

# 一种基于活跃周期的低端口数低能耗寄存器堆设计

赵雨来 李险峰 佟冬 孙含欣 陈杰 程旭

(北京大学信息科学技术学院 北京 100871)

**摘 要** 多端口寄存器堆有助于挖掘指令级和线程级并行性,但同时带来面积、能耗和访问时间的压力.文章面向超标量和 SMT 处理器,给出了一种方法,即通过增加一个小的活跃值堆(Active Value File, AVF)选择性地保存处于活跃周期(从产生到最后一次使用之间)的物理寄存器值. AVF 结构可分担主寄存器堆的访问压力并降低端口数目,实现简单且具有写过滤的特点.在获得较大幅度能耗降低的同时不影响时钟频率且 IPC 损失较小.

**关键词** 物理寄存器堆;寄存器重命名;寄存器生命周期;乱序执行;SMT

**中图法分类号** TP303

## Active-Cycle Based Register File Design for Reduced Ports and Energy

ZHAO Yu-Lai LI Xian-Feng TONG Dong SUN Han-Xin CHEN Jie CHENG Xu

(School of Electronics Engineering and Computer Science, Peking University, Beijing 100871)

**Abstract** Multi-ported register file helps exploiting instruction-level and thread-level parallelism but brings area, energy and access time pressure. Oriented for superscalar and SMT processor, this paper give a method that is to manage a small auxiliary active value file (AVF) and selectively store physical register values in active cycle (during the time between production and last use). The AVF structure can share the register file access pressure and reduce the number of register ports, is simple to implement, and can filter some writes. It achieves significant energy savings with no impact on frequency and only small IPC loss.

**Keywords** physical register file; register renaming; register life time; out-of-order execution; SMT

## 1 引 言

物理寄存器堆(Register File, RF)已经成为高性能微处理器的主要功耗来源,并且随着发射宽度和指令窗口大小增加而增加. RF 必须采用多端口以支持宽发射,大小也必须能够支持足够多的动态指令.例如在 8-发射的 Alpha 21464 处理器中,512-项/16-读/8-写的寄存器堆是 64KB 的一级 DCache 面

积的 5 倍,并且访问更频繁<sup>[1]</sup>. 为了降低对于周期时间的影响,用两个 8-读/8-写的复制寄存器堆实现,称为双复制 RF 结构. 每个周期产生的值被同时写到两个 RF 当中,每条指令可以从任意一个 RF 读取值,因此从外部看来,仍然是一个 16-读/8-写的 RF. 单一 RF 结构访问较慢,因为每个比特单元必须驱动更多的位线. 双复制 RF 结构访问快,但是复制 RF 的内容到两个存储阵列对于功耗有负面影响,因为所有动态值必须回写到两个备份当中. 这种

收稿日期:2006-12-05;最终修改稿收到日期:2007-09-04. 赵雨来,男,1980 年生,博士研究生,主要研究方向为计算机体系结构、性能评测、软硬件协同设计和系统芯片. E-mail: zhaoyulai@mprc.pku.edu.cn. 李险峰,男,1973 年生,博士,主要研究方向为微体系结构、实时系统和系统芯片. 佟冬,男,1971 年生,副教授,主要研究方向为计算机体系结构、存储系统、互连网络和系统芯片. 孙含欣,1980 年生,博士研究生,主要研究方向为计算机体系结构、软硬件协同设计和处理器验证. 陈杰,男,1979 年生,博士研究生,主要研究方向为计算机体系结构、功耗评测和低功耗设计. 程旭,男,1967 年生,博士,教授,博士生导师,主要研究领域为高性能微处理器、系统芯片、嵌入式系统、指令级并行、优化编译和软硬件协同设计.

权衡在最近的高性能微处理器设计中很普遍. 无论哪种情况, 较多的表项和端口数目均会导致高功耗、访问慢和面积大. 从功耗和面积而言, 降低端口数比降低表项数目效果更为显著.

同时多线程 (Simultaneous Multithreading, SMT) 处理器可以同时从多个无关的线程中取指, 并以共享数据通路资源的方式同时执行这些线程的指令以最大化资源利用率和吞吐率. 特别的, 物理寄存器堆一般被多个线程共享, 只要有充足的物理寄存器, 就可以为任意线程中的指令重命名. SMT 处理器对于数据通路资源的需求以及运行时刻的占用率一般较高. 对于支持  $N$  个线程, 指令窗口大小为  $Winsize$  的 SMT 处理器, 假设每个上下文需要 32 个整点和 32 个浮点状态, 则 RF 大小必须达到  $N \times 64 + Winsize$  表项才能避免分派级的暂停. 如果 RF 的大小小于该数目, 那么在没有空闲寄存器时, 必须暂停向 ROB 填充指令. 例如, 8-发射的 Alpha 21464 支持 4 个线程, 512-表项 RF 包含 4 个线程状态, 其中 256-表项存放提交值, 另外 256-表项存放推测值.

以往, 关于高性能微处理器的低能耗数据通路方面的研究多关注单线程处理器, 利用单线程负载的特性降低数据通路能耗. 由于多线程共享和竞争数据通路资源, 很多技术不适合于 SMT 处理器. 而作为一种片上多线程技术, SMT 已经成功应用于商业处理器, 并被广泛研究. 因此我们开展了面向支持 SMT 超标量处理器的低能耗数据通路设计研究<sup>[2]</sup>. 在该研究背景下, 本文提出一种有效降低 RF 端口数和能耗的方法. 该方法基于对物理寄存器生命周期的观察, 即在物理寄存器的整个生命周期中, 真正有用的区间为值产生到最后一次被使用, 我们称之为活跃周期. 通过增加一个小的活跃值堆 (Active Value File, AVF), 可以有效保存当前处于活跃周期的物理寄存器值, 分担主寄存器堆的访问压力, 从而降低主寄存器堆的端口数目和能耗. AVF 可以分担主寄存器堆的访问压力, 从而降低主寄存器堆的端口数目. 和目前主流的双复制 RF 结构相比, 由于大量的数据可以通过访问小寄存器堆 AVF 获得, 本文提出的新的寄存器堆结构运行时刻可以获得较大幅度的能耗降低, 并且保持时钟频率不变, 设计复杂度略有增加, 而 IPC 损失也较小.

