中国集成电路 CIC
China Integrated Circuit

基于 Crossbar 的多通道 DMA 控制器设计与实现

陈伍敏[1],刘荣生[2],罗闳[3],郭东辉[3]
(1.厦门大学物理系,福建 厦门,361005;
2.福建新大陆自动识别技术有限公司,福建 福州,350105;
3.厦门大学电子工程系,福建 厦门,361005)

摘要:本文给出了一种基于 Crossbar 的多通道 DMA 控制器的设计方案,它能有效地提高 DMA 数据传输的效率和减少系统 CPU 的中断次数,保证多核 SOC 系统的任务执行效率及传输接口的通信实时性。经 FPGA验证表明,所设计的多通道 DMA 控制器比传统的 DMA 有更好的效能及性价比。

关键词:DMA 控制器;Crossbar 结构;多通道传输;实时性

A Design of Multichannel DMA Controller Based on Crossbar

CHEN Wu-min[1],LIU Rong-sheng[2],LUO Hong-yin[3],GUO Dong-hui[3]
(1. Department of Physics, Xiamen University, Xiamen 361005,China;
2.Fujian Newland Auto-ID Tech.Co.,Ltd. Fuzhou,350105,China;
3. Department of Electronic Engineering , Xiamen University, Xiamen 361005,China)

Abstract: This paper gives a design scheme of multichannel DMA controller. It can improve the efficiency of DMA data transmission efficiently and interrupts the CPU as few as possible. It ensures the task implementation efficiency and the communication real-time of multi-core SOC systems. The result of FPGA verification shows this multichannel DMA controller has better performance than a traditional DMA.

Keywords: DMA controller; Crossbar architecture; multichannel transmission; real-time

1 引言

为了提高系统工作效率,SOC (System On a Chip) 系统通常采用存储器直接访问 (DMA) 技术来处理系统模块间的数据通信。然而,传统的 DMA 控制器 (DMAC) 不能保证稳定的传输带宽,且会频繁地中断系统微处理器 (CPU)[1],无法满足一些对实时性要求较高的系统要求。

针对这个问题,人们已提出了一些改进方法,如

在 DMA 控制器加入仲裁模块[2],对多个传输请求进行排序,通过存储器配置模式减少 CPU 的配置时间和中断次数，但还不能保证对多任务传输的宽带需求 ;也有把传输任务分为实时任务和非实时任务的，按实时优先级实现系统的实时传输[3],但是仍需系统对任务进行优先级排序，占用系统处理时间；文献[4-7]分别通过预存取和写循环、加入重排列单元、链模式和双缓冲器以及采用不同大小的缓存等方法提高 DMA 控制器的传输效率。但这些改进方法均只能保证 DMA 控制器在同一时间传输单一任务，为了实现系统多任务实时传输，本文拟引入 Crossbar 交换结构，并设计相应的多通道传输 DMA 控制器，保证系统多任务传输的同时提高系统响应的实时性。

为了具体介绍本多通道 DMA 控制器的设计方案,下面首先对基于 Crossbar 的多通道 DMA 控制器的工作原理进行详细分析，接着具体说明各个模块的设计实现，然后对设计进行验证和分析比较实验结果,并进行总结。

2 多通道 DMA 控制器

传统 DMA 控制器是采用共享总线方式来实现数据传输的,系统实现数据传输的过程为：(1) 首先设备向 DMAC 发出 DMA 请求。(2) DMAC 接到设备请求后,向 CPU 发出总线请求,请求接管系统总线。(3) CPU 在执行完当前指令周期后,向 DMAC 发出总线响应信号。(4) CPU 脱离对系统总线的控制,DMAC 接管控制系统总线。(5) DMAC 向设备发出应答信号。(6) DMAC 在存储器与设备之间进行数据传输。(7) 当设定的数据传输完后,DMAC 撤销总线请求信号，同时脱离对总线的控制,CPU 检测到总线请求信号变为无效后,撤销总线响应信号,恢复对系统总线的控制,同时跳回到中断前的状态。

