MCS-DMA:一种面向SoC内DMA传输的内存控制器优化设计

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

摘 要: 当前主流片上总线协议——AHB存在访存带宽利用率较低的问题.本文基于SoC内DMA传输较多的特点,提出一种新的优化设计:在内存控制器内部增加MCS-DMA模块,并通过驱动程序将MCS-DMA模块与目标DMA传输绑定.一方面实现数据预取,提升了单个DMA传输时的总线带宽利用率;另一方面使访存请求在内存控制器内部流水化完成,提升多个DMA并发时的总线带宽利用率.将该设计应用到北大众志SK SoC后,单个DMA传输时的总线带宽利用率提升至100%,多个DMA并发时的总线带宽利用率从33.3%提升至85.5%,而芯片的设计面积仅增加2.9%.

关键词: 系统芯片; 内存控制器; 直接内存访问
中图分类号: TP302 **文献标识码:** A **文章编号:** 0372-2112 (2010) 03-0598-07

MCS-DMA:An Optimization Design of Memory Controller for DMA Transfers in SoC

HUANG Kan, TONG Dong, LIU Yang, YANG Shou-gui, CHENG Xu
(*Microprocessor Research and Development Center of Peking University, Beijing* 100871, *China*)

Abstract: Current mainstream on-chip bus protocol — AHB has a problem that the bandwidth utilization of memory accesses is quite low. This paper proposes a new optimization design based on the feature that there are massive DMA transfers in SoC. Proposed method adds MCS-DMA modules inside the memory controller and bind MCS-DMA modules to target DMA transfers via software drivers. On the one hand, it prefetches data to increase the bandwidth utilization of single DMA transfer; on the other hand, it makes memory requests pipelined inside the memory controller, which increases the bandwidth utilization of multiple parallel DMA transfers. After applying the design to PKUnity-SK SoC, the bus bandwidth utilization when transferring single DMA increases to 100%. When transferring multiple DMAs in parallel, the bus bandwidth utilization increases from 33.3% to 85.5%. However, the chip area only increases by 2.9%.

Key words: System-on-Chip; memory controller; direct memory access (DMA)

1 引言

随着半导体器件集成度的提升,系统芯片(System-on-Chip,SoC)设计规模日益增大.为了缩短研发周期和降低流片风险,设计时通常基于知识产权(Intellectual Property,IP)复用的方法学[1].根据文献[2]的统计,目前市场上,使用AHB(Advanced High-performance Bus)[3]总线接口的IP产品数量最多.故相当多的SoC都会使用AHB总线连接内部各个IP模块.

然而,AHB总线本身存在性能缺陷.文献[4,5]基于模拟评测结果指出,相比于其他总线协议,由于其不支持流水化交易[2],AHB总线访存时带宽利用率较低.文献[6]进一步模拟了内存控制器和IO设备的行为,并分析了多层总线的通信架构,指出即使使用多层总线架构也不能显著提升性能.总线带宽利用率较低,会导致设备因带宽不足而降低性能.对于具有硬实时性访存带宽需求的设备(如显示控制器),带宽不足将导致设备无法正常工作.从另一个角度看,较低的带宽利用率也迫使总线必须工作在更高的时钟频率下,才能满足设备带宽需求,增加了系统功耗.上述问题在北大众志-SK系统芯片[7](简称SK SoC)流片后的性能评估中凸显出来.

为了在后续设计中解决该问题,最直接的方法是升

收稿日期:2009-01-10;修回日期:2009-06-25
基金项目:国家863高技术研究发展计划(No.2006AA010202)

级到更高效率的总线架构,如 AXI(Advanced eXtensive Interface)[3].但由于项目时间和成本的限制,短期内完成升级十分困难.除此之外,人们也尝试了其他优化方法.在总线端,文献[8]使用更长的突发(Burst)交易提升总线带宽利用率,但当访存延迟较大时,该方法提升效果有限,并且增加突发交易长度使每次交易占用总线更久,增加了其他设备的最大访存延迟.文献[9]提出了一种与 AHB 完全兼容的总线架构,其支持同时发出多个交易,但该方法需要对每个主从设备接口进行协议转换,改变了总线架构,实现复杂度较大.在内存控制器端,文献[10]基于多媒体应用访存的规律性,在内存控制器内部自动生成访存地址,从而减少地址总线的通信量,但其改变了主从设备间的通信方式,因此需要修改访存主设备模块后才能使用.文献[11]通过提前终止内存端的突发传输和提前激活行两项技术,减少访存延迟,从而提升访存带宽利用率,但其访存延迟的减少能力有限,故带宽利用率提升效果也有限.

