基于行冲突预测的内存控制器 QoS 管理机制

黄 侃,佟 冬,程 旭
(北京大学微处理器研究开发中心,北京 100871)

摘 要: 传统内存控制器 QoS 管理机制对处理器访存延迟控制能力有限.针对该问题,本文提出 PRCC 机制.该机制基于对后续处理器访存的预测,分析发出设备访存对处理器访存延迟的影响,阻止发出会导致处理器访存从行命中变为行冲突的设备访存,从而减少设备访存对处理器访存延迟的影响.面向北大众志 SK SoC 的评测结果表明:相比于传统方法,应用 PRCC 机制使设备访存对程序执行时间的影响从 24%减少到 12%,并且通过参数调节,能够在处理器与设备间取得更优的性能折衷.

关键词: 内存控制器; 访存调度; 系统级芯片; 服务质量
中图分类号: TP302 **文献标识码:** A **文章编号:** 0372-2112 (2011) 02-0358-06

A QoS Management Mechanism of Memory Controller Based on Row Conflict Prediction

HUANG Kan, TONG Dong, CHENG Xu
(*Microprocessor Research and Development Center of Peking University, Beijing* 100871, *China*)

Abstract: Traditional QoS management mechanism of memory controller cannot effectively control the latency of processor accesses. To solve this problem, this paper proposes the PRCC mechanism. Based on the prediction of the future processor accesses, PRCC analyzes the impact of device accesses on the latency of processor accesses, and prevents issuing the accesses that will result in that processor accesses change from row hits to row conflicts, so PRCC can reduce the impact of device accesses on the latency of processor accesses. The evaluation results for PKUnity-3(SK) SoC show that after applying PRCC service, the impact of device accesses on program execution time is reduced from 24% to 8%. By parameter adjustment, PRCC achieves better performance trade-offs between processor and devices than traditional methods.

Key words: DRAM; memory access scheduling; system-on-chip; quality of service

1 引言

随着人们对嵌入式产品功能与性能需求的提升,系统芯片(System-on-Chip, SoC)所集成模块的数量显著增加[1].由于不同类型的模块共享片外存储资源,不仅造成整体访存压力增大,并且使访存需求更加多样化[2].例如,处理器要求访存延迟尽可能小,而实时性设备则要求保障一定大小的访存带宽.为了满足各模块不同的访存需求,SoC 内部的内存控制器设计必须能有效管理各模块获得的访存服务质量(Quality of Service, QoS)[3].

内存控制器的 QoS 管理目标包括以下几点:(1)保障实时性设备所需的实时带宽;(2)尽量减少处理器访存延迟;(3)兼顾非实时性设备的访存带宽.传统的 QoS 管理机制能够有效控制设备所获得的访存带宽,但对于处理器访存延迟的控制能力有限,难以满足一些对处理器性能敏感的应用需求.根据本文的评测,在传统的 QoS 管理机制下,即使优先执行处理器访存也不能有效控制处理器访存延迟,甚至还可能导致访存延迟恶化.

本文提出基于预测的行冲突控制(Prediction-based Row Conflict Control, PRCC)访存服务机制,以有效控制非实时性设备访存对处理器访存延迟的影响,提升整体访存 QoS.PRCC 服务机制基于处理器访存历史记录,预测后续还未进入内存控制器的处理器访存对 DRAM 行缓冲器的使用情况,再根据预测结果,分析发出非实时性设备访存是否会导致后续处理器访存从行命中变为行冲突,并阻止满足条件的非实时性设备访存交易发出,从而减少非实时性设备访存对处理器访存延迟的影响.相比于传统方法,PRCC 具有以下几项优点:(1)在不同大小的设备访存压力下,均能有效控制处理器访存延迟;(2)通过参数调节,可在处理器与设备之间进行不同程度性能折衷,并具有更高的 DRAM 访问效率;(3)可与其他访存 QoS 管理机制配合,更高效的满足不同的访存需求.