但对于如图 1 所示的典型多媒体应用系统,内部有哈夫曼解码模块 (Huffman Decoder)、图像解码模块 (MPEG Decoder)、USB 模块、SPI 模块、SD 卡控制模块 (SDC) 和数模转换模块 (DAC) 等,哈夫曼解码模块和图像解码模块需要跟 USB 模块、SD 卡控制模块或 SPI 模块通信，即同时存在几路数据传输并且传输数据的模块是变化的，采用共享总线的 DMA 控制器无法同时处理多个通道任务，现有的多通道 DMA 控制器交换数据的设备只能是固定的，因而无法胜任这种系统需求。对于这类应用需要,本文引入了 Crossbar 互联结构,用 Crossbar 总线来代替共享总线。Crossbar 被称为交叉开关矩阵或纵横式交换矩阵,不会因为带宽资源不够而产生阻塞 [8]。图 2 所示是全 Crossbar 总线拓扑结构的原理图。由于全 Crossbar 总线所需连接线数量较多, 在实际中经常简化为部分 Crossbar 总线拓扑结构，即将不需要通信的模块间的连接去掉,以节省面积[9]。

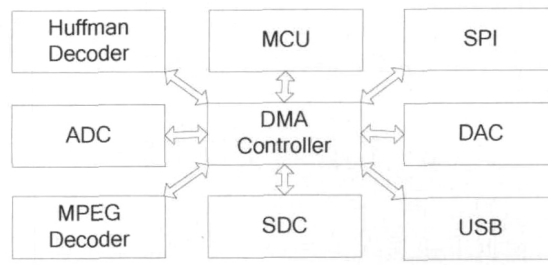

图 1 一种典型多媒体应用系统

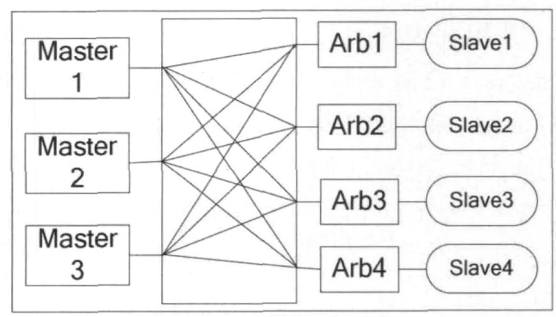

图 2 3×4 full crossbar topology

Crossbar 的互联结构实现在 MDMAC 的接口模块中,作为 DMA 传输的局部总线,DMA 传输不再占用系统总线,因此可以和 CPU 并行工作。所有需要采用 DMA 方式传输数据的设备都直接挂接在该接口模块上,而由于这些设备也需要同系统总线通信，所以每个设备需要附加一个多路选择单元，以选择来自系统总线的信号或者是来自 MDMAC 的信号。MDMAC 的接口模块实现了全 Crossbar 的互联结

构，其和设备的连接及系统总线的连接方式如图 3 所示，处于图中左边的任意设备可以和图中右边的任意设备进行 DMA 方式的数据传输。由于可以根据具体的应用来设计接口模块中设备端口的个数，所以可以保证系统中所有的 DMA 传输都能并行进行，而不需要像传统 DMA 控制器，要通过一定的仲裁算法来对多个传输请求进行排序，使得传输时间不能保证，并且仲裁模块会引入较大的延时，使得传输响应变慢。

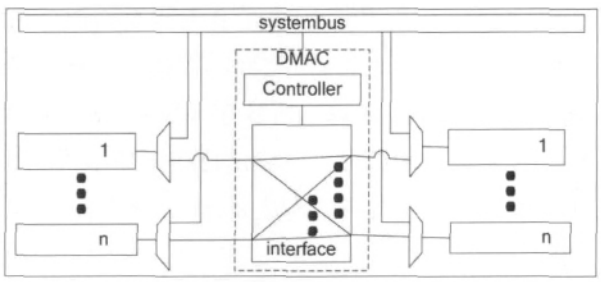

图 3 DMAC 和设备的连接

3 设计与实现方案

根据上述多通道 DMA 控制器的基本原理，可以按如图 4 结构来设计实现该多通道 DMA 控制器，其组成模块可以分为 4 部分：寄存器模块 Registers，控制模块 Controlunit，地址产生模块 Addressgen 和接口模块 Interface。下面详细介绍各模块的具体设计与实现方案。

