doi:10.3772/j.issn.1002-0470.2019.05.002

# 多核片上系统主控式内存控制器预取①

李 鹏②[*,**,***] 王 剑[***] 曾 露[****] 王焕东[****]

([*]计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)

([**]中国科学院计算技术研究所 北京 100190)

([***]中国科学院大学 北京 100049)

([****]龙芯中科技术有限公司 北京 100095)

**摘 要** 本文提出了一种多核片上系统(MPSoC)主控式内存控制器预取方法来解决多IP核导致内存控制器端预取资源竞争的问题。该方法综合考虑了不同访存流预取的及时性和访存冲突性,将预取数据及时性差的访存流进行过滤,使之在stream buffer资源紧张的情况下不占用流缓冲空间,同时利用流缓冲地址记录表使得存在冲突的访存流优先使用stream buffer,进一步降低了访存冲突的概率。实验表明,该方法可以提升近20%的最大访存带宽,而对带宽需求小的访存IP核可以降低60%左右的访存延迟。

**关键词** 多核片上系统(MPSoC),及时性,访存冲突,预取,流缓冲

## 0 引 言

预取是减少访存延迟的重要手段。传统的预取一般是由处理器核和不同层次的cache发出的,但是由于预取的准确度问题,也会带来cache污染的问题。为了避免该问题,文献[1]提出了内存控制器端的预取。内存控制器端的预取操作由内存控制器发出,主要针对的是流式访存,利用了流式访存的高效性以很小的访存代价将可能用到的数据预取回来并保存在内存控制器中专门的数据缓冲区中。当后续的cache失效访问在该数据缓冲区命中时,则可以避免访问动态随机存取存储器(dynamic random access memory, DRAM),从而大大缩短了访存延迟。多核片上系统(multi-processor system-on-chip, MPSoC)通常集成多个针对不同应用的IP(intellectual property)核,这些IP核的访存普遍具有较好的流式特性,多个IP核同时访存会给内存控制器端预取资源的分配带来严重的竞争问题,而不合理的预取资源分配会导致访存性能下降,因此对多核片上系统内存控制器端预取优化工作变得非常必要。

针对多核片上系统内存控制器端预取资源竞争问题,本文充分参考和借鉴现有的优化方法,在此基础上提出了一种主控式内存控制器预取优化方法。该方法综合考虑了不同访存流预取的及时性和访存冲突性,使得预取资源的分配更加合理,从而大幅提高了系统的访存性能。

## 1 相关工作

内存控制器端的预取不会造成cache污染问题,而且由于可以获得更精确的内存控制器端的访存队列信息,从而可以更好地控制预取的时机和激进程度,有效避免了对正常访存请求的干扰。很多

① 国家"核高基"科技重大专项课题(2009ZX01028-002-003, 2009ZX01029-001-003, 2010ZX01036-001-002, 2012ZX01029-001-002-002, 2014ZX01020201, 2014ZX01030101),国家自然科学基金(61521092, 61222204, 61432016)和中国科学院重点部署项目(ZDRW-XH-2017-1)资助项目。

② 男,1986年生,博士生;研究方向:计算机系统结构;联系人,E-mail: lipeng-cpu@ict.ac.cn

(收稿日期:2018-07-26)

研究[2]表明:内存控制器端的预取可以达到或者超过处理器核和 cache 预取的性能。

文献[1]最先提出了 stream buffer 的概念,通过在内存控制器中增加一个并行先入先出存储器(first input first output, FIFO)结构来存储预取数据,FIFO 中命中的地址数据可以直接返回给 cache,同时预取下一个 cache 行。由于采用并行 FIFO 来存储预取数据,当来自 cache 的访存请求在 FIFO 中不命中需要替换时,会导致某一项 FIFO 中的数据全部清空,这使得预取的数据并没有得到访问,浪费了访存带宽。

为了提高预取的精确度,文献[3]为 stream buffer 增加了一个历史地址记录表,只有当两个连续访存地址被找到时才对接下来的地址进行预取,从而对预取地址起到了很好的过滤作用。

文献[4]针对媒体应用程序的特点提出了与内存控制电路相结合的 stream buffer 预取。该方式通过编译器识别数据流,并将数据流的地址和长度信息发送给 stream buffer 进行相应的数据预取。

文献[5]提出了一种内存控制器内的自适应流检测预取机制,通过流过滤器来跟踪记录数据流的预取长度和生存周期等,然后更新流长度历史表来指导预取的激进程度,从而减少无用数据的预取,同时利用自适应的调度机制来减少预取带来的机会成本。

文献[6]提出了一种预取感知的内存控制器,该内存控制器的核心思想有 2 点:(1)根据预取的准确度,动态调整预取命令和正常命令之间的优先级;(2)根据预取的准确度,动态舍弃一些无用的预取命令,释放访存资源。通过以上两点来最大化有用预取带来的好处和最小化无用预取带来的负面影响。