本文第 2 节介绍相关工作; 第 3 节提出设计动机; 第 4 节描述新结构的设计与实现; 第 5 节进行实验分析; 第 6 节进行总结.

## 2 相关工作

关于寄存器堆的设计优化研究主要包括以下 3 个方面:

第 1 类研究关注于降低 RF 的大小, 通常面向一些发射宽度适中而指令窗口较大的微体系结构. 这类技术包括推迟物理寄存器分配<sup>[3]</sup>, 利用处理器或 RF 断点提早释放寄存器<sup>[4-5]</sup>, 将多个窄位宽的整点值压缩<sup>[6]</sup>, 将窄位宽的整点值内联到映射表<sup>[7]</sup>. 这些优化总体而言与我们的优化是独立的, 可以相结合.

第 2 类研究解决随指令窗口和发射宽度增长带来的 RF 的可扩展性问题. 可扩展性问题是指在未来更高主频的约束下, 使得处理器性能随 RF 的增长而增长. 这类技术包括管理 RF 层次<sup>[8]</sup>和使用多体多端口 RF<sup>[9]</sup>. 在层次式的结构中, 二级 RF 存储所有的值而一级 RF 存储一个子集并希望这些值会被后续指令使用. 研究人员提出了各种启发式策略, 利用负载访问寄存器的寻址局部性, 决定哪些值存入一级 RF 以获得更好的性能. 多体多端口 RF 使用较小的、端口数更少的体降低访问时间. 这种技术主要的复杂性在于需要额外的读写端口仲裁逻辑用于判定每个小 RF 体上潜在的端口冲突, 因此不可避免地要在发射级和读寄存器级之间增加流水级做仲裁. 仲裁级只能发现体冲突并暂停执行相冲突的指令, 但并不能防止冲突, 因为指令已经被发射. 由于小 RF 体上端口冲突的影响, 很多原本可以连续执行的指令将被暂停, 这对处理器性能是不小的损失.

第 3 类研究关注于降低 RF 的端口数目, 对于宽发射处理器效果更为显著. 这类技术包括前递指示和分离重命名<sup>[10]</sup>. Kim 扩展了前递指示技术, 采用延迟回写队列 (Delayed Writeback Queue, DWQ) 存储最近  $N$  个周期产生的值<sup>[11]</sup>. 其基于的观察是, 最近产生的值被后续指令消费的概率较大. 对于 8-发射的处理器而言,  $N$  个周期的 DWQ 采用  $8 \cdot N$  个表项. 所有动态产生的值都被写入 DWQ 和 RF, 而写入 DWQ 中的值在  $N$  个周期后被释放. 为发射队列 (Issue Queue, IQ) 的每个表项中的寄存器标签增加一个计数器, 在唤醒时刻初始化为  $N$ , 其后每个周期递减, 直至减到 0. 在发射级检测该计数器, 如果大于 0 则从 DWQ 读取寄存器值, 否则从 RF 读取. 需要在指令调度逻辑中判定 RF 和 DWQ 的

读请求数目是否超过读端口最大数目. DWQ 采用 CAM 结构实现,用寄存器标签寻址. DWQ 针对于 RF 设计,其工作原理类似于针对 L1 DCache 的 Load/Store 队列. 由于我们的技术也通过增加辅助性结构来减少 RF 端口和能耗,因此我们将在实验中与 DWQ 技术做比较.

3 物理寄存器的生命周期

编译器能够利用的是有限的逻辑寄存器. 编译器为中间语言使用的变量分配逻辑寄存器. 逻辑寄存器的生命周期从一条定义该寄存器的指令开始执行,到所有使用该定义的指令中最后一条执行完成. 对于依赖动态调度的超标量机器而言,编译器无法确切知道在哪个具体的 CPU 周期里逻辑寄存器被映射以及被重新映射.

我们研究分析了物理寄存器生命周期中的活动. 图 1 显示了生命周期中顺序发生的各个事件. 寄存器被分配以后,该寄存器不会被使用,直到结果回写入该寄存器. 只有回写之后,寄存器的值才可被具有依赖关系的指令消费(称为消费者). 一旦后一条改写相同逻辑寄存器号的指令(称为重定义者)完成重命名,并且最后一个消费者读取完寄存器的值,则该寄存器的维持只是用来在分支错误或异常情况下重建精确的处理器状态. 物理寄存器的生命周期被划分为 3 个阶段:(1)从分配到写入;(2)从写入到最后一次读完成;(3)从最后一次读完到释放. 我们将中间阶段称为活跃周期,称处于活跃周期的寄存器值为活跃值,而其他两个阶段称为静止阶段. 寄存器的读写都是在活跃周期进行,静止阶段不进行寄存器访问.

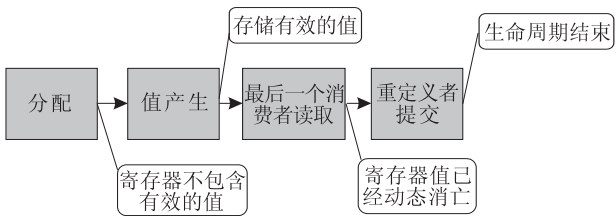


图 1 物理寄存器的生命周期

图 2 对 SPEC2000 单线程和多线程负载运行过程中寄存器的各个阶段的周期数进行了统计,活跃阶段平均分别为 10 和 12 个周期,而整个寄存器生命周期平均分别为 50 和 60 个周期. 寄存器活跃周期平均约占整个寄存器生命周期的 1/5.