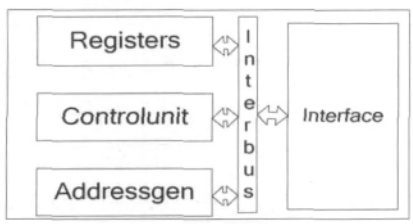

图 4 DMAC 原理框图

3.1 寄存器模块

Registers 寄存器模块集合了控制器的各种寄存器，包括全局寄存器和各个通道的专用寄存器，控制器和系统微处理器的交互主要靠这些寄存器来完成。全局寄存器对所有的通道都有效，包括配置寄存器，状态寄存器，中断屏蔽寄存器，中断源寄存器，通道的专用寄存器包括控制寄存器，传输量寄存器，源地址寄存器，源地址屏蔽寄存器，目的地址寄存器，目的地址屏蔽寄存器，配置描述符指针寄存器。在传输建立阶段，需往全局寄存器和相应通道的专用寄存器里写入必要的传输控制信息，如传输的源地址，目的地址，传输大小，传输模式，配置模式，中断允许和传输使能等，在传输过程中或传输结束时各种状态将会反映在状态寄存器中，供 CPU 读取以确定下一步的操作。在传输临时中止或者传输结束后源地址和目的地址的当前值将会返回源地址和目的地址寄存器中，这样在下一次传输时若是接着上一次的地址空间操作，就不需要再去重新设置地址寄存器，这对于在一个固定地址区间里连续而循环的操作是很方便的。

3.2 控制模块

Controlunit 控制模块主要产生传输的开始、结束、寄存器更新和其它控制信号，控制状态机的转换等，状态转换图如图 5 所示。从图中可以看出，MDMAC 可以用 CPU 通过总线来配置，传输开始后由 Idle 状态直接进入 read 状态。此外，它还可以通过读取存储在存储器上的配置描述符来进行自我配置，Idle 状态的下几个状态是读取描述符状态，读取完后再进入 read 状态。在此种模式下，需要在 pointer 寄存器中指定第一组描述符的起始地址。一个完整的描述符链表可以包含多组描述符，一组描述符可以完整地配置一个传输任务，包括源地址、目的地址、传输大小和传输模式等。每组描述符都包含下一组描述符的起始地址，地址可以是连续的也可以是不连续的。每组描述符都有一个状态位表示该组描述符是否是最后一组描述符。若最后一组描述符指向第一组描述符，并且其状态位没有指示其是最后一组，即 MDMAC 将会循环地执行这些传输任务，直到 CPU 发来强制中止信号。描述符配置方式能够减少中断 CPU 的次数，因为传输完成一个任务后，下一个任务的传输要求可以由 MDMAC 自行读

取，不需要 CPU 的参与，直到整个描述符链表执行完再产生中断通知 CPU，而传统的 CPU 配置模式每执行完一个任务都要产生中断，让 CPU 对下一个任务进行配置。每次传输的发起可以有两种模式，一种是软件使能方式，即只要将寄存器的 start 位置 1 就开始脱离 Idle 状态；另一种是硬件握手方式，寄存器的 start 位置 1 后，还需要有传输请求的模块发来申请并等待 MDMAC 确认后传输才开始。软件使能方式适用于数据复制操作，CPU 将某一地址区间的数据转移到另一地址区间。硬件握手方式是传统的 DMA 工作方式。