文献[7]提出了一种面向多核处理器的智能 stream buffer 预取方案,该方案的核心思想有 3 点:(1)延迟替换的流缓冲管理策略,该策略给予已经预取的数据更多的机会,从而避免流数据间的反复替换导致的访存带宽浪费;(2)动态调整预取的激进程度,该策略通过预取数据的使用频率来动态调整预取长度;(3)通过为不同处理器核设置不同的访存地址历史记录表和最近最少使用(least recently used, LRU)管理队列来降低一些无谓的比较操作,从而降低功耗和维护不同核间访存的公平性。

以上这些方法大都将注意力集中在访存流识别、预取时机、预取数据量以及流替换策略上,这些方案将所有访存流做统一考虑,将每个流都作为 stream buffer 的候选使用对象,而没有考虑一个流是否该使用 stream buffer。换句话说,这些方案没有考虑如何将 stream buffer 分给更需要的访存流从而达到更高的访存性能,没有充分发掘不同访存行为的访存流使用 stream buffer 带来的不同效果,因此难以充分利用 stream buffer 达到更好的访存性能。

由于面积和功耗的限制,内存控制器的预取数据缓冲区不可能做得很大。比如 stream buffer,已有的研究[7,8]表明:stream buffer 的项数一般为 4 ~ 8 项,超过此数值几乎不能获得性能提升,每一项的大小一般为 2 ~ 4 个 cache 行,超过此数值会带来无效预取的风险,浪费访存带宽。

多核片上系统中多个 IP 核会产生多个访存流,这些访存流的数量远大于 stream buffer 的项数,而现有的 stream buffer 设计方案并不会区分这些来自不同 IP 核的访存流,也不会限制这些流对 stream buffer 的使用情况,对 stream buffer 的流替换一般采用简单的 LRU 替换算法。这使得很多访存优化的机会被掩盖,因此如何合理分配 stream buffer 从而进一步提升访存性能成为一个挑战。

本文充分参考和借鉴现有的内存控制器预取优化方法,对多核片上系统中的访存流预取行为进行了深入的分析,提出了一种主动控制内存控制器端预取策略。该策略综合考虑了不同访存流预取的及时性和访存冲突性,有效提高了多核片上系统的访存带宽和服务质量。

## 2 Stream buffer 介绍

### 2.1 Stream buffer 结构

内存控制器预取模块通常集成在整个内存控制器模块当中,一个典型的内存控制器预取器在内存控制器中的位置如图 1 所示。

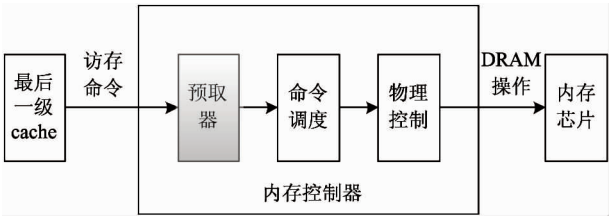


**图 1　预取器在内存控制器中的位置**

内存控制器端预取最常用的方式就是 stream buffer 预取,一个典型的 stream buffer 预取器的结构如图 2 所示。

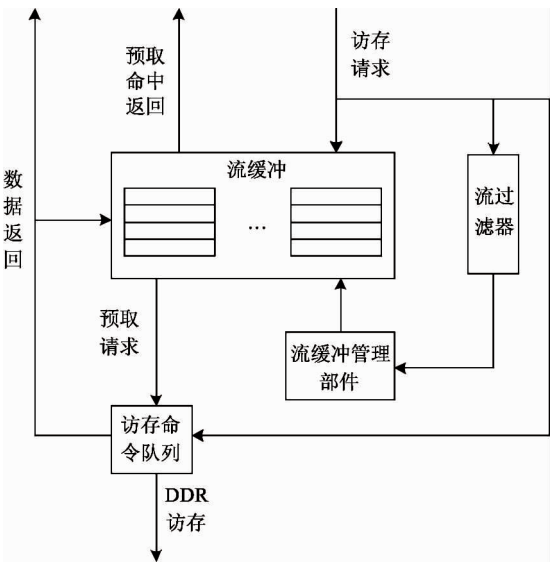


**图 2　Stream buffer 预取器结构**

图中的 stream buffer 预取器主要包含以下几个模块:流缓冲、流过滤器和流缓冲管理部件。流缓冲的作用是存放预取回来的数据,stream buffer 预取器通常包含多个流缓冲,每个流缓冲代表识别出的一个数据流,流缓冲的基本存储单位是一个 cache 行,通常一个流缓冲由多个 cache 行组成。流过滤器里面的核心部件是访存地址历史记录表,负责识别数据流,识别的方式是当收到上面发来的读访存请求时,就对该地址代表数据流的下一个可能的访存地址进行猜测,方法是对该地址进行一定步长的增减(以 cache 行为单位,对内存的访问通常以 cache 行为单位进行),然后保存到访存地址历史记录表中,当后面的访存地址与历史记录表中的某一项相同时,一个新的访存流被识别。流缓冲管理部件的作用是管理流缓冲的分配和替换,一般采用的是经典的 LRU 替换算法。stream buffer 预取器的核心部分是流过滤器和流缓冲管理部件,因为流过滤器决定了流识别的准确性,而流缓冲管理部件则决定了是否对识别出的数据流进行预取,两者共同决定了 stream buffer 预取器的性能。