利用寄存器值活跃周期较短的特性,可以管理

一个小结构,在值产生的时候选择性地冗余复制,而在确定该值不再使用的时候将其释放. 这样,相当一部分指令通过该结构获得寄存器值,从而降低主 RF 读端口的压力. 需要解决的问题是如何进行管理从而保证微体系结构实现软件是透明的,包括如何确定寄存器值的活跃阶段,如何保证推测式执行和精确中断的正确性等问题.

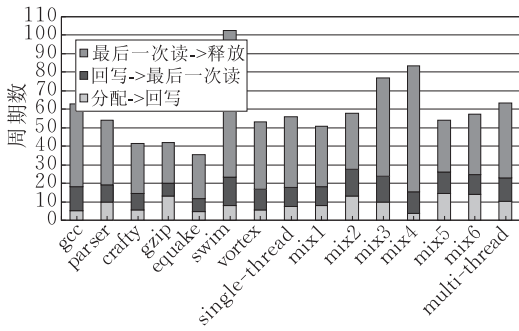


图 2 生命周期中各个阶段所花费的周期数

4 基于活跃值的选择性复制

4.1 选择策略

我们选择复制活跃值的条件有两个:(1)消费者存在;(2)重定义者已经完成重命名.

如图 3 所示,左边为原始代码,而右边是被处理器重命名过的代码. XOR 指令首先将 R3 映射到 P63,ADD 指令使用 R3(P63),SUB 指令重新将 R3 映射到 P64. 称 XOR 指令为 P63 的定义者,ADD 指令为 P63 的最后消费者,SUB 为 P63 的重定义者. SUB 指令将 R3 映射到新的物理寄存器 P64,也即重新定义了 P63. 这段代码在处理器中的执行如下:首先前端按指令顺序重命名,然后将指令派送到发射队列. 如果当 XOR 指令执行完回写 P63 的时刻,SUB 指令已经完成重命名,则可以断定,所有的消费者已经进入发射队列中. 因此只要 ADD 指令发射并读取 P63,则 P63 的活跃阶段结束.

原始代码		重命名代码	
XOR <b>R3</b> , R5, 50		XOR P63, P65, 50	
ADDR7, <b>R3</b> , 100		ADD P15, <b>P63</b> , 100	
SUB <b>R3</b> , R6, R8		SUB P64, P33, P40	
Num_Cons		1	
Renamed		1	
0		63	RF_size-1

图 3 关于选择策略的代码示例

#### 4.1.1 检测待发射的消费者

为每个物理寄存器维护一个 3-bit 的饱和计数器(称为 Num\_Cons),记录该寄存器的待发射消费者指令数目.在指令重命名时对源物理寄存器的计数器增 1,而当消费者读取该寄存器后计数器减 1. Num\_Cons 计数器在分支错误恢复处理时 also 做相应调整.

#### 4.1.2 检测重定义者

重定义者完成重命名,标志着物理寄存器不再有后续的消费者进入指令窗口.为了探测到重定义者,为每个物理寄存器维护一个 Renamed 比特位.在重命名级,每条指令对其目标逻辑寄存器所映射的旧的物理寄存器的 Renamed 位置 1.当物理寄存器被释放时,其相应的 Renamed 位置 0.

选择复制活跃物理寄存器  $P$ ,当且仅当满足

$$Renamed[P]=1, Num\_Cons[P]>0 \text{ 且}$$

$$Num\_Cons[P] \neq 7.$$

前两项条件保证  $P$  的所有消费者都在发射队列中等待,可以在唤醒时刻通知这些指令.受限于 Num\_Cons 计数器的位宽,当其达到饱和时,无法确切统计消费者的数目,因此增加最后一项条件避免复制该寄存器.在实际的负载程序中,多数情况下一个产生值有 1~3 个消费者,Num\_Cons 计数器几乎不会达到饱和.如图 2 所示的代码,XOR 将在最后一个执行周期检查上述条件,如果条件成立,则在寄存器回写级将 P63 同时写入 AVF 和 RF.对于所模拟的处理器和负载程序,大约 65%~83% 的动态寄存器值满足上述条件,意味着动态执行过程中有大量的值可以被选择性复制.

### 4.2 AVF 的管理

#### 4.2.1 结 构

图 4 左是宽发射处理器多采用的双复制 RF 结构,右是基于选择性复制的 RF 结构.双复制 RF 结构由两个 8-读/8-写的 RF 构成,从外部看来仍然构成 16-读/8-写.由于所有值都可以从任意的 RF 体中获得,因此同周期发射的 8 条指令不会产生体冲突,只要将同周期发射的指令分为 2 组.但是每周周期产生的结果需要同时回写到两个 RF 体.而基于选择性复制的 RF 结构中,使用一个非常小的 8-读/8-写的 AVF 结构代替了其中一个 RF 体.而不同的是,动态的值是选择性写入 AVF 当中的,并且每次读请求只能从 AVF 或 RF 获得.当 AVF 或 RF 的读请求超过最大端口数目时,则由于资源的限制只能延迟发射已经就绪的指令.而对于双复制 RF 结

构而言,只要发射的总指令数不超过最大发射宽度即可.因此,基于选择性复制的 RF 结构增加了端口资源的限制,从而会引起一些性能损失.基于延迟回写策略的 DWQ 类似于 AVF,也存在端口资源限制的问题,也会引起一些性能损失.不同于 AVF 的选择性写策略,DWQ 的工作方式要求动态产生的值同时写回 DWQ 和 AVF. AVF 和 DWQ 都需要采用 CAM 体实现,每个表项有一个 valid 位、tag 域和 data 域.