收稿日期:2009-12-09;修回日期:2010-04-09

2　相关工作

现代 SoC 产品普遍要求内存控制器提供 QoS 管理能力:MSoC[4]基于令牌分发机制,管理各设备对内存资源的使用;FlexFilm[5]使用流控制方法分配内存资源;AMD Griffin 北桥芯片[6]通过动态调节显示模块访存的优先级,在保障显示模块实时性访存带宽的同时,优化处理器访存延迟.此外,ARM[7]和 Sonics[8]的商业内存控制器 IP 都具有管理各设备访存 QoS 的能力.文献[3]通过对访存开销估计,动态控制处理器访存优先级,确保实时性设备的带宽需求总能被满足.但上述各 QoS 管理机制都着重于实时性设备的带宽需求,对于非实时性设备带宽与处理器访存延迟的控制能力有限.文献[9]提出了三层的内存控制器设计,其中第二层的 QoS 管理在原有方法基础上,增加了对处理器访存延迟的控制.当处理器访存到达内存控制器时,可以打断正在执行的设备访存,并在一段时间内禁止设备交易使用 DRAM 数据通路,但该方法也未考虑行冲突,处理器访存延迟仍会因行命中率下降而显著增加.此外,多核处理器也要求内存控制器提供 QoS 管理能力[10],但其访存需求与 SoC 有较大不同,其需要在各个线程间的保持公平.

处理器与设备不仅在 DRAM 端存在冲突,在共享总线上也存在冲突.腾跃-1 SoC[11]通过增加突发传输的长度,减少设备访存对处理器访存性能的影响.文献[12]提出 MCS-DMA 机制,通过对设备 DMA 访存进行预取,提升设备访存的总线带宽利用率,减少 DMA 访存与处理器访存在总线上的冲突.这些研究可以与本文工作形成互补,有效满足 SoC 整体的访存需求.

3　研究动机

传统 QoS 管理机制为处理器访存提供优先服务(Priority Service),而为非实时性访存提供尽力服务(Best Effort Service)[3,8].本文基于后文 5.1 节中介绍的模拟环境,评测了在不同大小的设备访存压力下该机制的 QoS 管理能力.其中处理器负载使用 SPEC2000 中访存密集的 mcf 基准程序,设备访存的读写类型、起始地址、传输长度均在合理范围内随机生成.

根据图 1(*a*)的评测结果,相比于无 QoS 管理时,传统的"优先服务 + 尽力服务"QoS 管理机制仅在设备请求带宽较大时,才能优化处理器访存性能,但程序执行时间相比于处理器单独运行时也增加了约 60%,而在设备请求带宽较低时,甚至使程序执行时间恶化.说明传统 QoS 管理机制不能有效控制处理器访存延迟.

传统 QoS 管理机制对处理器访存延迟控制不力的原因是其未考虑 DRAM 访问的特性.现代 DRAM 基于快速页模式,每个 Bank 内设置一个行缓冲器(row buffer),用于暂存最近一次访问所在行的全部数据,以提升访存性能.访存交易行命中(row hit)时,可以直接进行读写操作;而访存交易行冲突(row conflict)时,必须先将行缓冲器里的数据回写,然后再将所访问行的数据载入行缓冲器,最后才能进行读写操作.因此,行冲突会显著增加访存延迟.

如图 1(*b*)所示,随着非实时性访存压力的不断增大,处理器读交易行命中率持续下降,而且应用传统 QoS 管理机制后下降得更快.在整体访存带宽较小时,设备与处理器访存在 DRAM 命令和数据通道上的冲突较小,优先服务的作用有限,而行冲突次数比无 QoS 时更多,故处理器访存性能出现恶化;在整体访存带宽较大时,虽然优先服务发挥了较大作用,使处理器访存性能没有进一步恶化,但由于行命中率的显著下降,处理器访存性能仍有较大损失.

用本文提出 PRCC 服务机制替代传统的尽力服务后,能够有效保护处理器读交易的行命中率(见图 1(*b*)),从而使程序执行时间增幅明显减缓,最大增幅不足 20%(见图 1(*a*)).下面详细介绍 PRCC 服务机制.

