

程序行为分析指导 TLB 低功耗设计

史莉雯 樊晓桢 陈 杰 黄小平 郑乔石

(西北工业大学航空微电子中心 西安 710072)

摘 要 TLB(Translation Look-Aside Buffer, 变换旁视缓冲器)是存储管理单元中完成访存地址转换的核心。但研究发现 TLB 工作时可以消耗微处理器芯片约 17% 的功耗。因此, TLB 低功耗设计已经引起研究者的重视。通过对经典基准测试集程序访存行为的详细分析和仿真可知, 在页面非连续访问时, 页面间隔统计参数能够很好地指导 TLB 的低功耗设计。从这一角度出发, 提出了低功耗的 TLB 设计方法。实验结果显示, 改进后的 TLB 片上功耗明显降低。

关键词 变换旁视缓冲器, 低功耗, 非连续访问, 页面间隔

中图法分类号 TP303 **文献标识码** A

Instructing Low-power TLB Design by the Analysis of Program Behavior

SHI Li-wen FAN Xiao-ya CHEN Jie HUANG Xiao-ping ZHENG Qiao-shi

(Aviation Microelectronic Center, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract Translation Look-Aside Buffer(TLB) is a dedicated hardware component that the Memory Management Unit(MMU) utilizes to improve page address translation speed. However, some researchers indicate that working TLBs may occupy as much as 17% of a processor's total power consumption. The objective of this paper is to effectively reduce the TLB on-chip power consumption by looking into the program behavior in respect of page access traits. With careful analysis of the memory access patterns of SPEC CPU benchmarks, we demonstrated that the Page Interval which non-sequential page accesses heavily exhibit can be used to largely reduce the power of TLBs. Based on this observation, we proposed a novel low-power TLB design methodology. Experimental results show that using our design the on-chip power consumption can be further saved.

Keywords Translation look-aside buffer(TLB), Low power, Non-sequential accesses, Page interval

TLB是页表入口 PTE 的片上存储结构, 尽管容量不大, 却消耗了大量的能量。如在嵌入式处理器 Strong ARM110 中, TLB 的功耗占据了处理器全部功耗的 17%^[1]; 在 IBM 的 PowerPC, Hitachi 的 SH-3 等处理器中, TLB 也都是功耗消耗较大的部件之一。一些设计者为了追寻更高的命中率, 对 TLB 的结构不断改进, 然而代价却是更为巨大的能量消耗。功耗的增长为处理器的散热、系统稳定以及成本等带来了一系列的难题。目前, 低功耗与高性能一样也成为微处理器设计者努力追求的目标。如何在性能和功耗之间寻求一个最佳的折中更是成为了设计成功与否的关键因素。

1 相关研究

针对低功耗的 TLB 设计, 先前的研究已经提出了一些改进的方法。一种常见的思路是通过减少比较表项来降低功耗。最简单的方法是使用一个深度为 1 的寄存器来保存上一次访问的 TLB 表项, 但是这种方法主要针对连续访问指向同

一页面的情况^[2]。文献[3]中提出的 Banked Associative TLB 结构将 TLB 分为两个 bank, 查找 TLB 时则利用逻辑地址页号中的某几位选择一个 bank, 再用其余的位进行标记位的比较。因为 bank 只有 TLB 一半的容量并且每次只访问一个 bank, 所以功耗减少大概在 50% 左右。但是这种方法的一个主要缺点是容量缺失可能会导致性能降低。Filter TLB^[4-6] 的出现降低了功耗的同时, 相比 banked TLB 也减少了缺失率。它的基本设计思想是将整个 TLB 空间结构分为 filter TLB 和 main TLB, 再利用 filter TLB 中的两位地址来选择性地访问 filter TLB 和 main TLB。这种方法也属于通过硬件控制并减少同时访问的入口数来降低功耗。尽管如此, 无论是 banked TLB 或是 filter TLB 结构, 对于一个 TLB 体的所有入口全部参与比较, 与降低的功耗相比, 所用的硬件开销较大, 因此文献[7]结合 Block Buffering 技术提出了一种预比较的 TLB 结构。该结构将 CAM 分成两级比较, 第一级采用较少位参与比较, 快速筛掉不会命中的项, 确定出 CAM 中可能