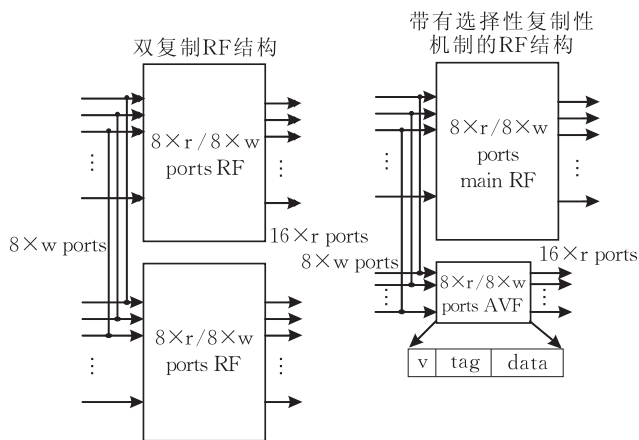


图 4 双复制 RF 结构与选择性复制 RF 结构

#### 4.2.2 AVF 分配与释放

物理寄存器的标志位在重命名级更新,而检测则在最后一个执行周期(即发射队列的唤醒阶段)进行.如果检测到寄存器满足 4.1 节所述条件,即所有消费者都在发射队列中,则需要通知所有消费者,从 AVF 当中获得该值.

修改后的流水线结构如图 5 所示.传统结构下,通过结果标签(result tag)与发射队列中的源操作数标签(src tag)进行比较设置 Ready 位.修改后的结构为发射队列中每个源操作数标签增加一个 A 位,标志该操作数是否从 AVF 中获得. A 位的设置过程如下:在驱动结果标签通过发射队列的 CAM 阵列,并与发射队列中的源操作数标签比较的同时(通常是唤醒逻辑延迟的主要部分)选取寄存器的标志位 Renamed 和 Num\_Cons,并扫描 AVF 的 valid 位向量,分配空闲表项.如果标志位满足条件且存在空闲 AVF 表项,则提升 active 信号,如图 6 所示是发射队列单个表项的唤醒逻辑电路.在唤醒的最后阶段,即设置 Ready 位阶段,active 信号与 match 信号通过一个与门设置 A 位.该逻辑的含义是:如果消费者都在发射队列中,并可为其分配 AVF 表项,则通知消费者从 AVF 获取相应操作数.指令调度(或称选择逻辑)负责分配读端口.通常在功能部件





在大指令窗口的超标量处理器中,不同的错误分支恢复技术对于推测路径指令的清除操作都是相同的,而区别主要在于对重命名表精确状态的重建. 本文假设利用前端 RAT,采用向后遍历(walking backward)方式恢复. 当检测到分支预测错误时,读入前端 RAT 表的当前状态,并从 ROB\_tail 指向的指令开始,遍历错误推测路径上的指令,并更新前端 RAT 表,直至到达错误推测分支的位置. 当错误推测指令更靠近 ROB\_tail 时,分支恢复的损失也越小. 一些研究表明,这种技术比其他技术性能更好,因此本文基于前端 RAT 向后遍历的恢复技术. 假设单个周期可以恢复 8 条指令并完成对 RAT 的更新,因此分支推测的失效损失取决于 ROB\_tail 与错误分支的距离. 在从 ROB\_tail 向后遍历 ROB 并利用 ROB 中推测路径指令信息恢复前端 RAT 的同时,进行上述标志位的更新和 AVF 表项的释放. 因此,恢复标志位与恢复前端 RAT 类似且并行完成,不会引起额外的性能损失.

5 实验分析

5.1 模拟工具与基准程序

我们使用 M-Sim 模拟器评估新的 RF 结构<sup>[12]</sup>. M-Sim 修改 SimpleScalar 3.0d<sup>[13]</sup>以支持 SMT 处理器模型,并且将 RUU 实现为分离的 IQ,ROB 和 RF. 在 SMT 执行模式下,各个线程共享 IQ,RF,功能部件和 Cache 系统,但是各个线程拥有独立的重命名表、PC 计数器、load/store 队列和 ROB. 每个线程拥有独立的分支预测器. 模拟的处理器参数如表 1 所示. 采用 I-Count 取指策略,每周期从 2 个线程取指.

表 1 类 Alpha21464 处理器模型

参数	配置
Machine Width	8-wide fetch, 8-wide issue, 8-wide commit
Window Size	128-entry issue queue, 64-entry load/store queue, 512-entry ROB
FU Latency (total/issue)	8 Int Add(1/1), 2 Int Mult(3/1)/Div(20/19), 2 Load/Store(2/1), 4 FP Add(2), 2 FP Mult(4/1)/Div(12/12)/Sqrt(24/24)
Physical Registers	512 physical registers
L1 I-Cache	64KB, 2-way associative, 128 Byte line
L1 D-Cache	64KB, 4-way associative, 256 Byte line
L2 Cache Unified	2MB, 8-way set-associative, 512 Byte line, 8 cycles hit time
Front-End	64 entry fetch queue, 2KB entry gshare, 10-bit global history per thread 2048-entry, 2-way set-associative BTB
Pipeline Structure	fetch, decode, rename, issue, register read, execution, write back, commit
Memory	128 bit wide first chunk 150 cycles, next chunk 2 cycles

我们选择了 7 个 SPEC2000 的整点和浮点基准程序作为单线程负载的代表,6 个基准程序的组合作为多线程负载的代表. 多线程负载包含了各种基准程序的组合,这些基准程序按照 ILP 的低、中、高进行分类,如表 2 所示. 各个基准程序的初始化部分被快速模拟跳过,并且执行 SimPoint 工具所推荐的 1 亿条指令构成的区间<sup>[14]</sup>. 对多线程负载,我们评测定量的任务,即当所有线程都执行完他们的模拟区间以后模拟器停止. 我们使用归一化的 IPC 评价性能损失. 实验采用了 4 种 RF 结构进行比较,包括双复制 RF 结构、N 表项的 AVF 结构、N 表项的 DWQ 结构以及 8-读/8-写的 RF 结构(SRF),如表 3 所示. 以 DRRF 结构作为基准对最终性能(IPC)归一化.