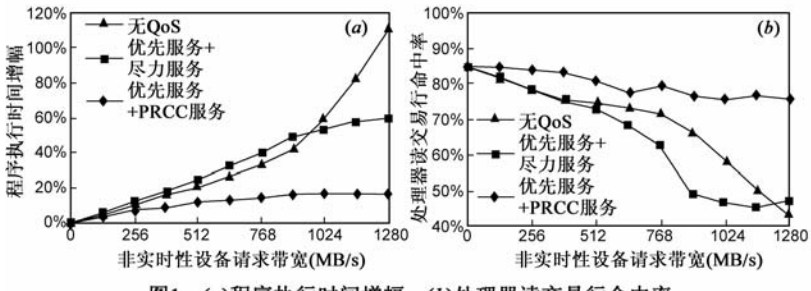

图1　(*a*)程序执行时间增幅;(*b*)处理器读交易行命中率随非实时性访存带宽变化趋势

4　PRCC 服务机制

4.1　基本原理

当内存控制器命令队列只有非实时性设备访存时,传统的尽力服务机制会立刻将其发出.但发出的非实时性设备访存可能会改变行缓冲器的状态,从而导致处理器访存从行命中变为行冲突,显著增加处理器访存延迟.和传统方法不同,即使在内存控制器中只有非实时性设备访存时,PRCC 服务机制首先分析发出各访存对后续处理器访存延迟的影响,如果发出的非实时性设备访存不会导致处理器访存从行命中变为行冲突,则将允许其发出,否则将阻止其发出直至这一影响消除.

为了实现 PRCC 服务机制,基于传统的内存控制器设计,在总线接口和调度器之

间增加了一个预测器,同时修改了调度器,如图2所示.预测器基于处理器读交易的历史记录,预测处理器发往各个 Bank 的下一次读交易的行地址是否与之前的处理器读交易行地址相同,并将预测结果传给调度器.调度器再基于预测结果,控制非实时性设备访存的发出,实现访存 QoS 管理.

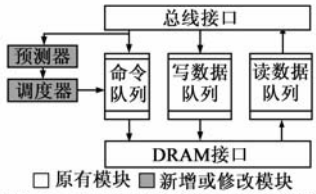

图2 实现PRCC服务内存控制器所需的修改

4.2 预测器

预测器的功能是根据处理器读交易历史记录,预测后续处理器读交易是否访问和上一次处理器读交易相同的行.由于各 Bank 具有独立的行缓冲器,故为每个 Bank 设置单独的预测器,分别预测后续处理器访问对各 Bank 内行缓冲器的使用情况.预测器具体结构如图3(*a*)所示,其中 *B* 为 DRAM 系统中的 Bank 总数,等于 Rank 数乘以各 Rank 内的 Bank 数.

当总线接口收到来自处理器的读交易时,交易译码器根据交易的 Bank 地址,译码产生 *Sel* 信号选中相应 Bank 预测器,并传递交易的行地址 *Row* 给 Bank 预测器.Bank 预测器使用指定的预测算法,预测未来一次访问该 Bank 的处理器读交易是否与当前交易访问相同的行,并将预测结果 *PSR*(Predict the Same Row)发送给调度器.

考虑到处理器访存的规律性和硬件实现的复杂度,本文使用如下 *N* 阶预测算法:对于某一个 Bank,如果最近连续 *N* 次访问该 Bank 的处理器读交易均访问相同的行,则预测下一次访问该 Bank 的处理器读交易也访问该行.当 $N=0$ 时,一直预测下一次访问该 Bank 的处理器读交易与当前交易访问相同行.