经观察发现,SK SoC 以及当前主流 SoC(如 ST Nomadik[12]、TI OMAP[13]等)具有一共同特征,即内部模块多数都基于直接内存访问(Direct Memory Access,DMA)方式进行数据传输,DMA 传输在 SoC 内所有访存交易中,占据了很大比重.优化 DMA 传输的总线带宽利用率可以有效提升 SoC 系统整体访存性能.因此,本文提出一种 DMA 访存优化方法——MCS-DMA(Memory Controller Side DMA).该方法借鉴了文献[14]中通过流缓冲器(Stream Buffer)预取以减少处理器访存延迟的思想,无须改变 SoC 现有总线架构,仅在内存控制器接口处和 DMA 驱动程序处做少量修改,通过预取 DMA 传输所需数据,减少访存延迟,并使各设备的访存请求在内存控制器内部流水化完成,从而提升总线带宽利用率.将该方法应用到北大众志 SK SoC 的设计中,仅增加较少的设计面积,即可有效提升 DMA 传输的总线带宽利用率.

后文将首先分析 AHB 总线访存带宽利用率的原因,再详细介绍 MCS-DMA 方法,接着以 SK SoC 为应用实例评估 MCS-DMA 方法的优化效果,最后进行总结.

2　DMA 传输性能分析

本节通过在 SK SoC 中的评测,详细介绍 AHB 总线带宽利用率低的现象,并分析其原因.

本文按表达式定义各设备交易的总线带宽利用率:

$$e = \frac{t_d}{t_o} \times 100\% \tag{1}$$

式中,t_d 为数据传输周期数,t_o 为总线被该设备占用的周期数.总线带宽利用率描述了设备对其所占用总线带宽的使用效率.最佳状况下,总线带宽利用率可达 100%,此时设备充分利用了所分配到的总线带宽.

本文基于后文 4.1 节中介绍的实验环境,分别评测了 SK SoC 中五个 I/O 控制器单独进行 DMA 读写访存时的总线带宽利用率,如表 1 所示.

表 1　SK SoC 各设备 DMA 传输总线带宽利用率

交易类型		传输周期	数占用周期数	带宽利用率
VGA*	读	512	1657	30.9%
SATA	读	256	699	36.6%
	写	256	352	72.7%
Ethernet	读	256	467	54.8%
	写	256	336	76.2%
USB	读	256	477	53.7%
	写	256	351	72.9%
NAND	读	528	1410	37.5%
	写	528	791	66.8%

根据表 1 中的数据,DMA 传输写交易的总线带宽利用率较高,几何平均值达到 72.1%.但 DMA 传输读交易的总线带宽利用率较低,几何平均值仅有 41.6%.

DMA 读交易访存带宽利用率偏低的原因是总线传输与内存传输间的并行度偏低.如图 1(*a*)所示,进行 DMA 写交易时,总线传输在数据被全部写入内存控制器内部缓存后,即可结束,然后开始下一次总线传输.下一次总线传输可以和前一次交易的内存传输并行进行,所以 DMA 写交易的带宽利用率较高.并且,写交易的带宽利用率还可以通过增加内存控制器内部缓存大小进一步提升.而对于 DMA 读交易,如图 1(*b*)所示,必须完成①内存控制器接收交易请求、②控制器发出访存请求、③数据从内存返回和④总线主设备接收数据四个阶段后才能结束.在等待数据返回的过程中,总线上不能开始新的交易,因此总线带宽利用率较低.此时,