表 2 评测所采用的单线程和多线程负载测试程序

负载	说明
gcc	SPECint
parser	SPECint
crafty	SPECint
gzip	SPECint
equake	SPECfp
swim	SPECfp
vortex	SPECfp
mix 1(twof, vpr, swim, parser)	4 low ILP
mix 2(applu, ammp, mgrid, galgel)	4 med ILP
mix 3(wupwise, gzip, vortex, mesa)	4 high ILP
mix 4(mcf, equake, mesa, vortex)	2 low ILP+2 high ILP
mix 5(art, lucas, galgel, gcc)	2 low ILP+2 med ILP
mix 6(gzip, wupwise, fma3d, apsi)	2 med ILP+2 high ILP

表 3 实验采用的几种 RF 结构配置

结构配置	说 明
DRRF	16-读/8-写/512-表项 RF,分为两个 8-读/8-写的复制体
AVF-N	8-读/8-写/512-表项 RF 和 8-读/8-写/N-表项 AVF
DWQ-N	8-读/8-写/512-表项 RF 和 8-读/8-写/N-表项 DWQ
SRF	8-读/8-写/512-表项 RF

5.2 性 能

由图 8 所示的性能图表看到,如果采用单个 8-读/8-写的 SRF 结构(即发射宽度为 4),单线程负载的 IPC 损失平均为 27%,多线程负载的 IPC 损失平均为 35%. 对 DWQ 和 AVF 结构,我们分别采用了 16-和 24-表项进行比较. 采用 16-表项的 DWQ 或 AVF 结构,单线程负载的 IPC 损失下降到 3.1%和 3.4%,多线程负载的 IPC 损失下降到 4%. 而当采用 24-表项的 DWQ 或 AVF 结构,单线程负载的 IPC 损失下降到 2.6%和 2.5%,多线程负载的 IPC 损失下降到 3.4%和 3.2%. 无论采用 DWQ 或 AVF 结构,由于动态情况下指令对于端口资源需求

不平衡,会导致指令推迟发射. 总体而言,增加了辅助性结构以后都会比 SRF 结构性能大幅提高,而相对于 DRRF 结构的性能损失较小. 采用 DWQ 或 AVF 两者之间在 IPC 性能上差别较小,两种辅助性结构的利用率都很高. 由于读端口的总数仍然是

16 个,且发射槽并不总能充分利用,因此相对于 DRRF 结构性能损失较小. 以往的研究还表明,25%~50% 的指令允许推迟发射 3~5 个周期而不影响整体性能<sup>[15]</sup>.

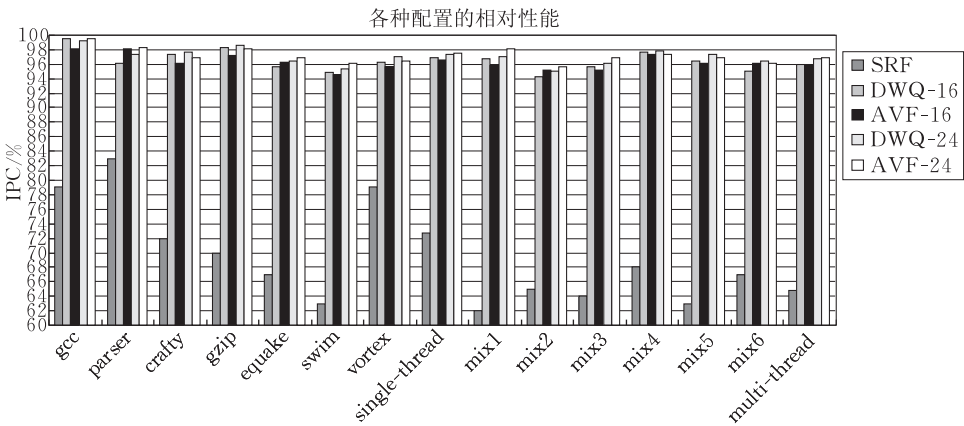


图 8 增加辅助性结构对 IPC 的影响

5.3 能 耗

为了比较能耗,首先需要评估各个存储组件单次访问(读或写)的能耗. 我们修改 CACTI 3.0 在 0.18μm 工艺参数下评估 RF 以及小的 CAM 结构的访问时间、能耗和面积<sup>[16]</sup>. CACTI 原来是为 Cache 设计的分析工具,与 Cache 不同的是,RF 不包含标签阵列. 表 4 给出的是各个组件的评估结果. 其中 16-读/8-写的 RF 由两个 8-读/8-写的复制体构成,AVF 或 DWQ 都以 CAM 体实现. 由表 4 的结果可知,由于 CAM 体非常小,其访问时间不会超过 RF 的访问时间,因此不会降低时钟频率. 而小 CAM 体读或写的能耗大概只有 8-读/8-写 RF 的 1/4,面积只有后者的 1/6. 减小面积对于缩短其他组件之间的互连线长度也是有益处的.