如图3(*b*)所示,为实现该算法,Bank 预测器内部设置了一个 *LastRow* 寄存器和一个饱和计数器 *PreCnt*.当 Bank 预测器被选中时,交易的行地址与 *LastRow* 进行比较,如果相等且当前 *PreCnt* 小于 *N*,则 *PreCnt* 加一;如果相等且当前 *PreCnt* 等于 *N*,则 *PreCnt* 保持为 *N*;如果

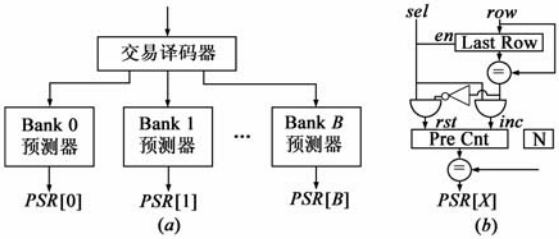

图3 (*a*) 预测器整体结构 (*b*) Dank预测器内部结构

不相等,则 *PreCnt* 归零.当 *PreCnt* 等于 *N* 时,预测下一次访问该 Bank 的处理器读交易访问当前行,*PSR* = 1.此外,在 Bank 预测器被选中时,*LastRow* 寄存器的值也同步更新.

4.3 调度器

和传统方法一样,调度器对处理器访存提供优先服务,对实时性设备访存提供分配带宽服务,故调度器优先发出实时性设备访存和处理器访存.但对于非实时性设备访存,用 PRCC 服务代替尽力服务.当命令队列中仅有非实时性设备访存时,对各交易发出后对后续处理器访存延迟的影响进行分析,决定是否可以发出该交易.对于任意一个非实时性设备访存交易(其访问 Bank *X*),当下列三个条件同时满足时,PRCC 服务不允许该交易发出预充电(Precharge)命令改变 Bank *X* 行缓冲器的状态,直至其中某一条件不再满足:

条件 *A*:Bank *X* 对应预测器输出结果 *PSR* 为真.

条件 *B*:Bank *X* 的行缓冲器当前被处理器读交易占用.

条件 *C*:访存空闲时间尚未超过预设的阈值 t_{TH}.

当条件 *A* 与条件 *B* 同时满足时,下一次访问 Bank *X* 的处理器读交易极可能为行命中,故阻止非实时性设备交易向 Bank *X* 发出预充电命令,以避免产生额外的行冲突.此外,增加条件 *C*,一方面避免内存长时间空闲,另一方面也可以兼顾非实时性设备的带宽.其中 t_{TH} 由软件预设,取值范围为 0 到 *T*,*T* 为对应计时器的最大值.

为了上述实现调度策略,需要对调度器进行一定修改,如图4所示.其中灰色模块为原有内存控制器部件.设原有控制器命令队列共有 *Q* 个记录项,为了实现 QoS 管理,每个记录项必包含 Bank 地址(BA)、行地址(Row)、列地址(Col)、交易设备号(Dev)和交易优先级(Pri)等信息.现在每一记录项上增设一比特 PF(PRCC Flag),以标志上述三条件是否同时满足.对于处理器读交易和实时性设备访存交易,调度逻辑忽略 PF 位;对于写交易和非实时性设备交易,仅当 PF 为零时,调度逻辑才允许发射该交易.

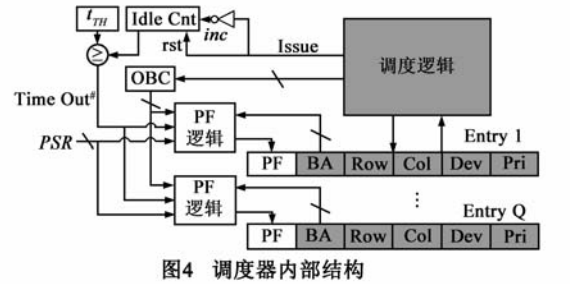

图4 调度器内部结构

每一个 Entry 均有一个单独的 PF 更新逻辑,其内部结构如图5所示,其根据四项输入实现上述三个条件.