到稿日期: 2010-06-21 返修日期: 2010-10-27 本文受国家自然科学基金重点项目(60736012), 国家自然科学基金项目(60773223)以及国家“863”高技术研究发展计划基金项目(2009AA01Z110)资助。

史莉雯(1982—), 女, 博士生, 主要研究方向为计算机系统结构, E-mail: shilw1001@gmail.com; 樊晓桢(1962—), 男, 教授, 主要研究方向为计算机系统结构; 陈 杰(1985—), 男, 硕士生, 主要研究方向为计算机系统结构; 黄小平(1982—), 男, 博士生, 主要研究方向为计算机系统结构; 郑乔石(1984—), 男, 博士生, 主要研究方向为计算机系统结构。

会命中的项来参加第二级比较,从而降低全部比较时的项数。其它文献[8-10]也从 CAM 单元进行研究并改进,从而达到降低 TLB 功耗的目的。

与上面的研究不同,本文的研究从分析经典基准测试程序集的访存操作流入手,提出一个页面间隔的问题,即当前页面在经过多少不同页面之后会再次被访问(这里并不考虑每个页面被连续访问的次数)。从对页面非连续访问这一角度出发,探讨非连续访问中页面间隔数的平均分布情况,根据研究结果提出低功耗的 TLB 设计方法。

2 仿真与改进策略

2.1 程序行为分析

本文选用 SimpleScalarV3.0^[11] 中的时钟精确仿真器 Sim-Outorder 作为平台,内核参数为默认设置^[12]。为了适应仿真需要,对 Sim-Outorder 的存储管理单元部分进行了修改,以便提取和统计与页面间隔相关的参数。仿真程序则从 SPEC CPU 95 和 SPEC CPU2000 基准测试集中选取。

首先对相邻两次访问在同一页面的比率进行分析,结果如图 1 所示。对于指令访问,相邻两次访问在同一页面的比率很高,平均达到 95% 以上;对于数据访问,相邻两次访问在同一页面的比率远不如指令,平均大概仅有 52% 左右,这就说明对于数据,大多数的连续访问指向不同的页面,页面局部性较差。文献[13]中也提到,虽然整体来看程序访存行为有很好的页面局部性,但连续的访存操作序列并不具备明显的页面局部性。

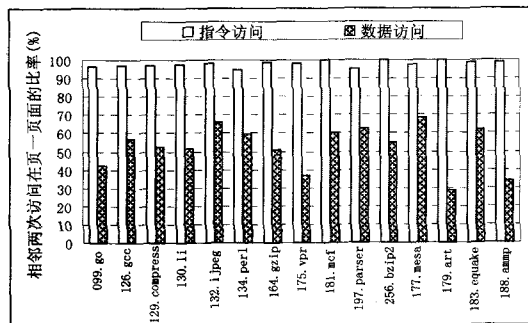


图1 相邻两次访问在同一页面的比率

随后令这些测试程序依次输出访存地址,经过分析发现,无论是指令或是数据,一个被访问过的页面在经过一些不同页面访问之后再次被访问到的比率很高。对于数据访问,该特性更加明显。那么,页面一般在经过多少次不同页面访问之后会再次被访问呢?

假设在某个时间段,追踪到的连续页面访问流为:AAAABBBCCDAACCB,如图2所示。在不考虑每个页面被连续访问的次数后,得到的就是这样这样一个页面访问序列:ABCDACB,在这个序列中前后两次对A的访问中间夹着对页面B,C和D的访问,因而这种对页面A的访问为非连续的访问。定义某个页面距离下一次非连续访问经过的不同页面个数为页面间隔。这里,页面A再次被访问的页面间隔为3,即在经过页面B,C和D之后又重新被访问;同理,页面C再次被访问的间隔为2。值得注意的是,对页面B的两次访问之间间隔了页面C,D,A和C,由于我们计算的是不同的页面个数,两次对C的非连续访问是同一页,因而页面B再次访