表 4 各种存储组件在 CACTI 3.0 的 0.18μm 工艺下的访问时间、能耗和面积估计

存储组件	访问时间/ns	能耗/nJ	面积/cm <sup>2</sup>
512-表项 16-读/8-写 RF	1.47	14.8	0.80
512-表项 8-读/8-写 RF	1.47	7.4	0.40
16-表项 8-读/8-写 CAM	1.28	1.9	0.07
24-表项 8-读/8-写 CAM	1.36	2.0	0.08

单次访问的能耗不足以反应负载运行时刻的特征,因此我们更关心运行时刻的能耗情况. 由于 AVF 和 DWQ 管理机制不同,因此在各个结构上的访问分布是不同的. 为了比较,我们首先给出下面的分析模型以便于理解.

$$E_{\text{DRRF\_total}} = E_{\text{RF}} \cdot N_{\text{RFread}} + 2 \cdot E_{\text{RF}} \cdot N_{\text{RFwrite}} \quad (1)$$

$$E_{\text{DWQ\_total}} = E_{\text{RF}} \cdot (N_{\text{RFread}} + N_{\text{RFwrite}}) + E_{\text{DWQ}} \cdot (N_{\text{DWQread}} + N_{\text{DWQwrite}}) \quad (2)$$

$$E_{\text{AVF\_total}} = E_{\text{RF}} \cdot (N_{\text{RFread}} + N_{\text{RFwrite}}) + E_{\text{AVF}} \cdot (N_{\text{AVFread}} + N_{\text{AVFwrite}}) \quad (3)$$

式(1)~(3)分别表示三种结构下寄存器堆结构总的能耗. 其中  $E_{\text{RF}}$ ,  $E_{\text{DWQ}}$  和  $E_{\text{AVF}}$  分别是访问 8-读/8-写的 RF 体、DWQ 和 AVF 一次的能耗(读或写),  $N$  表示相应结构动态运行时刻读和写的次数. 在运行负载指令数目相同的情况下,忽略错误推测路径上指令数目的差异(通常较小),则动态读请求和写请求的数目是相同的. 首先,相同大小的 DWQ 和 AVF 单次访问的能耗相同,即  $E_{\text{DWQ}} = E_{\text{AVF}}$ , 而  $E_{\text{RF}} : E_{\text{AVF}}$  大致是 4 : 1, 因此尽量避免读 RF 而从辅助结构获得值,将有利于降低能耗,特别是位线上的能耗. 动态执行过程中任意一条指令中的读请求,要么由 RF 完成,要么由辅助性结构完成. 其次,减少辅助结构的写次数也有利于能耗降低,由于 DWQ 要求全部产生的值都写入 DWQ,而 AVF 采用选择性写策略,因此后者在写机制上更节能. 图 9 和图 10 分别统计运行时刻辅助结构上读写访问占总访问次数的百分比.

由图 9 可以看到,相同大小的 AVF 比 DWQ 结构可获得更多的读请求. 特别的,AVF 结构可获得的读请求百分比随大小增长显著,而 DWQ 则增长较小. 对于单线程负载,24-表项 DWQ 和 AVF 可以获得的读请求分别占 55.8% 和 64.8%. 对于多线程

负载,24-表项 DWQ 和 AVF 可以获得的读请求分别占 29.2%和 48.3%.AVF 中存放的值都处于活跃周期,在不久的将来一定会被使用.DWQ 机制则期望写入的值在最近被访问,但事实上很多表项被占用而其中的值却不会被访问.在相同表项下 AVF 效率更高,能够获得更多的读请求,从而尽量避免访问 RF.特别是在 SMT 执行情况下,通过 DWQ 获得的读请求明显少于单线程的情况,这是由于多个线程中的指令同时竞争发射带宽引起的.特定线程产生的值,由于发射带宽被其他线程占用,该线程内后续可发射的指令也会被推迟,因此降低了访问 DWQ 的机率.

由图 10 可以看到,由于采用选择性复制机制,对于动态产生的值,大约只有一半会被写入 AVF.

24-表项 AVF 写次数平均约比 16-表项 AVF 高 10%.利用 Watch<sup>[17]</sup> 在 CC3 模式下对各种配置下寄存器堆的运行时刻能耗进行估计,该模式假设在电路级已经实现了很好的时钟门控,因此本身具有较高的能耗效率.图 11 给出所有负载运行下寄存器堆相对于基准结构的能耗节省.AVF-24 结构相对能耗最低,在单线程负载下平均节省 45%,比 DWQ 结构多节省 5%,而多线程负载下平均节省 39%,比 DWQ 结构多节省 10%.AVF-24 比 AVF-16 更节省能耗,主要是由于 AVF-24 获得读请求的百分比更大, $E_{RF}:E_{AVF}=4:1$ ,在读访问上获得的能耗节省较多.而实现更大的 AVF 访问时间将超过 RF,影响时钟频率.