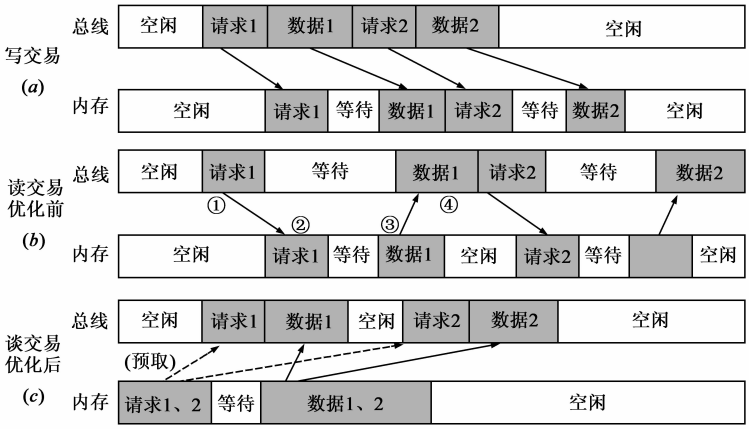

图1　访存交易过程示意图

* VGA 模块只能发出读交易.

$$e = \frac{t_{\text{burst}}}{l_b + t_{\text{burst}}} \times 100\% \quad (2)$$

其中 t_{burst} 为单次交易的数据传输周期数,即突发交易(Burst)的拍数,l_b 为单次交易延迟的总线周期数.根据表达式,通过减少每次交易的延迟,可以提升总线带宽利用率.为降低 DMA 读交易的访存延迟,本文提出使用 MCS-DMA 方法.

3 MCS-DMA 方法

本节首先介绍 MCS-DMA 的基本原理和设计基本结构,然后介绍其工作流程,并分析如何配置参数使总线交易延迟最小,最后介绍针对特殊 DMA 控制器所需做的修改.

3.1 MCS-DMA 基本原理

MCS-DMA 的基本原理是在内存控制器内部增加若干个和 DMA 控制器功能相似的模块——MCS-DMA.通过软件控制,使 MCS-DMA 预取目标 DMA 传输所需数据,当 DMA 传输总线交易到达内存控制器时可以直接返回所需数据,从而在总线上隐藏了访存延迟,提升总线带宽利用率.

图 1(*c*)中给出了读交易优化后的示意图.值得注意的是,MCS-DMA 使访存交易与总线交易间不再需要一一对应,只需保证所请求数据块整体对应即可,从而使单次访存请求数据量可以大于单次总线请求的数据量,减少了访问外部内存的次数,提升了内存访问效率.

3.2 MCS-DMA 设计基本结构

MCS-DMA 设计的基本结构如图 2 所示,其中用阴影标识的模块为内存控制器原有功能部件.

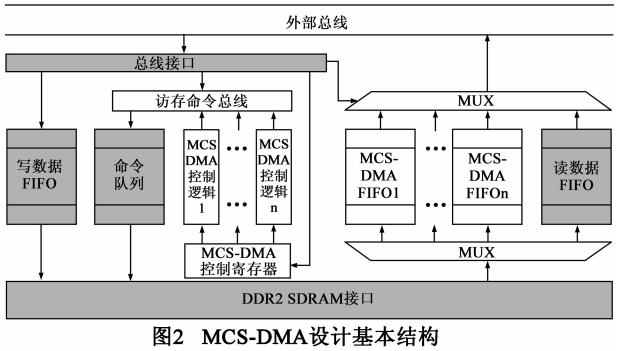

图2 MCS-DMA设计基本结构

为了支持多个并发的 DMA 传输,在一个控制器内部需实现多组 MCS-DMA,组数 n 应根据 SoC 中可能同时进行的 DMA 传输个数确定.每一组 MCS-DMA 由控制寄存器、控制逻辑和 FIFO 三部分组成.控制寄存器是软件配置的接口;控制逻辑用于产生实际的访存命令,写入到命令队列中; FIFO 用于缓存交易返回的数据.FIFO 为异步时钟 FIFO,写端口时钟为 DDR2 时钟,读端口时钟为总线时钟.由于 DDR2 的数据传输率是时钟频率的两倍,为了带宽匹配, FIFO 位宽为 DDR2 接口位宽的两倍.