### 2.2　Stream buffer 预取器工作流程

Stream buffer 预取器工作过程如下:上级发来的读访存请求首先在流缓冲中进行查询,如果流缓冲中存在该地址对应的数据,则流缓冲直接响应该访存请求,同时流缓冲检查内部保存的该数据流是否全部被访问过,如果是,则继续预取接下来的数据。如果当前读访存请求没有在流缓冲中命中,则将其发送给内存的访存命令队列进行正常访存,同时将地址发送给流过滤器,该地址会与访存地址历史记录表中的所有地址进行比较,如果存在相同地址,则一个新的流被检测出来,如果没有相同的地址,则将该地址进行相应步长的增减(通常进行 +1 操作,也就是下一个 cache 行),然后保存到访存地址历史记录表中。新检测流的相关信息送给流缓冲管理部件,该部件根据流缓冲是否有空闲项以及相应的替换策略决定是否给该数据流分配流缓冲空间,如果分配则将预取地址和需要替换的项发送给流缓冲进行预取操作。整个过程的流程如图 3 所示。

### 2.3　预取性能指标

在预取的相关研究领域中,通常用来评价预取性能的指标有 3 个:准确率(accuracy)、覆盖率(coverage)和及时性(timeliness)。

准确率是影响预取性能的一个重要指标,它指的是有用预取占全部预取数据的比例。所谓有用预取,指的是预取回的数据在被替换前被真正访问到,否则就是无用预取。对于预取来说,有用预取能够真正减少访存延迟,提升访存性能,而无用预取则会浪费访存带宽和功耗,同时也白白占用数据缓存空间。因此预取准确率越高则带来的好处越大,而预取准确率越低则带来的好处越小,甚至在某些访存资源紧张的情况下带来负面影响。

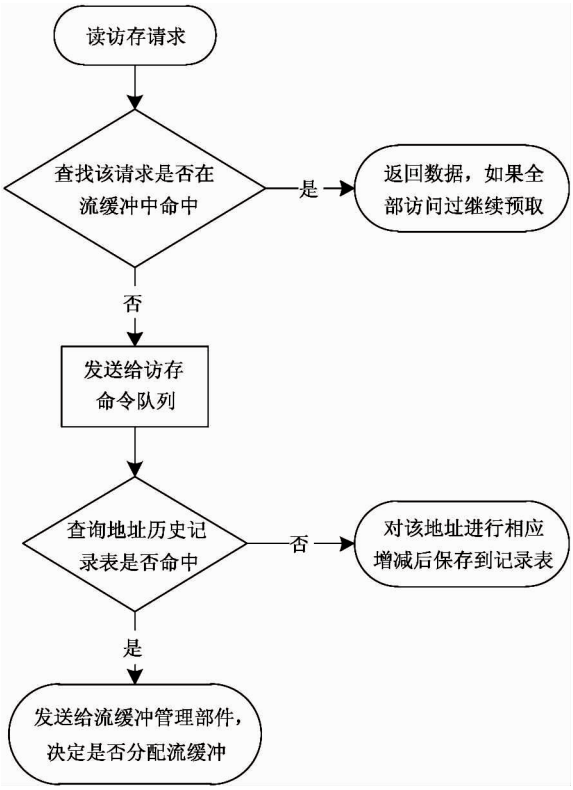


**图 3 Stream buffer 预取器工作流程**

覆盖率是影响预取性能的另一个重要指标，它指的是有用预取数据占到所有访存请求数据的比例。假设在某种预取方式下，预取的准确率很好，但是预取数据占所有访存请求数据的比例很小，那么此种预取对性能的提升也是非常有限的。因此对于预取来说，在准确率相同的情况下，覆盖率越高则会带来越显著的性能提升。

及时性是影响预取性能的第 3 个重要指标，它指的是预取数据返回的时刻与该数据被真正访问时刻的时间差，这个时间差越小意味着及时性越好。如果预取数据返回得过早，则意味着它可能挤占了其他数据的缓存空间，而且可能在使用前被替换出去，造成访存带宽和缓存空间的浪费；反之如果预取数据返回得过晚，则会发生该数据被访问时尚未取回的情况，于是发生访存等待，使得预取对掩盖访存延迟的效果大打折扣。所以及时性越好对访存性能的提升越明显。

综上所述，预取是一把双刃剑，有效的预取可以大大降低访存延迟，从而提升系统的整体性能，而无效的预取则会浪费访存带宽和功耗，给系统访存性能带来负面影响。预取的有效性非常依赖于预取的准确率、覆盖率和及时性。已有的内存控制器预取工作多集中于访存流识别、预取时机、预取数据量以及流替换策略的优化上，从而从一定程度上提高预取的准确率和覆盖率，鲜有对及时性的研究。多核片上系统中访存流的数量远大于 stream buffer 的项数，而不同的 IP 核也有不同的服务质量需求，因此如何合理分配预取资源从而满足整体的性能需求成为一个亟待解决的问题。