问的间隔为3,间隔了页面C,D和A。

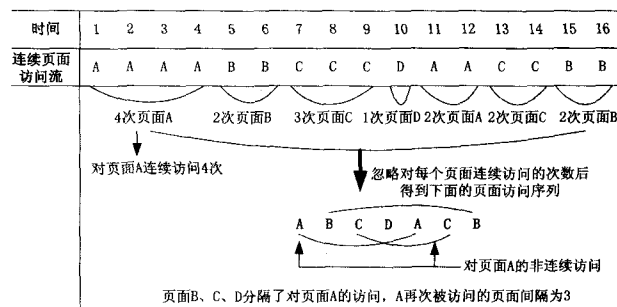


图2 页面间隔的定义举例

我们详细统计了每个测试程序的各种页面间隔的比率分布,如图3和图4所示,图中d代表了页面间隔。对于指令访问,从整体来看,页面在间隔1个不同页面后再次被访问的情况平均约为50.6%左右,间隔为2平均约占到13.5%左右,间隔为4的情况相对来说最少,平均只有4.5%。而对于数据访问,前后两次非连续访问之间页面间隔为1的情况达到了平均62.1%,页面间隔为4仅占了2.2%,依然是最少的情况。页面间隔大于16的情况也只有2.8%。

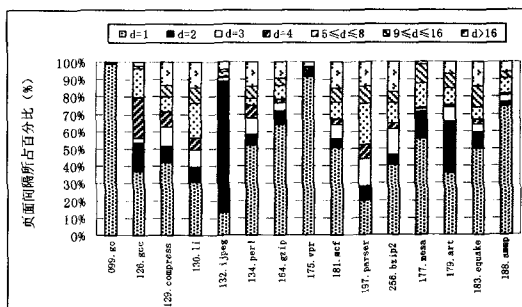


图3 页面间隔数所占的百分比(指令访问)

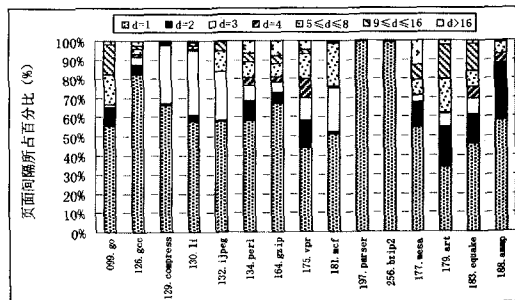


图4 页面间隔数所占的百分比(数据访问)

因此,当采用文献[2]中设计的深度为1的寄存器会经常发生替换颠簸,特别是在程序需要交替访问存放在不同页面上数据的情况时(如数组复制,两个数组在不同页面上,这时页面访问序列就会为ABABAB...且每个页面每次仅访问一次),不停地替换颠簸可能导致寄存器中没有一次命中,寄存器实际并没有发挥多大的作用。和文献[2,14]一样,不少研究者都把目光放在了针对相同页面被连续访问多次的特性上,但实际上除了这些连续的访问,还存在了很多对页面非连续的访问,从这一角度出发考虑改进TLB的设计能获得更好的结果。

2.2 TLB的改进策略

首先以“龙腾”R2处理器^[15]为例说明地址转换的过程。“龙腾”R2处理器通过逻辑地址的最高4位选择16个片上段

寄存器 SR,SR 中所包含的段描述符用来控制页地址转换。转换过程中,逻辑地址先按段描述符的信息转换成一个 52 位的中间虚拟地址。随后,中间虚拟地址中高 40 位的虚页地址段转换为对应的 20 位实页地址段,最后拼接上表示页内地址中的 12 位低地址,生成 32 位物理地址用于访存。

一般来说,地址转换时,如果两个逻辑地址高位相同,说明它们在同一页中,那么这两个地址经过转换后的物理地址的高位也是相同的。考虑到对页面的非连续访问特性以及页面间隔数的分布范围,可以为指令 TLB 和数据 TLB 分别设置深度为 n 的历史访问缓冲区,用来保存当前指令/数据访问的逻辑地址高位以及相对应的经过转换的物理地址高位。当有新的访问到达时,其地址高位首先与缓冲区中所保存的逻辑地址高位比较,如果相同,则直接从缓冲区中取出相应的物理地址高位,与当前访问的地址低位直接拼接形成完整的物理地址。这样,无论是对页面连续的访问,还是经过一定页面间隔后的非连续的访问,其地址转换大多可以在历史访问缓冲区中实现,减少了对 TLB RAM 的重复访问。另外,这种方法在比较后直接进行地址拼接,也加快了地址转换过程。