未启用 MCS-DMA 功能时,读内存交易通过总线接口模块转换成相应的访存命令,写入命令队列,再经过调度后,发送给 DDR2 接口.从内存条返回的数据先写入读数据 FIFO,再从读数据 FIFO 返回总线.

启用 MCS-DMA 功能后,由于 MCS-DMA 预取了所需数据,与 MCS-DMA 相对应的 DMA 读交易可直接从对应 MCS-DMA FIFO 中取出相应数据返回总线.而其他交易的数据通路保持不变,因此该优化方法不会影响非 DMA 访存的性能.

由于总线接口和各组 MCS-DMA 控制逻辑都会向命令队列发送命令,必然存在资源竞争,故增加一条访存命令总线,通过总线仲裁器管理竞争.仲裁策略应根据实际需求确定.SK SoC 中选择实现最简单的固定优先级仲裁策略,总线接口优先级最高,各 MCS-DMA 的优先级从 1 到 n 逐次降低.

高性能的内存控制器为了能处理多个并发的访存请求,其内部设计必定支持流水化交易.从 MCS-DMA 设计基本结构可以看出,MCS-DMA 方法利用了这一特性,使各模块访存请求的竞争发生在内存控制器内部,使其能流水化完成,从而提升了带宽利用率.

3.3 MCS-DMA 的工作流程

MCS-DMA 的软件配置流程如图 3 所示:

图3 MCS-DMA软件配置流程

驱动程序在启动目标 DMA 传输前,先将 DMA 传输的起始地址、交易长度和 DMA 设备号写入 MCS-DMA 的对应功能寄存器中,并启动 MCS-DMA 预取.之后,再配置目标 DMA 控制器,启动 DMA 传输.后续的传输完全由硬件控制,不再需要软件干预,直到 DMA 传输完成.可以看出,每个 MCS-DMA 对各个 DMA 设备都是通用的.因此实际所需的 MCS-DMA 数量可以少于 SoC 中 DMA 控制器的数量.

为了保证在 DMA 传输开始前,所需数据已经被写入到 MCS-DMA FIFO 中,应在启动 MCS-DMA 和启动目标 DMA 传输两条指令间保留一定延迟,实际应用中可通过插入一些无关指令来实现.当指令数量 N_{ins} 满足表达式(3)时,可以确保 DMA 传输开始前,数据已经被写入到 MCS-DMA FIFO 中.

$$N_{\text{ins}} \geqslant l_d \times \frac{T_{\text{ddr}}}{T_{\text{cpu}}} \quad (3)$$

式中 l_d 为访存延迟的 DDR2 周期数, T_{ddr} 为 DDR2 时钟

周期,T_{cpu}为 CPU 时钟周期.

MCS-DMA 硬件控制流程如图 4 所示.当 MCS-DMA 被使能后,首先进行第一次访存交易,交易的地址从 DMA 起始地址顺序递增.如果 DMA 长度大于 MCS-DMA FIFO 的容量,则交易的数据量等于 MCS-DMA FIFO 容量;否则,交易数据量为 DMA 长度.如果尚未完成全部 DMA 数据的预取,则监测 MCS-DMA FIFO 中数据量的变化.DMA 总线读交易进行时,会不断从 MCS-DMA FIFO 中弹出数据,为后续返回的数据留出空间.当 MCS-DMA FIFO 剩余容量大于一次后续访存交易的数据量(该参数通过软件配置)时,MCS-DMA 控制逻辑发起新一次访存请求,返回的数据将被写入 MCS-DMA FIFO 中,供后续的目标 DMA 传输访问.如此往复,直至目标 DMA 所需数据全部预取完成后,MCS-DMA 停止工作,直到下一次被软件启动.