## 3 主控式 stream buffer 预取

本节提出的主控式 stream buffer 预取器综合考虑了不同访存流预取的及时性和冲突性来优化预取资源的分配和替换策略，进而提高访存性能。下面介绍主控式 stream buffer 预取器实现中的一些关键点。

### 3.1 多核片上系统 stream buffer 预取行为分析

多核片上系统中往往存在多个共享内存的 IP 核，这些 IP 核同时工作时会发出多个数据流。一般来说，非 CPU 的 IP 核访存具有较好的流特性，而 CPU 的访存特性则与具体的应用场景有关。通常指令类访存有较好的流特性，而数据类访存则与程序内部的数据规模有关，当数据规模较大比如运行大规模矩阵运算操作时具有较好的流特性，反之数据规模较小比如读取一些小型堆栈时则流特性较差，只会形成一些很短的数据流。

现有的方法大都将注意力集中在访存流识别、预取时机、预取数据量以及流替换策略上，这些方案将每个流都作为 stream buffer 的候选使用对象，而没有考虑一个流是否该使用 stream buffer。考量一个流是否该使用 stream buffer 需要依据这个流的两个参数：访存长度和访存间隔。

访存长度代表了一个连续访存流的长度，以 cache 行为单位。内存控制器中的 stream buffer 识别一个流通常是依据 2 ~ 4 个连续 cache 行地址访存，然后对接下来的 cache 行进行预取，预取的长度通常也为 2 ~ 4 个 cache 行。如果一个流的长度过短，那么就会发生这个流刚刚被识别出来就失效的

情况,后面预取的数据都成为无用的预取,严重影响访存性能。解决这个问题的方式可以通过增大流识别的长度要求或者采用延迟替换[7]策略,来避免访存长度很短的流的干扰。

访存间隔指一个流连续访问内存的时间间隔,以时钟周期为单位。现代高性能内存控制器中的访存队列通常有几十项,也就意味着可以调度几十个访存命令,调度方式多种多样,但是都会采用将落在同一个内存页的访存请求连在一起进行访存这一普遍原则,利用的是内存页打开的局部性来减少访存开销。stream buffer 预取之所以有效,利用的也是这一点,通过发出落在同一个页的连续访存请求将数据以很小的代价从内存中提前取回,从而达到减少访存延迟的效果。如果一个流的访存间隔很小,那么给这个流分配 stream buffer 预取的结果很可能是预取的请求还在访存队列中等待,后续真正的访存请求已经到达,此时需要等待预取数据的返回。这种情况下使用 stream buffer 完全体现不了预取的优势,因为即使不给该流分配预取空间,由于后续访存请求很快到达,访存队列通过调度也会将这些连续的访存请求连在一起进行访存,产生的效果与使用预取空间几乎相同,此时分配预取空间会造成预取空间的浪费。当然,一个流访存间隔过大也不适合使用 stream buffer 预取,因为可能会造成数据未使用被替换的情况,造成预取效率的下降。一个比较理想的情况是预取回的数据很快被使用,这样可以使得 stream buffer 预取达到最佳效果。

除了以上两个分配 stream buffer 预取空间需要考虑的因素之外,还有一个因素经常被忽略,那就是两个页访问冲突的流如何分配预取空间。考虑当多个性能指标相近的访存流竞争 stream buffer 预取空间时,如果仅仅使用 LRU 策略,则很有可能存在冲突的访存流无法获得 stream buffer 使用权,这会导致访存出现较多的页冲突,影响访存性能。而如果让存在访存冲突的流优先使用 stream buffer,则可以让存在冲突的访存流在一定程度上连续访问内存,从而可以减少访存冲突的次数,提升访存性能。

### 3.2　主控式 stream buffer 预取实现方案

图 4 所示为主控式 stream buffer 预取实现结构图,灰色部分为在传统 stream buffer 的基础上增加的逻辑。

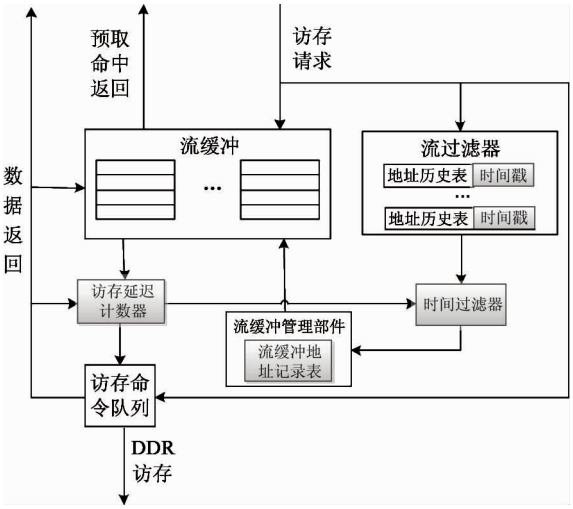


**图 4　主控式 stream buffer 预取器结构**

访存延迟计数器用来统计读访存请求从发出到接收到返回数据的时间延迟,该模块通过统计一段时间内多个读访存请求来获得一个读访存请求的平均响应时间,响应时间越长代表访存命令队列越繁忙,一般是由于访存命令密集或者访存冲突严重导致的。读响应时间被送到时间过滤器,用于决定是否对某个识别出的访存流进行过滤。