3 性能评估

3.1 缓冲区深度的研究

要降低 TLB 的片上访问功耗,就要尽可能提高历史访问缓冲区的命中率。缓冲区的深度如果过小,命中率不高,缓冲区可能遭遇频繁替换,TLB RAM 依然需要被频繁访问;如果深度过大,那么实现代价增加,对命中率的改进也并不一定有很大帮助。根据前面的实验分析已知非连续访问的平均页面间隔超过 16 的情况较少。因此将缓冲区的深度 n 分别设为 1,2,4,6,8,16,并分别记录缓冲区的命中情况。缓冲区暂采用简单的 FIFO 替换算法。

图 5 为指令 TLB 历史访问缓冲区的命中率。从图中可以看出,由于指令的页面局部性良好,当缓冲区深度设置为 1 时,平均获得了 98% 的命中率;当深度加深到 2 时,已经获得了平均 99% 以上的命中率,继续加深时,命中率变化已经不大。图 6 为数据 TLB 历史访问缓冲区的命中率,与指令访问不同,当缓冲区深度设置为 1 时,只获得平均 58% 左右的命中率,而一些测试程序如 vpr,art 和 ammp 仅仅获得了 20%~30% 的命中率。当缓冲区深度从 1 逐渐加深到 6 时,命中率提升速度很快,深度为 6 时已经获得了平均 97% 的命中率。继续加深深度到 16 时,命中率提升已经不明显,仅仅获得了 1% 左右的提升。实验结果显示,由于指令访问和数据访问的特性不同,在实际应用中可以对其分别设置不同深度的历史访问缓冲区。

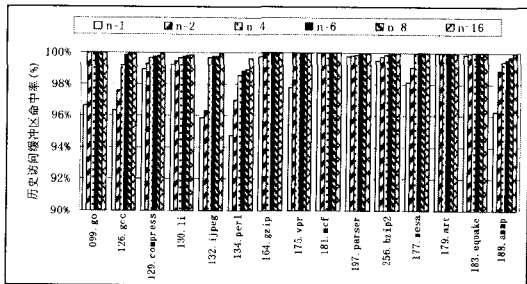


图 5 指令 TLB 历史访问缓冲区的命中率

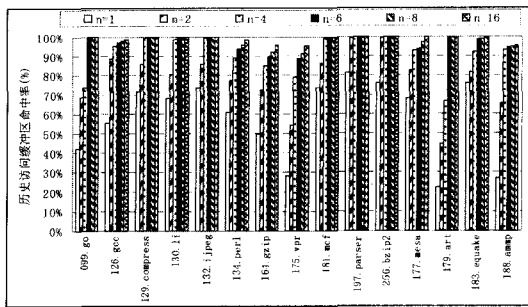


图 6 数据 TLB 历史访问缓冲区的命中率

3.2 缓冲区替换策略的研究

下面研究替换策略对历史访问缓冲区命中率的影响。我们选择先进先出策略(FIFO)、最近最少使用策略(LRU)和随机法(RANDOM),通过运行基准测试程序分别统计这 3 种替换策略下缓冲区的平均命中率。图 7 是指令 TLB 历史访问缓冲区替换策略的比较,图 8 是数据 TLB 历史访问缓冲区替换策略的比较,横轴表示缓冲区深度。从图中可以看出这 3 种替换策略下所获得的命中率相差并不是很大,平均差距仅在 1%~2% 左右。因而替换策略对于历史访问缓冲区命中率影响并不大。