条件 *A* 用对应 Entry 的 *BA* 域选择预测器 *PSR* 的对应位实现.为实现条件 *B*,增设一个 *B* 位宽的 OBC(Owned By CPU)寄存器,以记录各 Bank 是否被处理器占用.OBC 由调度逻辑更新,当调度逻辑因发出处理器读交易而激活 Bank *X* 时,OBC[*X* – 1]位设为 1;当调度逻辑 Precharge Bank *X* 时,OBC[*X* – 1]位设为 0.通过对应 Entry 的 *BA* 域选择 OBC 对应位以实现条件 *B*;为实现条件 *C*,增设一个空闲计数器 *IdleCnt*,当调度逻辑发出访存命令时,*IdleCnt* 归零,调度逻辑不发出访存命令时,*IdleCnt* 每周期自增.比较 *IdelCnt* 与 t_{TH} 寄存器的值,产生 *TimeOut* # 信号,表示当前访存空闲时间是否超过阈值,以实现条件 *C*.

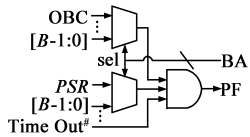

图5　PF逻辑内部结构

5　实验评测

5.1　实验环境与工作负载

本文基于两款开源的周期精确模拟器:M5[14] 2.06beta 和 DRAMsim[15]1.2,对北大众志 PKUnity-3(SK) SoC[16](简称 SK SoC)进行建模,所建模的硬件平台如图 6 所示.为了排除通信结构冲突的影响,各访存设备均与内存控制器直连.处理器和各设备在 M5 中建模,内存控制器和内存条在 DRAMsim 中建模.两个模拟器通过函数调用实现交易传输和时钟同步.

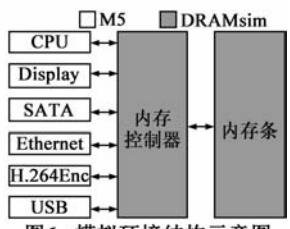

图6　模拟环境结构示意图

处理器配置与 SK SoC 完全一致,内存条基于工业界主流产品参数[17],如表 1 所示.处理器的负载选择了五个访存密集型的 SPEC2000 基准程序:mcf、art、gcc、galgel 和 equake,其中 mcf、art 和 gcc 的行命中率高于 50%,galgel 和 equake 的行命中率低于 50%.各设备的访存特性与 SK SoC 保持一致,如表 2 所示.

表 1　模拟硬件平台主要参数

项目名称	特性
处理器核	600MHz,单发射,顺序执行
L1 I-cache	16KB,4 路组相联,600MHz,写返回,LRU 替换策略,块大小 64B
L1 D-cache	16KB,4 路组相联,600MHz,写返回,LRU 替换策略,块大小 64B
DRAM 系统	DDR2-533,1GB,64-bit 位宽,1 Channel,1 Rank,8 Bank,16384 Row
DRAM 时序	*CL* = 4, *RCD* = 4, *RP* = 4
地址映射方式	物理地址从低到高依次为列地址、Bank 地址、行地址
内存控制器	Open Page 策略,命令队列深度 *Q* = 32, *T* = 2048

表 2　各设备访存特性

设备名称	需求带宽	访存类型	特性
Display	200 MB/s	实时性	仅读交易,顺序地址访存
SATA	300 MB/s	非实时性	包含读写交易,地址具有任意性
Ethernet	125 MB/s	非实时性	包含读写交易,地址具有任意性
USB	60 MB/s	非实时性	包含读写交易,地址具有任意性
H.264Enc	150 MB/s	非实时性	包含读写交易,地址具有任意性

5.2　预测算法评测

本文分别评测了零阶、一阶、二阶和三阶预测算法的预测正确率.不同预测算法的预测正确率如图 7 所示.零阶预测算法虽然实现简单,但对于命中率较低的程序(如 equake 和 galgel)预测正确率较低.一阶预测算法平均正确率最高,达到 88%,且各程序的预测正确率均在 75%以上.二阶和三阶预测算法虽然有个别程序正确率较高,但平均正确率不如一阶预测算法.因此,后续的评测都基于一阶预测算法.