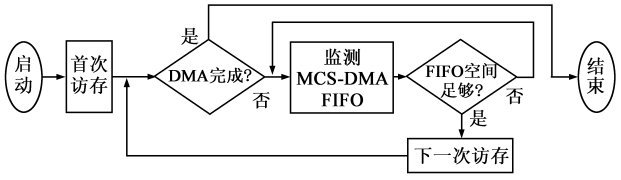

图4　MCS-DMA硬件控制流程

MCS-DMA 硬件控制流程在数据一致性方面,为了确保 DMA 总线交易所读取的数据与此刻内存中的数据一致,需保证从MCS-DMA预取访存到 DMA 总线交易完成的过程中,内存里相应数据不被修改.上述这一点要求不难保证.由于 DMA 传输过程对软件透明,软件须保证在 DMA 传输过程中,对应内存数据保持不变.MCS-DMA 方法可视作提早启动的 DMA,故在软件里将一致性保护的时间起点,从启动 DMA 处提早到启动 MCS-DMA 处即可满足一致性要求.

3.4　MCS-DMA 的参数配置

由于 DMA 传输数据量很大,往往无法将所有数据一次性缓存到 MCS-DMA 内,只能在 DMA 传输的过程中不断发起新的访存交易,向 MCS-DMA FIFO 中持续缓存数据.后续访存若不能在相应 DMA 总线交易到达前完成,将导致 DMA 传输过程出现总线交易延迟.本节将量化分析如何配置 MCS-DMA 的各项参数,可保证 MCS-DMA 访存交易的适时完成,以减少整个 DMA 传输的交易延迟.

先考虑系统中只有一个 DMA 主设备访存的情况.要保证在目标 DMA 读交易到达之前完成数据缓存,应满足下列两个条件:

(1)DMA 数据传输需求带宽小于等于内存系统可提供的最大带宽.

(2)MCS-DMA 预取访存的延迟小于等于 DMA 总线交易取走 MCS-DMA FIFO 中所剩数据所需的最少时间.

条件(1)通过 SoC 整体架构设计保证,而条件(2)则通过 MCS-DMA 参数的设置保证.

由条件(2)得,

$$l_d \times T_{ddr} \leqslant \frac{2W_{ddr}(M_{FIFO}-P)}{b_{bus}} \tag{4}$$

整理后,得

$$M_{FIFO}-P \geqslant \frac{b_{bus}}{\frac{2W_{ddr}}{T_{ddr}}} \times l_d = \frac{b_{bus}}{B_{ddr}} \times l_d \tag{5}$$

式中,W_{ddr}为 DDR2 接口位宽,B_{ddr}为 DDR2 理论带宽,b_{bus}为该设备可获得的最大总线带宽,M_{FIFO}为 FIFO 深度,P 为一次后续 MCS-DMA 预取的数据在 MCS-DMA FIFO中所占的深度.

M_{FIFO}是硬件设计时静态配置的参数,M_{FIFO}值越大,所占用的硬件资源就越多.P 为可软件动态配置的参数,P 值越大,内存端的访问粒度就越大,内存端的访问效率就会越高.表达式(5)为 M_{FIFO}和 P 的配置提供了参考依据.该设备可获得的最大总线带宽与 DDR2 的理论带宽的比值越大,MCS-DMA FIFO 所需的深度也就越大.

当系统中同时存在多个 DMA 设备访存时,访存冲突会增大 l_d,而总线冲突又会降低 b_{bus}.因此当设备位于同一总线时,总线冲突影响更大,M_{FIFO}与 P 的差值可减小一些;当设备位于不同总线时,只存在访存冲突,需要增大 M_{FIFO}与 P 的差值.

3.5　对其他类型 DMA 的支持

为了满足某些 IP 的特殊需求,SoC 中还可能存在一些特殊的 DMA 控制器设计.以 SK SoC 为例,其中的 VGA 控制器在 DMA 启动后,循环的从显存区域内读取数据,循环过程由硬件控制,软件不干预.以太网控制器使用 Scatter-Gather DMA 控制器,软件只配置描述符链表的入口地址,即能完成若干个不连续内存区域的数据传输.通过实现与目标 DMA 具有完全相同工作模式的特殊 MCS-DMA 模块,如循环模式或 Scatter-Gather 模式,可以支持相应模式的 DMA 控制器.