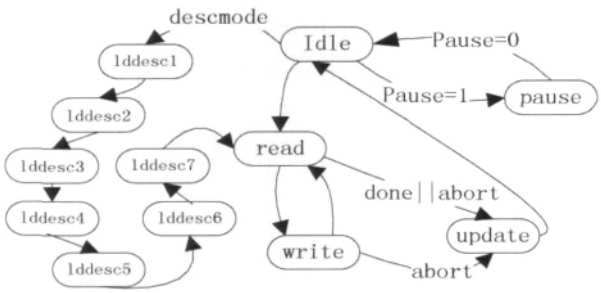

图 5 控制单元状态转换图

3.3 地址产生模块

Addressgen 地址产生模块主要由加法器组成，产生传输的源地址和目的地址，加法器的设计采用两级流水线技术，以减少运算的延时。源地址和目的地址的控制逻辑是独立的，但它们都受传输量的约束，当传输完成时它们都保持着当前地址，直到被重新赋值或复位。源地址和目的地址可以配置为两种变化模式。一种是逐步加 1 模式，由最初设定的起始操作地址开始，每对外读取或者写入一个地址单元后输出地址加 1，适合于对连续地址空间搬移数据。另一种是保持不变模式，即在传输过程中输出地址一直保持最初设定的起始操作地址，此种模式适用于对象是缓存器 FIFO 的情况，因为 FIFO 只有一个入口寄存器，通过对这个入口寄存器操作就可以访问 FIFO 内部的数据。输出地址的各个位都设置了屏蔽功能，这样在对某个连续地址区间写入数据时，可以通过屏蔽某些地址位来保护某些地址单元的值不被覆盖。

3.4 接口模块

Interface 接口模块内部结构框图如图 6 所示，实现接口的 Crossbar 互联结构，向外界输出多路源和目的设备的地址、数据和读写控制等信号，这些信号由图中的 Inputsignals 部分输入。mux 部分根据控制寄存器的配置选通连接的 master 和 slave。本 MDMAC 不同于共享总线型的 DMAC，它可以同时传输几个任务，并且任意设备间都可以进行任务传输，而不是固定某两个模块间才能进行通信。本文实现了一个可以同时处理 3 个通道的控制器，3 个源连接端口和 3 个目的连接端口，默认连接是源端口 0 连接目的端口 0，源端口 1 连接目的端口 1，源端口 2 连接目的端口 2。在使用过程中可以通过寄存器设置任意的源端口连接任意的目的端口，即是一个 3×3 的全 Crossbar 连接。这三个通道是互相独立的，传输任务互不影响，也不必同时开始工作。

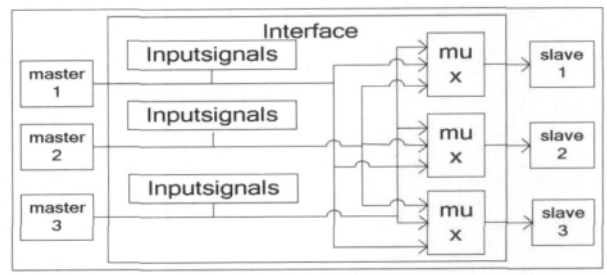

图 6 Interface 模块内部结构

4 实验结果

本文首先在 modelsim6.5 中对设计进行仿真，3 个源端口和 3 个目的端口分别连接着一个存储器 RAM。分别仿真了单个通道操作，多个通道同时操作，多个通道不同时操作，通过系统总线配置，通过描述符配置，软件使能传输，硬件握手传输等模式。本文设计了一个系统在 FPGA 上运行，系统的内部框图如图 7 所示，采用 Xilinx 的 virtex5 开发板。图中的 USB 模块是一个 USB1.1 主机控制器，里面包含一个 64 字节的缓存器 FIFO。USB 主机控制器和存储器 Memory3 连接在 MDMAC 的源端口，存储器 Memory1 和存储器 Memory2 连接在目的端口。USB

主机控制器从 U 盘读取数据到其内部的 FIFO，然后使能 MDMAC 把 FIFO 中的数据搬移到 Memory1 中，与此同时，另一个通道把 Memory2 中的数据复制搬移到 Memory3 中，然后通过串口把搬移后的数据输出到 PC 机上的超级终端显示出来。图 8 上显示的是从 U 盘第 0 扇区读出的内容，由于编程直接使用 keil 软件提供的打印函数，该函数显示的数据是 16 位的，故图中每个数据的后两个数字都是无效的。用 winhex 软件读取 U 盘里面的内容，传输前后的内容一致。