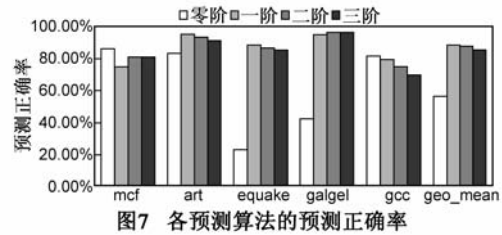

图7　各预测算法的预测正确率

5.3　性能评测

本文面向 SK SoC 的访存需求,对三种不同的 QoS 管理策略进行对比.如表 3 所示,策略 1 源自文献[4,5],策略 2 源自文献[3,8],策略 3 为本文所提出的 QoS 策略,其中 PRCC 服务 t_{TH} 参数取最大值 *T* = 2048.

表 3　QoS 策略组合

QoS 策略	内存服务类型		
	Display	处理器	其他设备
策略 1	分配带宽服务	尽力服务	尽力服务
策略 2	分配带宽服务	优先服务	尽力服务
策略 3	分配带宽服务	优先服务	PRCC 服务

各种 QoS 策略下,Display 的实时性带宽均得到保障,程序执行时间均会因为其他设备的访存而增加.相比于无设备访存时,设备满负荷工作时程序执行时间的增幅如图 8(*a*)所示,在策略 1 和策略 2 下,程序执行时间分别平均增加 27%和 24%,而策略 3 下,程序执行时间平均仅增加 12%.图 8(*b*)对比了三种 QoS 策略下非实时性设备带宽的变化:在策略 1 和策略 2 下,设备带宽变化很小,而在策略 3 下,设备带宽能够随处理器访存压力而变化,处理器访存压力较大时(如 art 和 mcf),设备带宽较低,处理器访存压力较小时,设备带宽较高(如 galgel).评测结果说明:如果为非实时性设备访存提供尽力服务,优先服务难以使处理器优先于非实时性设备使用内存资源,用 PRCC 服务替代尽力服务

后,虽然牺牲一些非实时性设备访存带宽,但有效减少了处理器访存延迟,使调度结果更加符合设计预期.

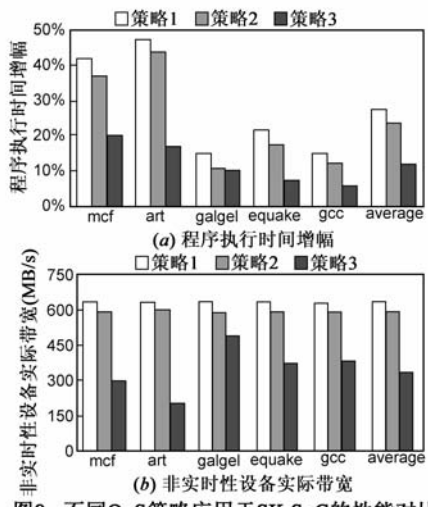

图8 不同QoS策略应用于SK SoC的性能对比

5.4 性能权衡

为了满足不同应用场景的需求,内存控制器需要能够在处理器与设备间进行细粒度的性能折衷.传统方法为设备访存提供限制带宽服务[4],当设备占用带宽超过预设的带宽上限时,内存控制器将在一段时间内不响应其访存请求,从而使处理器获得更好的访存性能.通过调节设备的带宽上限值,可以进行细粒度的性能折衷.

PRCC 服务通过调节 t_{TH} 参数大小,也可以在处理器访存延迟与非实时性设备访存带宽间进行细粒度性能折衷.t_{TH} 值越大,处理器访存延迟越小,非实时性设备带宽也越小;t_{TH} 值越小,处理器访存延迟越大,非实时性设备带宽也越大;当 $t_{TH}=0$ 时,PRCC 服务等同于尽力服务;当 $t_{TH}=T$ 时,条件 C 总是满足.

通过调节 t_{TH},可以拟合出设备与处理器访存性能的折衷曲线,如图 9 所示.纵轴为程序执行时间相对于无设备访存时的增幅,表征处理器访存性能的损失,横轴为设备带宽损失率,其表征设备访存性能的损失,表达式为:

$$\text{设备带宽损失率}=1-\frac{\text{设备实际获得带宽}}{\text{设备请求带宽}}\times 100\%$$

其中,设备请求带宽为设备访存总是被立即响应时可获得的最大带宽.图 9 中,PRCC 服务的权衡曲线更加接近于坐标轴,说明其整体性能损失更小.如果为设备提供相同大小的访存带宽,PRCC 服务能使程序执行时间更短.