流过滤器中的访存地址历史记录表中每个地址项增加一个时间戳,这个时间戳代表该地址项产生的时间,用来计算与访存流的下一个请求的时间间隔。这个时间间隔被送往时间过滤器,与读响应时间共同决定是否对访存流进行过滤。

时间过滤器用来比较某个识别出的访存流的请求间隔和读响应时间,当两者的比值和差值未达到阈值时将其过滤掉,这意味着如果对这个访存流预取则及时性不好,因此该访存流不会被送往流缓冲管理部件,也不会参与流缓冲空间的分配。时间过滤器的阈值可以通过软件配置的方式进行调整,从而可以根据流缓冲使用的实际情况对其空间分配进行动态约束。

实际上时间过滤器除了将及时性不好的流进行过滤,从而可以使得预取资源的分配更加合理外,还可以起到控制预取时机和激进程度的作用。这是因

为时间过滤器根据读请求响应时间进行流过滤操作,而读请求响应时间取决于访存队列的繁忙程度以及访存冲突情况等,可以直接反映内存控制器的繁忙程度,这使得时间过滤器可以在第一时间做出反应。当内存控制器繁忙时,读请求响应时间会相对较长,此时时间过滤器会让访存间隔较长的访存流进行预取,从而变相达到减少预取量的作用,避免对正常访存请求的过多干扰,减小访存压力;当内存控制器空闲时,读请求响应时间会相对较短,此时时间过滤器会让访存间隔较短的访存流进行预取,从而增大预取的收益和内存控制器的吞吐率,提升整体的访存性能。

流缓冲管理部件中增加一个流缓冲地址记录表,用来记录已经占用流缓冲的访存流的 bank 和 row 地址信息。当需要为一个新的访存流分配流缓冲时,需要比较新的访存流和已经占用流缓冲的访存流之间的页冲突情况,在使用 LRU 替换策略的基础上优先替换出没有页冲突的流缓冲项,从整体上减少访存队列的页冲突。

图 5 给出了主控式 stream buffer 预取器的工作流程,深色部分是修改后增加的流程。

本方法实现的硬件开销非常小,只是增加了一些寄存器逻辑用来存储相应的时间和地址参数,不会对原有逻辑产生影响,也不会增加访存通路的延迟。

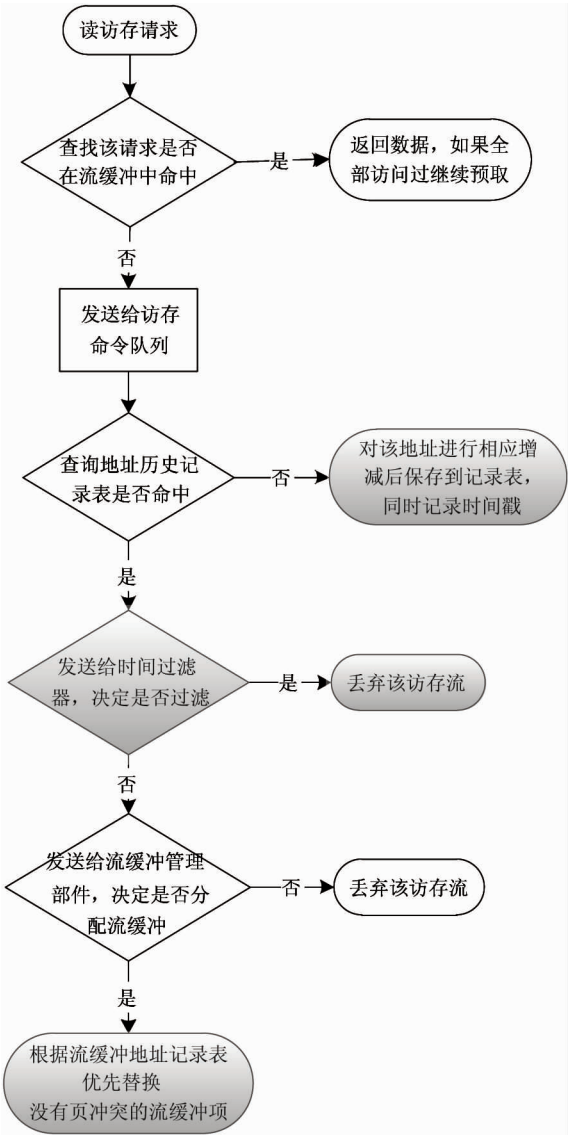


**图 5　主控式 stream buffer 预取器工作流程**

## 4　实验平台和实验结果

### 4.1　实验平台

本文的实验平台基于龙芯 2K1000 处理器的仿真环境,龙芯 2K1000 处理器是 2017 年发布的龙芯 2 号处理器系列的最新产品,是一款主要面向网络应用的高性能多媒体 SoC,结构如图 6 所示。