4　实验评估

本节评估应用 MCS-DMA 到 SK SoC 的实际效果.首先,根据 SK SoC 参数确定合适的 MCS-DMA 参数配置,评估设备单独进行 DMA 传输时的带宽利用率,再评估多设备同时进行 DMA 传输时各设备的带宽提升,最后在 FPGA 芯片上评估在低时钟频率下真实应用的性能提升.

4.1　模拟实验环境

实验评估基于寄存器传输级(Register Transfer Level,RTL)硬件代码,使用 Synopsys VCS MX Y-2006.06 模

拟器进行硬件模拟.SK SoC 总线和内存的基本参数如表 2 所示.各模块典型 DMA 传输的参数如表 3 所示.

表 2 SK SoC 硬件平台总线与内存参数

名称	频率(MHz)	位宽(Bit)	所连接模块	仲裁策略
高速总线	600	64	CPU、VGA	静态优先级，VGA 高于 CPU
外设总线	200	32	Ethernet、USB、SATA、NAND	Round-Robin 轮转仲裁
DDR2	266	64	高速总线、外设总线	Round-Robin 轮转仲裁

表 3 SK SoC 各模块典型 DMA 传输参数

模块	带宽(MB/s)	DMA 长度(Byte)	突发拍数
VGA	225	4096	16
SATA	300	1024	8
Ethernet	125	1024	16
USB	最大带宽	1024	16
NAND	最大带宽	2112	8

虽然 USB 和 NAND Flash 的 I/O 带宽不大,但由于其控制器工作时,必须先从内存读取全部数据到 I/O 控制器缓存内,才能开始向外部设备传输数据.因此其需求带宽为总线的最大带宽,以尽可能快的完成从内存到 I/O 控制器内部缓存的数据传输.

4.2 MCS-DMA 参数配置

SK SoC 中高速总线上的 DMA 访存设备只有 VGA 控制器,但其进行图形和图像两个通道的 DMA 访存,故在内存控制器的高速总线接口上设置两组 MCS-DMA.外设总线上 DMA 访存设备较多,但考虑到所有设备同时发出 DMA 传输的场景较少,为了节省芯片面积,仅在外设总线接口上设置 4 组 MCS-DMA.

根据实际 RTL 模拟结果,当访存页失效时,访存延迟最大为 $l_d = 21$.代入表达式得,$M_{\mathrm{FIFO}} - P \geqslant 24$ 对于高速总线接口,

$M_{\mathrm{FIFO}} - P \geqslant 4$ 对于外设总线接口,

考虑到设备间的访存带宽竞争,设定高速总线上 MCS-DMA FIFO 深度为 32,外设总线上 MCS-DMA FIFO 深度为 16.P 值配置为 8

4.3 单设备 DMA

首先评估 MCS-DMA 对单个设备单通道 DMA 传输性能的优化.优化前后,各设备单独进行 DMA 读交易时的总线带宽利用率如图 5 所示.可以看出,使用 MCS-DMA 后,各设备总线带宽利用率全部达到 100%,说明 DMA 总线交易访存延迟为 0,实现了设计的预期目标.

4.4 多设备 DMA

接下来评估五个设备同时进行典型 DMA 读交易时各设备和外设总线的实际带宽,如图 6 所示.需要指出的是,设备与总线带宽计算时所基于的时段不同:计算设备带宽的时段,是从该设备 DMA 启动开始,到该设备 DMA 传输完成;计算总线带宽的时段,是从总线上第一个 DMA 总线交易开始,到总线上最后一个 DMA 总线交易结束.由于后者计算所用的时段更长,因此即使各设备带宽总和大于总线的理论带宽,也是合理的.

