.

[22] Zhou Jingren,John Cieslewicz,Kenneth A Ross,et al.Improving database performance on simultaneous multithreading processors [C]. Trondheim, Norway: Proceedings of the 31st VLDB Conference,2005:49-60.

[23] John S Seng,Dean M Tullsen,George Z N Cai.Power-sensitive multithreaded architecture[C]. Austin,TX,USA: Proc of the International Conference on Computer Design,2000:199-206.

[24] 赵荣彩,唐志敏.低功耗SMT体系结构研究[J].计算机工程与设计,2002,23(8):7-17.

(上接第962页)

3 结束语

根据引导方式的不同,AGV机器人的灵活性、精确性、自主性和成本也会有很大不同。卷烟企业生产现场要求AGV机器人系统能够高效、准确地采集卷接包机组生产线上各工位补料信息,并高效准确地将辅料、废料等物料送达目标工位。

参考文献:

[1] 周徐昌, 沈建森. 惯性导航技术的原理和应用 [J]. 兵工学报, 2006,25(9):55-58.

[2] 尤辉,樊跃进,杜晶,等.卷烟辅料自动运送系统研究[J].物流科技,2002,12:53-58.

[3] 林雄,郑千里,黄槐仁,等.基于FPGAs的智能机器人导航系统[J].计算机工程与设计,2005,26(3):586-587.

[4] 蔡自兴,邹小兵.移动机器人环境认知理论与技术的研究[J].机器人,2004,26(1):87-91.

[5] 姜涌,曹杰,杜亚玲.基于视觉的码头集装箱AGV导引系统[J].南京航空航天大学学报,2006,38(5):628-633.

[6] 刘俊承,王淼鑫,彭一准.一种基于视觉信息的自主搬运机器人[J].科学技术与工程,2007,7(3):314-319.

[7] 胡明昊,杨文杰,任明武,等.一种基于视觉的道路检测算法[J].计算机工程与设计,2005,26(7):1704-1706.

[8] 唐鸿儒,宋爱国,章小兵.基于传感器信息融合的移动机器人自主爬楼梯技术研究[J].传感技术学报,2005,18(4):828-833.