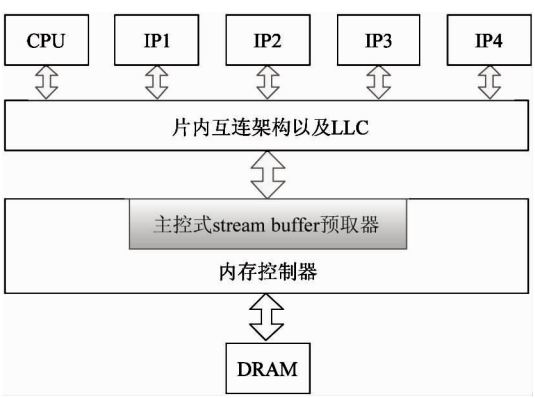


**图 6　龙芯 2K1000 仿真环境结构**

图 6 中的互联架构使用了 AMBA AXI 总线协议;内存控制器为龙芯自主研发的高性能内存控制器,读写命令队列分别有 16 项;stream buffer 预取器的流缓冲个数为 4,预取长度为 4 个 cache 行,使用 LRU 替换策略;DRAM 颗粒采用了 DDR3-800 内存标准,时序参数为 5/5/5/15,数据总线宽度为 64 bit,提供的最大带宽为 6.4 GB/s。

本实验平台共设置了 5 个仿真 IP 核:1 个模拟 CPU 访存行为,该核在随机的访存间隔内发出访存

请求,包括 4 个顺序访存流和一些随机地址的访存请求,平均访存带宽为 800 MB/s;其他 4 个模拟多核片上系统中其他访存的 IP 核,每个 IP 核的访存地址按照顺序发出,用于模拟访存空间局部性,使得同一个 IP 核的访存请求尽量在同一个页中命中。每个仿真 IP 核均可对访存地址、带宽需求以及 outstanding 数量进行配置。

为了研究不同应用场景下访存流带宽和冲突对多核片上系统的性能影响,我们设置了 5 组不同的带宽需求,如表 1 所示,其工作负载量逐渐增加,并且每组都进行 0 ~ 4 个不同数量访存流冲突的测试(以 BW1 为例,当运行 0 或 1 个访存流冲突时,则只有 IP1 工作,带宽需求为 800 MB/s,当运行 2 个访存流冲突时,IP1 和 IP2 工作,带宽需求分别为400 MB/s,以此类推),以便研究访存流存在冲突时是否合理使用 stream buffer 对性能的影响。

**表 1　IP 核的带宽需求(MB/s)**

| 工作负载 | 带宽需求 |
|---|---|
| BW1 | 800 |
| BW2 | 1600 |
| BW3 | 3200 |
| BW4 | 4200 |
| BW5 | 5000 |

### 4.2　实验结果

图 7 展示了使用龙芯 2K1000 自带 stream buffer 在不同带宽需求和访存冲突下的 IP 核实际访存带宽。

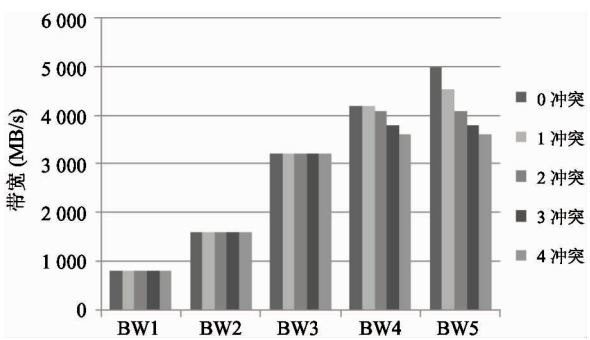


**图 7　龙芯 2K1000 不同场景下的 IP 核实际带宽**

从图中可以看出,当 IP 核带宽需求较小时,不同访存冲突数量并没有影响实际访存带宽,IP 核的访存带宽都可以得到满足。但是随着带宽需求的上升和访存冲突数量的增加,IP 核的实际访存带宽逐渐不能满足需求。同时可以看出,不同访存流冲突下 IP 核的实际访存带宽存在一个最大瓶颈带宽,比如在访存流冲突为 3 和 4 时,IP 核的最大访存带宽分别只能达到 3 800 MB/s 和 3 600 MB/s,超过该带宽需求则无法满足。