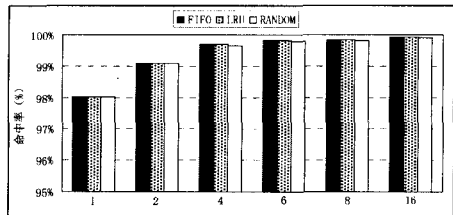


图 7 指令 TLB 历史访问缓冲区替换策略的比较

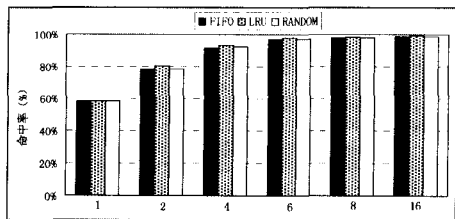


图 8 数据 TLB 历史访问缓冲区替换策略的比较

3.3 功耗的降低

为了进行功耗的评估,设计中 TLB RAM 和历史访问缓冲区分别采用由 Artisan 提供的 SMIC 0.18 μ m 的 RAM 模型和寄存器模型。因为 Artisan 只能生成深度为 4 以上的寄存器模型,无法生成深度为 1 和 2 的寄存器模型,所以采用数据处理软件 Origin7.0 对已生成的寄存器模型中与功耗计算相关的数值进行公式拟合,进一步得到以寄存器深度为自变量的寄存器功耗近似计算式(1)和式(2):Yread 表示对寄存器进行读操作时产生的功耗;Ywrite 表示对寄存器进行写操作时产生的功耗;变量 x 表示寄存器深度。根据拟合出的公式可以计算出深度为 1 和 2 时,对寄存器执行读/写操作产生的近似功耗。

每次读/写操作产生的平均功耗计算公式为:

$$Y_{read} = -0.00265x^2 + 0.08798x + 13.39191 \quad (1)$$

$$Y_{write} = -0.00262x^2 + 0.10124x + 13.85739 \quad (2)$$

参考文献[7]的方法来计算功耗:

未增加历史访问缓冲区的功耗计算如式(3):

$$PTLB = R_{TLB_hit} \times P_{TLB_hit} + R_{TLB_miss} \times P_{TLB_miss} \quad (3)$$

增加历史访问缓冲区之后的功耗计算如式(4):

$$P'_{TLB} = R_{buffer_hit} \times P_{buffer_hit} + R'_{TLB_hit} \times P'_{TLB_hit} + R_{TLB_miss} \times P_{TLB_miss} \quad (4)$$

式(3)中, R_{TLB_hit} 和 P_{TLB_hit} 分别表示 TLB 的命中率 and 命中功耗; R_{TLB_miss} 和 P_{TLB_miss} 分别表示 TLB 的缺失率和缺失时的功耗开销。式(4)中, R_{buffer_hit} 和 P_{buffer_hit} 分别表示历史访问缓冲区的命中率和命中功耗; R'_{TLB_hit} 是请求在历史访问缓冲区中缺失而在 TLB 中命中的比率, P'_{TLB_hit} 则是此时的功耗, 在这种情况下, 由于需要将在 TLB 中命中的数据写入缓冲区, 因此这部分功耗还需要包含写缓冲区的功耗; 式(4)中 R_{TLB_miss} 和 P_{TLB_miss} 的定义与它们在式(3)中一样, 依然分别表示 TLB 的缺失率和缺失时的功耗开销。值得注意的是, 我们的设计目标不在改善 TLB 原本的缺失情况, 因此式(3)和式(4)中 TLB 的缺失率是相同的, 片外访问功耗也相同。本文的目标主要是减少对 TLB 的访问次数, 从而降低片上功耗, 因此功耗的比较主要针对片上访问这部分, 并不考虑片外访问功耗, 结果如图 9 和图 10 所示。

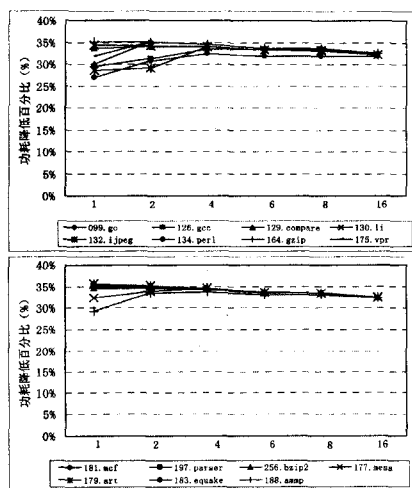


图 9 优化后片上访问功耗的降低(指令访问)