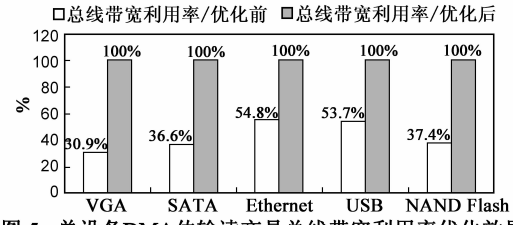

图 5 单设备DMA传输读交易总线带宽利用率优化效果

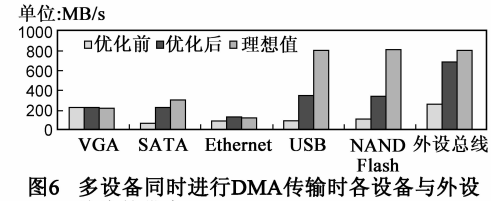

图6 多设备同时进行DMA传输时各设备与外设总线的带宽

VGA 由于位于高速总线上,带宽未受其他设备影响,达到了理想带宽.其余四个设备均位于外设总线上,总线竞争导致各设备实际带宽均低于理想值,设备性能受到较大影响.使用 MCS-DMA 方法优化后,Ethernet 模块带宽达到理想带宽,SATA、USB 和 NAND 三个模块虽未能达到理想带宽,但带宽平均提升了 2.4 倍.设备带宽未能达到理想带宽的原因是,外设总线的理论带宽仅有 800MB/s,并且 Round-Robin 仲裁使带宽较平均的分配给了各个设备.优化后,外设总线的带宽利用率从 33.3%提升到 85.5%.总线带宽利用率虽未能达到 100%,但这是由于设备间的授权转换引起了总线空闲,而并非由访存延迟造成.综上,实验评测结果表明,MCS-DMA 方法能在多设备同时进行 DMA 传输时,有效提升总线带宽利用率.

4.5 FPGA 原型评测

SK SoC 是一款面向 UMPC 设计的芯片,功耗控制至关重要.系统在运行过程中,如果发现当前系统较空闲,会自动降低时钟频率,以节省功耗.在降低时钟频率的同时,须确保系统能正常显示,因此必须使 VGA 控制器始终拥有足够的访存带宽.

将 SK SoC 的 RTL 设计下载到一片 Xilinx Virtex-4 LX200 FPGA 芯片上,并运行 Linux 操作系统.将两条 AHB 总线和 DDR2 的时钟频率均设为 33MHz,评估此时 VGA 控制器在各种显示模式下能否稳定工作,结果如表 4 所示.VGA 控制器之所以在某些模式下会出现不稳定的现象,是由于 VGA 控制器的访存带宽会因处理器访存而减小.

表 4　优化前后 VGA 最优工作模式对比

分辨率	色彩	刷新率	需求带宽	优化前	优化后
640×480	16bpp	60Hz	36.9MB/s	稳定工作	稳定工作
640×480	16bpp	75Hz	46.6MB/s	稳定工作	稳定工作
800×600	16bpp	60Hz	57.6MB/s	稳定工作	稳定工作
800×600	16bpp	75Hz	72MB/s	不稳定	稳定工作
640×480	32bpp	60Hz	73.7MB/s	不稳定	稳定工作
640×480	32bpp	75Hz	92.2MB/s	无法工作	稳定工作
800×600	32bpp	60Hz	115.2 MB/s	无法工作	稳定工作
800×600	32bpp	75Hz	144MB/s	无法工作	稳定工作

结果显示,应用 MCS-DMA 方法后,在相同的低时钟频率下,VGA 控制器可以支持更高质量的色彩模式和更快的刷新频率.

4.6　软硬件开销

从性能角度看,MCS-DMA 方法在软件开销上仅增加了几条配置寄存器的指令,其性能影响可以忽略不计.从实现难度看,由于多数的 DMA 控制器都需软件配置 DMA 传输的起始地址和交易长度,而主设备号在 SoC 设计完成时就已经确定,所以在驱动程序中完成对 MCS-DMA 所需参数的配置是非常容易的.

SK SoC 共增加了 6 组 MCS-DMA 控制逻辑,6 组控制寄存器,6 个 MCS-DMA FIFO(2 个 32×128 位,4 个 16×128 位).面向 TSMC 0.13um LV 工艺,使用 Synopsys Design Compiler Z-2007.03-SP3 工具进行逻辑综合后,设计面积增加约 0.9mm^2,仅占整个 SK SoC 设计面积的 2.9%.证明 MCS-DMA 方法所增加的硬件开销是可接受的.