图 8 展示了使用主控式 stream buffer 时不同带宽需求和访存冲突下的 IP 核实际访存带宽。

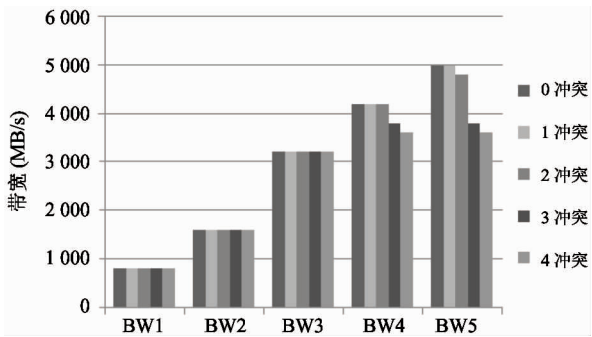


**图 8　使用主控式 stream buffer 时不同场景下的 IP 核实际带宽**

从图中可以看出,当 IP 核带宽需求较小时,是否使用主控式 stream buffer 并没有影响实际访存带宽。随着 IP 核带宽需求量的上升,IP 核的实际访存带宽也会达到一个瓶颈,但是与图 7 相比,实际带宽有明显不同。比如在 BW5 需求下,当冲突数量为 1 和 2 时,瓶颈带宽明显增大(分别从 4 550 MB/s 和 4 100 MB/s增加到 5 000 MB/s 和 4 800 MB/s,最大提升近 20%),但是当冲突数量为 3 和 4 时,瓶颈带宽与图 7 几乎相同。

经过分析,我们发现当 IP 核的带宽需求不高时,比如 BW1 和 BW2 带宽需求下,此时内存控制器访存队列相对比较空闲,预取数据的及时性都很好,所以两种方法并没有明显的区别,都是倾向于让数据需求量较大的流占据 stream buffer,从而最大发挥 stream buffer 预取的作用。

当 IP 核的带宽需求进一步增加,两种预取方式的区别开始体现出来:在原始 stream buffer 预取方式下,当访存流冲突个数从 0 到 4 的过程中,由于 IP 核访存流的带宽需求量较大,在只采用 LRU 策略的情况下,stream buffer 中的流缓冲项总是优先被 IP

核访存流占据,有剩余项时才会被带宽需求量较小的 CPU 访存流占据;而在采用主控式 stream buffer 预取方式下,当访存流冲突个数较少时,由于单个 IP 核访存流带宽需求量很大,此时 IP 核访存流使用 stream buffer 的及时性较差,所以此时 stream buffer 中的流缓冲项优先由带宽需求量较小的 CPU 访存流占据,而当访存流冲突个数继续增加时,由于单个 IP 核访存流带宽需求量逐渐减少,所以预取数据的及时性也逐渐变好,此时 stream buffer 中的流缓冲项优先被带宽需求量大的 IP 核访存流占据,有剩余项时才会被带宽需求量较小的 CPU 访存流占据。也就是说虽然访存流冲突个数较少时两种方式都能满足 IP 核的带宽需求,但是内在对 stream buffer 的使用则并不相同,而实际效果则会体现在访存延迟上,这在后面的实验结果中会有更加明显的体现和讨论。

图 9 展示了使用龙芯 2K1000 自带 stream buffer 在不同 IP 带宽需求和访存冲突下 CPU 的访存延迟情况(由于 CPU 对访存延迟非常敏感,因此仅对 CPU 访存延迟变化进行分析,其他 IP 核的访存延迟变化对性能提升并不明显)。

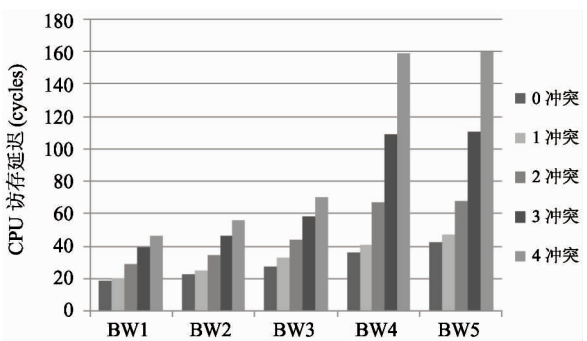


**图 9 龙芯 2K1000 不同场景下的 CPU 访存延迟**

从图中可以看出,在不同的带宽需求下,随着访存流冲突数量的增加,CPU 访存延迟也逐渐增加。增加的原因一方面是因为访存流冲突数量的增加导致内存整体访问延迟增加,更主要的原因是因为 IP 核访存流逐渐占据 stream buffer 的使用权,使得 CPU 访存流无法充分利用 stream buffer 预取来降低访存延迟,这样的结果在 IP 核访存流预取及时性较好时可以使得整体的访存延迟降低,但是在 IP 核访存流预取及时性不好时会导致整体访存性能的下降。

图 10 展示了使用主控式 stream buffer 时在不同 IP 带宽需求和访存冲突下 CPU 的访存延迟情况。

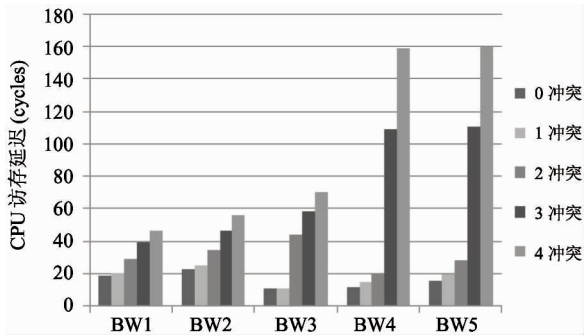


**图 10 使用主控式 stream buffer 时不同场景下的 CPU 访存延迟**

从图中可以看出,当 IP 核带宽需求较小时,是否使用主控式 stream buffer 并没有影响 CPU 的访存延迟。这主要是因为此时 IP 核访存流使用 stream buffer 预取的及时性较好,因此占用 stream buffer 进行预取,结果与图 9 相同。