虽然 PRCC 服务不能将设备带宽限制得非常小,但将其与限制带宽服务混合使用可以解决这一局限性.设备访存首先使用 PRCC 服务进行控制,如果实际带宽仍然超过带宽上限,再使用限制带宽服务控制设备占用的带宽大小.如图 9 中虚线部分所示,混合服务的性能也优于传统的限制带宽服务.综上,PRCC 服务能够更加高效的在处理器与设备间进行细粒度性能折衷.

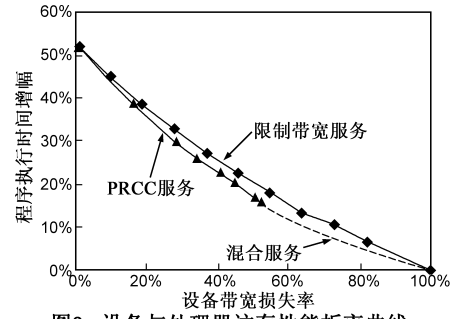

图9 设备与处理器访存性能折衷曲线

5.5 硬件实现分析

实现 PRCC 机制在原有内存控制器基础上,增加了预测器,并对调度器进行了修改.预测器共包含 B 个 Bank 预测器,每个 Bank 预测器内部包含一个 $\log_2 R$ 位的寄存器 *LastRow*、一个 $\log_2(N+1)$ 位的饱和计数器 *PreCnt*、一个 $\log_2 R$ 位比较器、一个 $\log_2(N+1)$ 位比较器和少量逻辑门.调度器包含一个 B 位的寄存器 *OBC*、一个 $\log_2 T$ 位的寄存器 t_{TH}、一个 $\log_2 T$ 位的计数器 *IdleCnt*、Q 个 *PF* 位以及 Q 个 PF 逻辑,而每个 PF 逻辑由两个 B 选一的多选器和一个三输入与门组成.代入 5.1 节的具体参数后,实现 PRCC 共需增加不足 200 位的寄存器和不超过 200 个逻辑门的组合电路,说明实现 PRCC 服务机制的硬件开销很小.同时,从 PRCC 的硬件实现机制上看,新增电路均并行于原有控制逻辑,且控制逻辑简单,而对原有调度逻辑的修改仅增加了对 *PF* 位的判断,故本文方法不会恶化内存控制器的时序关键路径.

6 结论

传统 QoS 管理机制未考虑因 QoS 管理而导致的行冲突,因此对于处理器访存延迟的控制能力有限.本文提出为非实时性设备访存提供一种新的 PRCC 服务,替代传统的尽力服务.PRCC 服务预测后续处理器读交易对各 Bank 行缓冲器的使用情况,阻止发出会导致后续处理器访存从行命中变为行冲突的非实时性设备交易.同时,通过超时控制,确保内存带宽不出现过度浪费,实现处理器与设备之间访存性能的折衷.面向北大众志 PKUnity-3(SK) SoC 的评测结果显示,相比于传统方法,应用本文提出的 QoS 管理机制,使内存资源的分配更加符合应用需求,并能够在处理器与设备之间取得更高效的访存性能折衷.

参考文献:

[1] Theo A C M Claasen. An industry perspective on current and future state of the art in system-on-chip (Soc) technology[J].

Proceedings of the IEEE,2006,94(6):1121 – 1137.

[2] Youn-Long Steve Lin. Essential Issues in Soc Design: Designing Complex Systems-on-Chip[M]. Dordrecht, The Netherlands: Springer,2006.73 – 118.

[3] Menghao Su, Xiang Gao, et al. Efficiency-aware Qos dram scheduler[A]. IEEE International Conference on Networking, Architecture, and Storage[C]. Zhang Jia Jie, Hunan, China: IEEE Society,2009.223 – 226.

[4] C Macian, S Dharmapurikar, et al. Beyond performance: Secure and fair memory management for multiple systems on a chip [A]. Proceedings of IEEE International Conference on Field-Programmable Technology[C]. Tokyo, Japan: IEEE Society, 2003.348 – 351.