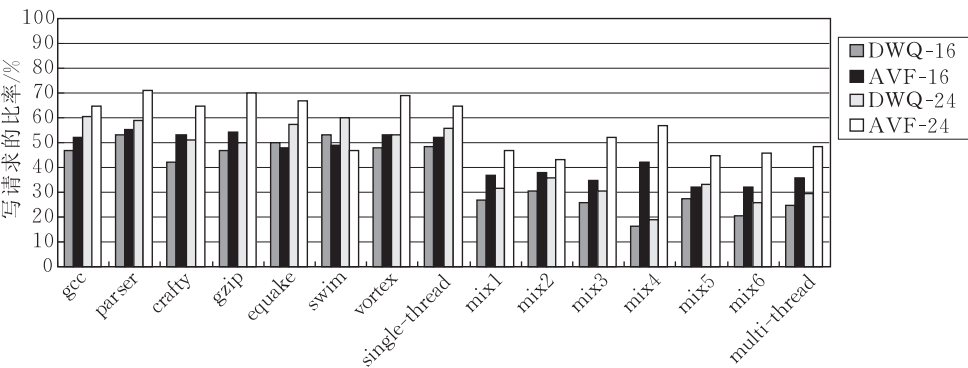


图 9 辅助结构上读请求的百分比

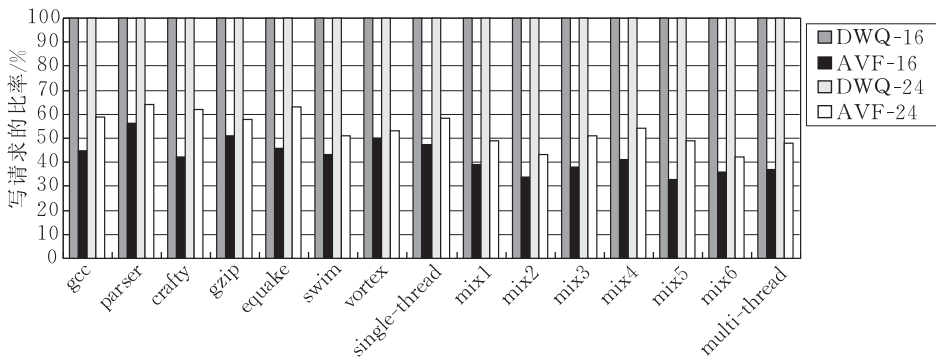


图 10 辅助结构上写请求的百分比

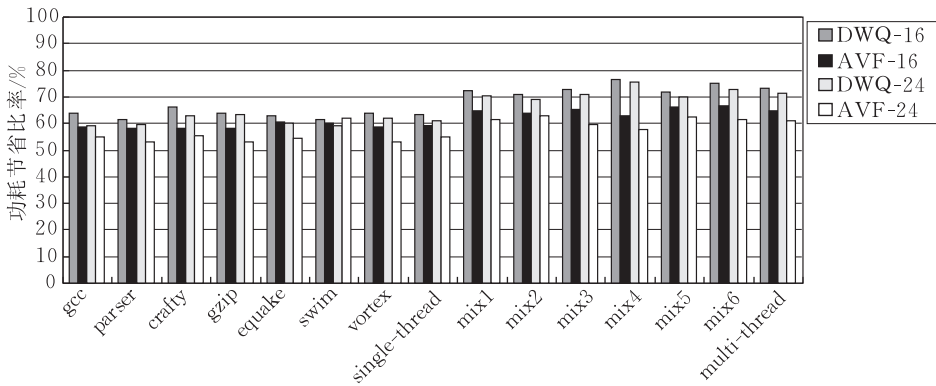


图 11 运行时刻寄存器堆结构节省的总的能耗



AVF 技术为每个物理寄存器增加了标志位和计数器. 在 Wattch 中, 5-bit 的计数器每次访问的能耗为 0.18pJ, 模拟采样显示每个寄存器的计数器平均 20 多个周期访问一次, 因此所有物理寄存器的标志位向量平均每周期的能耗约 4.6pJ, 比 RF 单次访问的能耗低 3 个数量级, 不会成为瓶颈. 发射队列中增加的 A 位设置逻辑由一些与门和或门实现, 也不会构成主要的能耗开销.

#### 5.4 与多体多端口寄存器堆的比较

区别于采用辅助性结构降低单一寄存器堆的端口数, 多体多端口寄存器堆是另外一种重要的降低寄存器堆访问时间和功耗的技术<sup>[8]</sup>. 与单一寄存器堆相比, 多体多端口寄存器堆提供数目相同的全局读写端口, 但内部划分为多个交叉的寄存器体, 每个寄存器体具有数目较少的局部读写端口, 但需要额外的交叉多选电路连接局部端口的位线到全局端口的位线. 当全局端口访问数目超过寄存器体的端口数限制时导致端口冲突. 多体多端口寄存器堆的主要问题在于额外交叉多选电路的开销、端口仲裁逻辑以及端口冲突带来的控制逻辑的复杂性和潜在的性能损失.

从控制逻辑的复杂性角度来看, 多体多端口寄存器堆需要在发射级与寄存器读级增加一个仲裁流水级, 利用寄存器地址进行体和端口的选取, 并同时检查各个寄存器体上的读写冲突. 如果在仲裁级发现读写端口冲突, 则需要暂停存在端口冲突的指令. 在检测到端口冲突的后一个周期, 需要利用恢复机制将引起端口冲突的指令重新插入发射队列, 并将这些指令的目标寄存器标签广播到发射队列以清除后续等待指令的 Ready 位, 从而阻止相关指令发射. 引入这种新的推测式发射的机制对电路时序的要求非常高, 而且大大增加了控制逻辑的复杂度. 而利用辅助性结构 (AVF 或 DWQ 结构) 降低主寄存器端口数目不需要复杂的控制逻辑, 仅在唤醒时刻产生一个标志信号并在选择阶段进行读端口冲突检测, 不会显著影响发射逻辑.

从功耗性能角度来看, 文献[8]基于 4 发射处理器, 相对于单一结构的寄存器堆, 多体多端口寄存器堆的访问时间降低 20%, 峰值功耗降低 40%, IPC 损失为 5%. 从实验结果比较, AVF 或 DWQ 结构在寄存器堆访问时间上慢于多体多端口寄存器堆, 峰值功耗的降低比率相当, 但 IPC 性能损失较小, 并且控制逻辑复杂度低. 文献[8]未在 SMT 环境下进行性能评估, 因而难以判定多体多端口寄存器堆的

端口冲突对多线程负载的潜在性能影响.

体系结构设计者可以根据特定的设计目标选择合适的寄存器堆结构和技术. 从追求频率的目标而言, 多体多端口寄存器堆是一个很好的选择, 但其代价是控制和恢复逻辑的复杂度较高. 利用辅助性结构的优点是其利用了负载程序的行为特征, 设计简单, 并且能达到很好的功耗降低效果. 采用一个简单的技术降低功耗, 通常容易与其他微体系结构技术相结合.

## 6 结 论

降低端口数目对降低宽发射处理器寄存器堆的能耗具有直接和显著的影响. 根据物理寄存器活跃周期较短的特点, 提出了选择性复制活跃值的管理技术, 利用非常小的结构 AVF 分担 RF 读访问, 从而降低 RF 读端口数目. 相同大小的结构下, AVF 比 DWQ 结构的利用率更高, 可以获得更多的读请求, 从而尽量减少 RF 读, 同时写过滤机制也更进一步降低了写开销. 实验表明, 24-表项 AVF 可以降低 RF 一半读口, IPC 损失仅为 3%. 对于单线程负载和多线程负载, 寄存器堆上的能耗分别降低 45% 和 39%, 而不影响时钟频率. 另外, 其面积降低还将有利于缩短全局互连线.