由于内存控制器时序关键路径位于访存调度逻辑内,而 MCS-DMA 方法对内存控制器的修改仅位于内存控制器接口处,且控制逻辑简单,因此未影响内存控制器的设计时序.

5　结论

本文针对 SoC 中 DMA 传输量较大的特点,对 DMA 传输进行专项优化,提出在内存控制器中增加 MCS-DMA 模块,并通过驱动程序进行控制,有效解决了 AHB 总线访存带宽利用率较低的问题.在北大众志 SK SoC 实际设计中的评估结果表明,应用 MCS-DMA 后,仅增加了 2.9%的设计面积,使各设备单独进行 DMA 传输的总线带宽利用率均提升至 100%;多个设备 DMA 传输同时进行时,总线带宽利用率从 33.3%提升至 85.5%;在低时钟频率下,VGA 控制器可工作在带宽需求更大的显示模式.但是,本文方法也存在一定局限性,其只适用于 DMA 访存较多的系统中.DMA 访存较少时,优化效果不明显,需要在后续工作中进行改进.

参考文献:

[1] Michael Keating, Pierre Bricaud. Reuse Methodology Manual for System-On-A-Chip Designs (3rd Edition)[M]. Norwell, USA: Kluwer Academic Publishers, 2002.

[2] Power. org. Embedded bus architecture report [EB/OL]. http://www. power. org/resources/downloads/Embedded_Bus_Arch_Report_1.0.pdf, 2008.

[3] ARM Ltd. AMBA 2.0/3.0 Specifications[EB/OL]. http://www. arm. com/products/solutions/AMBAHomePage. html, 2008.

[4] Mirko Loghi, et al. Analyzing on-chip communication in a MPSoC environment[A]. Design, Automation and Test in Europe Conference and Exhibition[C]. Paris, France: IEEE Computer Society, 2004.752 – 757.

[5] Martino Ruggiero, et al. Scalability analysis of evolving SoC interconnect protocols[A]. International Symposium on System-on-Chip[C]. Tampere, Finland: IEEE Inc, 2004.169 – 172.

[6] Simone Medardoni, et al. Capturing the interaction of the communication, memory and I/O subsystems in memory-centric industrial MPSoC platforms[A]. Design, Automation and Test in Europe Conference and Exhibition[C]. Nice Acropolis, France: IEEE Inc., 2007.660 – 665.

[7] 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)

[8] 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.
Lei Wang, 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)

[9] Sanghun Lee and Chanho Lee. A high performance SoC on-chip-bus with multiple channels and routing processes[A]. IFIP International Conference on Very Large Scale Integration and System-on-Chip[C]. Nice, France: IEEE Computer Society, 2006.86 – 91.

[10] Lee Kun-bin, 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.

[11] Chih-da Chien, et al. A low latency memory controller for video coding systems[A]. International Conference on Multimedia and Expo[C]. Beijing, China: IEEE Computer Society, 2007.1211-1214.

[12] Maurizio Paganini. Nomadik: A mobile multimedia application processor platform[A]. Asia and South Pacific Design Au-

tomation Conference[C]. Yokohama, Japan: IEEE Computer Society Press, 2007. 749 – 750.
[13] James Song, et al. A low power open multimedia application platform for 3G wireless[A]. IEEE International Systems-on-Chip Conference[C]. Tampere, Finland: IEEE, 2003. 377 – 380.
[14] Sally A. McKee, et al. Dynamic access ordering for streamed computations[J]. IEEE Transactions on Computers, 2000, 49(11): 1255 – 1271.

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.

刘 洋 男,1985 年生于重庆,北京大学计算机系硕士研究生.主要研究方向为系统芯片的设计与验证.

杨寿贵 男,1983 年生于云南大理,北京大学计算机系硕士研究生.主要研究方向为系统芯片的设计与验证.

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.