[5] S Heithecker, R Ernst. Traffic Shaping for an Fpga based sdram controller with complex Qos requirements[A]. Proceedings of 42nd Design Automation Conference[C]. Anaheim, CA, USA: IEEE Society,2005.575 – 578.

[6] J Owen, M Steinman. Northbridge architecture of Amd's griffin microprocessor family[J]. IEEE Micro,2008,28(2):10 – 18.

[7] ARM Inc. Primecell® Ddr2 Dynamic Memory Controller Technical Reference Manual [EB/OL]. http://infocenter.arm.com/help/topic/com.arm.doc.ddi0418a/DDI0418.pdf,2009-03-29.

[8] Sonics Memmax® Dram access scheduler[EB/OL]. http://www.sonicsinc.com/uploads/pdfs/MM_datasheet_013108.pdf,2006.

[9] Lee Kun-Bin, Lin Tzu-Chieh, et al. An efficient quality-aware memory controller for multimedia platform Soc [J]. IEEE Transactions on Circuits and Systems for Video Technology. 2005,15(5):620 – 633.

[10] Onur Mutlu and Thomas Moscibroda. Parallelism-aware batch scheduling: enabling high-performance and fair shared memory controllers[J]. IEEE Micro,2009,29(1):22 – 32.

[11] 王蕾,陆洪毅,王进,戴葵,王志英.一种面向嵌入式应用的片上系统:腾跃-1[J].电子学报,2005,33(11):2036 – 2039.
Wang Lei, Lu Hong-yi, Wang Jin, Dai Kui, Wang Zhi-ying. A high performance embedded SoC: tengyue-1[J]. Acta Electronica Sinica. 2005,33(11):2036 – 2039.(in Chinese)

[12] 黄侃,佟冬,等.MCS-DMA:一种面向 SoC 内 DMA 传输的内存控制器优化设计[J].电子学报,2010,38(3):598 – 604.
Huang Kan, Tong Dong, et al. MCS-DMA: An optimization design of memory controller for DMA transfers in SoC[J]. Acta Electronica Sinica,2010,38(3):598 – 604.(in Chinese)

[13] Shao Jun, B T Davis. A burst scheduling access reordering mechanism[A]. International Symposium on High Performance Computer Architecture[C]. Phoenix, Arizona, USA: IEEE Society,2007.285 – 294.

[14] Nathan L Binkert, Ronald G. Dreslinski, et al. The M5 simulator: modeling networked systems[J]. IEEE Micro, 2006, 26(4):52 – 60.

[15] David Wang, Brinda Ganesh, et al. Dramsim: a memory system simulator[J]. SIGARCH Computer Architecture News 2005,33(4):100 – 107.

[16] 程旭,陆俊林,易江芳,刘姝.面向 UMPC 的北大众志-SK 系统芯片设计[J].计算机学报,2008,31(11):1877 – 1887.
Cheng Xu, Lu Jun-lin, Yi Jiang-fang, Liu Shu. Architecture of PKUnity-SK SoC for UMPC[J]. Chinese Journal of Computers.2008,31(11):1877 – 1887.(in Chinese)

[17] Micron Technology. 1Gb: X4, X8, X16 DDR2 SDRAM[EB/OL]. http://download.micron.com/pdf/datasheets/dram/ddr2/1GbDDR2.pdf,2004.

作者简介:

黄　侃　男,1983 年生于湖北武汉,北京大学计算机系博士研究生.主要研究方向为软硬件协同设计、系统芯片的设计和验证.
E-mail: huangkan@mprc.pku.edu.cn

佟　冬　男,1971 年生于吉林长春,北京大学计算机系副教授.主要研究方向为高性能微处理器、系统芯片、体系结构等.

程　旭　男,1967 年生于新疆乌鲁木齐,北京大学计算机系教授,博士生导师.主要研究方向为高性能微处理器、系统芯片、嵌入式系统、指令级并行、优化编译、软硬件协同设计等.