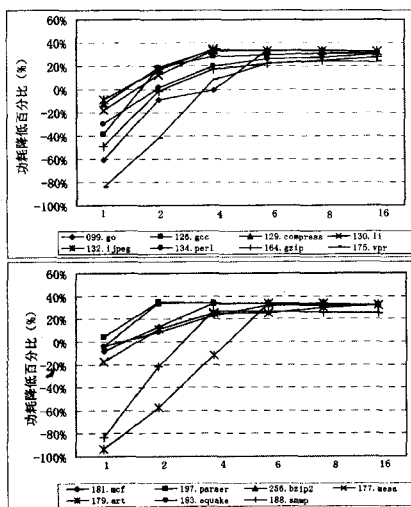


图 10 优化后片上访问功耗的降低(数据访问)

因为测试程序较多,为了清晰地表达数据,将每个图都划分为两部分,图中横轴为缓冲区深度。图 9 为指令访问时,较之原先的 TLB 片上访问功耗,在增加历史访问缓冲区后片上功耗所降低的百分比。从图中可以看出,缓冲区深度设置为

1 时,平均功耗已经降低 32% 左右,当深度加至 2 时,平均功耗降低达最大,约为 34% 左右。参照图 5,当缓冲区设置为 2 时已经获得平均约 99% 以上的命中率,继续加深时所获得的命中率相当,但是缓冲区深度加深,相应的读/写功耗代价也会增加。因而,在图 9 中,到深度为 4 后,功耗降低百分比呈略微下降趋势。

而对数据的访问结果如图 10 所示,当缓冲区深度设置为 1 时,除了 parser,所有的测试程序功耗降低均为负增长,也就是说,功耗非但没有降低,反而增加了,一些程序几乎增加了接近一倍左右。让我们再对比一下图 6,在缓冲区深度为 1 时,平均命中率并不高,如 vpr 的命中率只有 28% 左右,这就意味着 72% 左右的请求还要再次访问 TLB;由于这部分请求在缓冲区中缺失,还要产生缓冲区替换操作,因而这些操作所产生的片上功耗远远高于原先的 TLB 的片上功耗,因此就出现图中的负增长,其它程序亦是如此。这也就说明了,若缓冲区深度设置不足以捕获大部分访问时,缓冲区对于功耗的降低不仅起不到帮助,反而会出现负面影响。当缓冲区逐渐加深时,图中曲线上升趋势很快,缓冲区对于功耗降低的作用逐渐呈现出来。当深度加深到 6 时,平均功耗降低已达最高,达到 30% 左右,对比图 6 我们发现,此时缓冲区命中率也基本达到最高。继续加深深度时,平均功耗已经变化不大,由于缓冲区加深也增加了相应的读/写功耗,因而一部分程序功耗降低百分比也呈略微下降趋势。

实验表明,对于指令访问,缓冲区深度为 2 时,平均功耗降低率已达到最大值;对于数据访问,缓冲区深度为 6 时,平均功耗降低率已达到最大值。

结束语 本文从分析经典基准测试程序的访存特性入手,结合先前的一些研究,对低功耗的 TLB 设计进行了探索。根据指令访问和数据访问的不同特点以及页面间隔统计参数,提出为指令 TLB 和数据 TLB 分别配置不同深度的历史访问缓冲区,保存最近访问的地址的高位以及经过转换后的物理地址的高位,使得大部分地址转换尽可能在缓冲区中完成,减少了对 TLB RAM 的重复访问,从而降低了 TLB 的片上访问功耗。实验结果显示这种方法对于降低 TLB 的片上访问功耗有较大的帮助。