当 IP 核的带宽需求逐渐增加时,两种预取方式下 CPU 访存延迟的区别开始体现出来:在原始 stream buffer 预取方式下,由于 IP 核访存流的带宽需求量较大,在只采用 LRU 策略的情况下,stream buffer 中的流缓冲项优先被 IP 核访存流占据;而在采用主控式 stream buffer 预取方式下,当访存流冲突个数较少时,此时 IP 核访存流使用 stream buffer 预取数据的及时性较差,所以此时 stream buffer 中的流缓冲项优先由 CPU 访存流占据;而当访存流冲突个数继续增加时,由于 IP 核访存流预取数据的及时性也逐渐变好,此时 stream buffer 中的流缓冲项优先被带宽需求量大的 IP 核访存流占据。由此产生的结果就是当访存流冲突个数较少时(比如为 0、1、2 时),CPU 访存延迟很小(从平均 45 拍下降到 16 拍,降低超过 60%),这主要是因为大部分 CPU 访存请求都会在 stream buffer 中命中,而当访存流冲突个数继续增加时,结果与图 9相同。

综上所述,主控式内存控制器预取可以使得多核片上系统中的预取资源分配更加高效合理,有效提高系统的访存带宽,降低系统的访存延迟。

## 5　结　论

本文对多核片上系统内存控制器端的访存流预取行为进行了深入的分析,提出了一种主动控制内存控制器端的预取策略。该策略综合考虑了不同访存流预取的及时性和访存冲突性,将预取数据及时性差的访存流进行过滤,使之在 stream buffer 资源紧张的情况下不占用流缓冲空间,同时利用流缓冲地址记录表使得存在冲突的访存流优先使用 stream buffer,进一步降低了访存冲突的概率。实验表明,该方法可以提升近 20% 的最大访存带宽,而对带宽需求小的访存 IP 核可以降低 60% 左右的访存延迟。

主控式内存控制器预取综合考虑了多核片上系统不同访存流使用预取资源对性能造成的影响。未来的工作包括进一步研究不同 IP 核的访存行为,比如结合软件层面获取更多的信息,从而更加精确地指导周期性猝发类型访存流的预取;此外还要研究如何使得不同层次的预取能够更好地相互配合,避免重复性预取以及相互之间的干扰,进一步提升系统的访存性能。

**参考文献**


[1] Jouppi N P. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers[C]. In: Proceedings of the 17th Annual International Symposium on Computer Architecture, Seattle, USA, 1990. 364-373

[2] Yedlapalli P, Kotra J, Kultursay E, et al. Meeting midway: Improving CMP performance with memory-side prefetching[C]. In: Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques, Edinburgh, UK, 2013. 289-298

[3] Palacharla S, Kessler R E. Evaluating stream buffers as a secondary cache replacement[C]. In: Proceedings of the 21st International Symposium on Computer Architecture, Chicago, USA, 1994. 24-33

[4] Mckee S A, Oliver C W, Wulf W, et al. Evaluation of dynamic access ordering hardware[C]. In: International Conference on Supercomputing, University of Virginia, USA, 1995. 125-132

[5] HurI, Lin C. Memory prefetching using adaptive stream detection[C]. In: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO06), Orlando, USA, 2006. 397-408

[6] Lee C J, Mutlu O, Narasiman V, et al. Prefetch-aware memory controllers[J]. *IEEE Transactions on Computers*, 2011, 60(10):1406-1430

[7] 陈新科. 面向多核处理器的内存控制器研究[D]. 北京:中国科学院大学,2015. 57-69

[8] 李文. 存储控制系统性能优化技术研究[D]. 北京:中国科学院计算技术研究所, 2005. 69-91


# Proactive control method for memory controller prefetching on multi-processor system-on-chip


Li Peng[*,**,***], Wang Jian[***], Zeng Lu[****], Wang Huandong[****]

([*] State Key Laboratory of Computer Architecture (Institute of Computing Technology, Chinese Academy of Sciences), Beijing 100190)

([**] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([***] University of Chinese Academy of Sciences, Beijing 100049)

([****] Loongson Technology Corporation Limited, Beijing 100095)



**Abstract**

Aiming at the competition of prefetching resources on the memory controller side caused by multi-IP (intellectual property) cores, a proactive control optimization method for memory controller prefetching is proposed for multi-processor system-on-chip (MPSoC). This method takes into consideration the prefetching timeliness and memory conflict of different streams, and filters the streams with poor timeliness, so that it does not occupy the stream buffer space when the stream buffer resources are tight. The address record table is used to give the conflicting streams priority to use the stream buffer, which further reduces the probability of memory conflicts. Experiments show that this method can increase the maximum memory bandwidth by nearly 20%, and reduce memory access latency by about 60% for the IP cores with low bandwidth requirement.



**Key words**: multi-processor system-on-chip (MPSoC), timeliness, memory access conflict, prefetch, stream buffer