## 参 考 文 献

- [1] Preston R P et al. Design of an 8-wide superscalar RISC microprocessor with simultaneous multithreading//Proceedings of the IEEE International Solid-State Circuits Conference. San Francisco, 2002: 266-500
- [2] Zhao Yu-Lai, Li Xian-Feng, Tong Dong, Cheng Xu. An energy-efficient instruction scheduler design with two-level shelving and adaptive banking. Journal of Computer Science and Technology, 2007, 22(1): 15-24
- [3] Monreal T, Gonzlez A, Valero M, Gonzlez J, Vinals V. Delaying physical register allocation through virtual-physical registers//Proceedings of the 32nd International Symposium on Microarchitecture. Haifa, 1999: 186-192
- [4] Martinez J F, Jose Renau, Huang M C, Milos Prvulovic, Josep Torrellas. Cherry: Checkpointed early resource recycling in out-of-order microprocessors//Proceedings of the 35th International Symposium on Microarchitecture. Istanbul, Turkey, 2002: 3-14
- [5] Oguz Ergin, Deniz Balkan, Dmitry Ponomarev, Kanad Ghose. Increasing processor performance through early register release//Proceedings of the International Conference on Computer Design. San Jose, 2004: 480-487

- [6] Oguz Ergin, Deniz Balkan, Dmitry Ponomarev, Kanad Ghose. Register packing: Exploiting narrow-width operands for reducing register file pressure//Proceedings of the 37th International Symposium on Microarchitecture. Oregon, Portland, 2004; 304-315
- [7] Lipasti M H, Mestan B R, Gunadi Erika. Physical register inlining//Proceedings of the 31st International Symposium on Computer Architecture. München, Germany, 2004; 325-335
- [8] Tseng J H, Asanovic K. Banked multiported register files for high-frequency superscalar microprocessors//Proceedings of the 30th International Symposium on Computer Architecture. San Diego, 2003; 62-71
- [9] Balasubramonian R, Dwarkadas S, Albonesi D H. Reducing the complexity of the register file in dynamic superscalar processors//Proceedings of the 34th International Symposium on Microarchitecture. Istanbul, Turkey, 2001; 237- 248
- [10] Park I, Powell M D, Vijaykumar T N. Reducing register ports for higher speed and lower energy//Proceedings of the 35th International Symposium on Microarchitecture. Istanbul, Turkey, 2002; 171-182
- [11] Kim Nam Sung, Mudge Trevor. Reducing register ports using delayed write-back queues and operand pre-fetch//Proceedings of the 17th International Conference on Super-computing. San Francisco, 2003; 172-182
- [12] Sharkey J. M-sim: A flexible, multi-threaded simulation environment. Department of Computer Science, SUNY Binghamton; Technical Report CS-TR-05-DP1, 2005
- [13] Burger D, Austin T. The simplescalar tool set: Version 2. 0. Department of Computer Science, University of Wisconsin-Madison; Technical Report #1342, 1997
- [14] Sherwood T, Perelman Erez, Hamerly Greg, Calder Brad. Automatically characterizing large scale program behavior//Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems. San Diego, 2002; 45-57
- [15] Brian Fields, Rastislav Bodik, Hill M D. Slack; Maximizing performance under technical constraints//Proceedings of the 29th International Symposium on Computer Architecture, Alaska, 2002; 47-58
- [16] Shivakumar P, Jouppi N P. CACTI 3. 0: An integrated cache timing, power, and area model. Western Research Laboratory; Technical Report 2001/2, 2001
- [17] Brooks D, Tiwari V, Martonosi M. Watch: A framework for architectural-level power analysis and optimization//Proceedings of the 27th International Symposium on Computer Architecture Vancouver. Canada, 2000; 83-94



**ZHAO Yu-Lai**, born in 1980, Ph.D. candidate. His research interests include computer architecture, performance evaluation, HW/SW co-design, and System-on-Chip.

**LI Xian-Feng**, born in 1973, post-doctoral. His research interests include computer architecture, real-time systems, and System-on-Chip.

**TONG Dong**, born in 1971, assistant professor. His research interests include computer architecture, storage sys-

tem, interconnection network, and System-on-Chip.

**SUN Han-Xin**, born in 1980, Ph. D. candidate. His research interests include computer architecture, HW/SW co-design and microprocessor validation.

**CHEN Jie**, born in 1979, Ph. D. candidate. His research interests include computer architecture, power estimation and low power design.

**CHENG Xu**, born in 1967, Ph. D. , professor, Ph. D. supervisor. His main research fields include high performance microprocessor, System-on-Chip, embedded system, instruction level parallelism, HW/SW codesign and compiler optimization.

## Background

Power and energy consumption has become the major constraint as the instruction window and issue width scale to achieve more ILP and TLP. Physical register file (RF) is power-critical due to its large size and number of ports. It is prevalent to replicate the contents of RF in dual copies to reduce the number of ports in recent microprocessor such as Alpha21464. However, it is still not energy-efficient since each produced value must be written twice. Although previous studies have been focused on reducing RF ports or size, they paid less attention to SMT processors where multiple contexts share and compete for datapath resources. The authors aim at reducing the number of RF ports and energy on

wide-issue superscalar microprocessors which support SMT execution. To achieve this goal, it is important to characterize the behavior of both single- and multi-threaded workloads. Based on the observation that the active cycles of registers are quite short, the authors propose to manage the active register values in a small active value file (AVF) if can keep tracking of its usage. Compared with previous techniques utilizing auxiliary structure for energy reduction, AVF can circumvent more read requests on the main RF, reduce the number of writes on the auxiliary structure and therefore save more energy. Meanwhile, it achieves the goal with small IPC loss, no frequency impact and little additional design complexity.