参考文献

- [1] Santhanm T S. StrongARM SA110, a 160mhz 32b 0.5w CMOS ARM processor. In Hot Chips 8, Aug. 1996
- [2] 黄海林,范东睿,许彤,等. 嵌入式处理器中访存部件的低功耗设计研究[J]. 计算机学报, 2006, 29(5): 815-821
- [3] Manne S, Klausner A, Grunwald D C, et al. Low Power TLB Design for High Performance Microprocessors[R]. Univ. of Colorado, 1997
- [4] Choi J-H, Lee J-H, Jeong S-W, et al. A Low Power TLB Structure for Embedded Systems[J]. IEEE Computer Architecture Letters, 2002, 1(1)
- [5] Choi J-H, Lee J-H, Park G-H, et al. An advanced Filter TLB for Low Power Consumption[C]// Proceedings of the 14th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD. 02). Oct. 2002: 93-99
- [6] Lee J-H, Park G-H, Park S-B, et al. A Selective Filter-Bank TLB System[C]// Proceedings of the 2003 International Symposium on Low Power Electronics and Design. Aug. 2003: 312-317
- [7] 侯进永, 刑座程. 一种低功耗预比较 TLB 结构[J]. 国防科技大学学报, 2006, 28(5): 84-89

- [8] Lin Chi-sheng, Chang Jui-chuan, Liu Bin-da. A Low-Power Pre-computation-Based Fully Parallel Content-Addressable Memory [J]. IEEE J. Solid-State Circuits, 2003, 38(4): 654-662
- [9] Liu S C, Wu F A, Kuo J B. A novel low-voltage content-addressable-memory(CAM) cell with a fast tag-compare capability using partially depleted(PD) SOI CMOS dynamic-threshold(DT-MOS) techniques [J]. IEEE J. Solid-State Circuits, 2001, 36: 712-716
- [10] Lin P F, Kuo J B. A 1-V 128-kb four-way set-associative CMOS cache memory using wordline-oriented tag-compare (WOTC) structure with the content-addressable-memory(CAM) 10-transistor tag cell[J]. IEEE J. Solid-State Circuits, 2001, 36: 666-675

(上接第 278 页)

区域检测和空间分割。从图 5 中可以看出,由于水纹和船上小旗对时域运动区域检测的影响,分割结果虽然能够完整提取出前景对象,得到对象的主要轮廓,但是边缘定位还不够准确,有待于进一步的提高。Coastguard 序列由于进行了全局运动补偿,处理速度约为 12 帧/s,基本满足实时性。

图 6 是 Akyio 序列的分割结果,Akyio 序列是背景静止的头肩序列,时域检测的运动对象区域准确,空间也得到了较好的分割,所以得到了较好的分割结果。Akyio 序列不用进行全局运动补偿,空间分割区域合并量小,处理速度约为 26 帧/s,满足实时性。

图 7 中,Silent 序列是背景复杂、局部运动快速的图像序列,而 Claire 序列是背景简单的头肩序列,运动较缓慢,从实验结果来看,本文的算法都得到了准确的视频运动对象分割结果。而且本文方法的准确性高于 Kim 提出的方法。

结束语 提出了一种在通用视频序列中联合时空信息分割运动对象的算法。对于运动背景,采用匹配加权的全局运动估计算法,消除了背景全局运动的影响;在时域检测中,使用基于直方图拟合的显著性检测及对称差分法来获得运动对象模板,克服了依据经验设定阈值的缺点,提高了运动对象模板的准确性;在空间区域分割中,对分水岭算法从形态重建滤波、多尺度形态梯度计算、粘性形态学修正及区域合并等方面进行了改进,克服了分水岭算法易受噪声影响及过分割问题,得到了有效的区域分割;最后使用双阈值比重法提取出视频运动对象。从实验结果来看,本文算法对运动背景、静止背景和复杂背景的图像序列都能够进行较好的分割,实时性较好,而且,通过对彩色图像每个像素的 R、G、B 取均值平均后,该算法也可用于彩色图像。但是,由于运动补偿的精度有限,在纹理丰富的背景区域和明显边缘处,会有大的运动误差,对运动背景视频图像序列的分割结果形成了一定影响,因此时域

(上接第 289 页)

- [5] The 35th Top500 List [EB/OL]. <http://www.top500.org>, 2010
- [6] Franck Cappello INRIA. Fault Tolerance & PetaScale Systems; Current Knowledge, Challenges and Opportunities [C] // EuroPVM/MPI 2008. LNCS 5205. Berlin Heidelberg: Springer-Verlag, 2008
- [7] Plank J S, Li Kai, et al. Diskless Checkpointing. IEEE Transaction on Parallel and Distributed Systems [J]. 1998, 9(10): 972-986
- [8] Chen Zi-zhong, Fagg G E, Gabril E, et al. Building Fault Survivable MPI Programs with FT-MPI Using Diskless Checkpointing

- [11] Austin T, Larson E, Ernst D. SimpleScalar. An infrastructure for computer system modeling [J]. IEEE Computer, 2002, 35(2): 59-67
- [12] Burger D, Austin T M. The SimpleScalar Tool Set [R]. CS-TR-97-1342, 1997
- [13] 张浩, 林伟, 周永彬, 等. 通用处理器的高带宽访存流水线研究 [J]. 计算机学报, 2009, 32(1): 142-150
- [14] Clark L T, Choi B, Wilkerson M. Reducing Translation Lookaside Buffer Active Power [C] // ISLPED. 2003: 10-13
- [15] 屈文新, 樊晓桢. “龙腾”R2 微处理器存储管理单元的设计与实现 [J]. 西北工业大学学报, 2007, 25(1): 137-141

分割准确性还需要进一步改进。如何进一步改进全局运动估计算法的准确性,提高运动对象分割结果还应做进一步研究。

参考文献

- [1] 张泽旭, 李金宗, 李宁宁. 基于光流场分割和 Canny 边缘提取融合算法的运动目标检测 [J]. 电子学报, 2003, 31(9): 1299-1302
- [2] 彭小宇, 杨明, 邹北骥, 等. 基于局部图金字塔的不规则块匹配视频分割方法 [J]. 计算机科学, 2008, 35(4): 233-237
- [3] Chien S Y, Huang Y W, Hsieh B Y, et al. Fast video segmentation algorithm with shadow cancellation [J]. IEEE Trans. on Circuits Systems for Video Technology, 2004, 14(5): 732-748
- [4] 张晓波, 刘文耀, 吕大伟. 基于时空信息的自动视频对象分割算法 [J]. 光电子激光, 2008, 19(3): 384-387
- [5] 杨高波, 张兆扬, 余圣发. 一种基于小波分解和分水岭变换的视频对象自动分割算法 [J]. 通信学报, 2005, 26(3): 7-13
- [6] 李宏, 李翔, 胡可成. 基于时空融合的视频分割算法研究 [J]. 信号处理, 2009, 25(1): 72-76
- [7] Tsaig Y, Averbuch A. Automatic segmentation of moving object in video sequences; a region labeling approach [J]. IEEE Trans on Circuits and Systems for Video Technology, 2002, 12(7): 597-612
- [8] Aach T, Kaup A, Mester R. Statistical model-based change detection in moving video [J]. Signal Processing, 1993, 31(2): 165-180
- [9] Meryer F, Vachier C. Image segmentation based on viscous flooding simulation [C] // Proceedings of ISMM 2002. CSIRO, Sydney, 2002: 69-77
- [10] Vincent L. Morphological gray scale reconstruction in image analysis; applications and efficient algorithms [J]. IEEE Trans. on Image Processing, 1993, 12(2): 176-201
- [11] Kim C, Hwang J N. Fast and automatic video object segmentation and tracking for content-based application [J]. IEEE Trans on Circuits and Systems for Video Technology, 2002, 12(2): 122-129

- [C] // Proceedings of the 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP). Chicago, Illinois, 2005: 213-223
- [9] Vaidya N H. A Case for Two-Level Distributed Recovery Schemes [C] // ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems. Ottawa, Canada, 1995: 65-73
- [10] Hwang K, Hai Jin, Chow E, et al. Designing SSI Clusters with Hierarchical Checkpointing and Single I/O Space [J]. IEEE Concurrency, 1999, 7(1): 60-69
- [11] Oracle. Lustre [EB/OL]. <http://www.sun.com/software/products/lustre/index.xml>, Jun. 2010