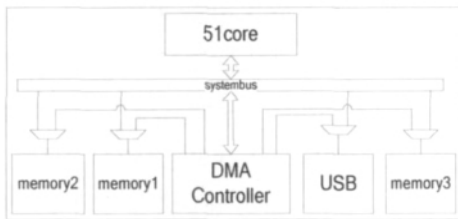

图 7 FPGA 演示系统

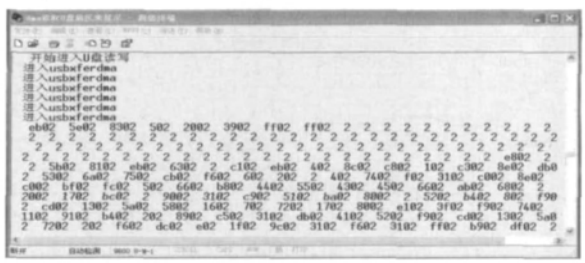

图 8 读取到的 U 盘的内容

图 9 是 3 个通道同时操作的仿真波形图，其中源端口 0 连接目的端口 1，源端口 1 连接目的端口 2，源端口 2 连接目的端口 0，图中的 m 表示源端口，s 表示目的端口。对于传统 DMA，必须等待一个任务传输完毕后才能传输下一个任务，因此不能保证将相关数据实时地传输到目的地址，图 9 显示了本文设计的 MDMAC 能够同步地处理多个任务，因此比传统 DMAC 更能适应实时性系统的要求。

本文设计的 MDMAC 在 Xilinx 的 xc5vlx110t 上实现的最大频率是 232MHZ，每两个时钟周期可以传输一个字节的数据，若 3 个通道同时工作，则最大的数据吞吐率可达 348MB/s，表 1 给出了几种 DMAC 在 FPGA 上实现的最大吞吐率的对比。其中文献[6]得出的结论是采用其提出的链模式对于提

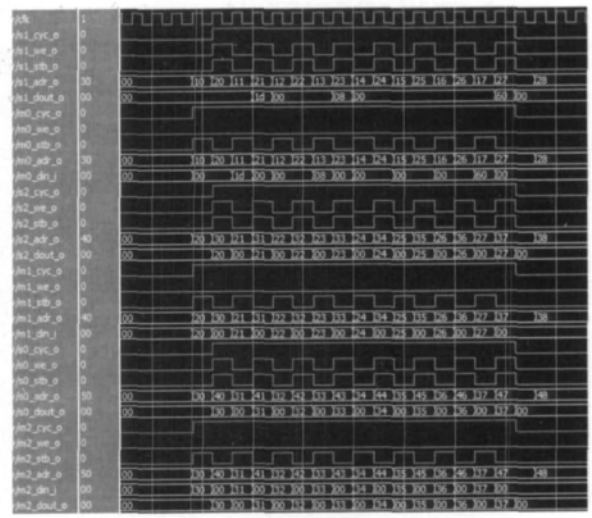

图 9 3 个通道同时传输

高数据吞吐率的贡献在于减少了相连传输任务之间的中断服务时间，本文提出的多通道模式不但在单个通道的实现上采用了类似的优化方式，而且多通道同时工作的模式也包含了类似的优化思想，因为这省去了传统 DMA 机制中多个设备同时发起请求时仲裁排序和切换服务对象的时间，并且，由于多个通道同时工作，整个系统的数据吞吐率得到了很大的提高。

表 1 几种 DMAC 吞吐率的对比

文献	本文	[4]	[6]	[7]
数据吞吐率（MB/s）	348	166	59.5	66

本文采用标准的 180nmCMOS 工艺库，在 Design Compiler 中对 MDMAC 进行了综合，在 300MHZ 的时钟频率下，综合出的面积是 104322.6 μm^2，功耗是 13.1mW。为了进行比较，本文还设计了一个没有 Crossbar 互联接口功能而其他功能完全一样的 DMAC，在同样的约束环境下，在 300MHZ 的时钟频率下综合出的面积是 46037.4 μm^2，功耗是 6.5mW。数据吞吐率的提高有可能是以面积或功耗为代价的，为了说明数据吞吐率的改进效率，本文采用在每单位消耗（面积或功耗）上数据吞吐率的相对大小来衡量，本文定义一个变量 η 为数据吞吐率和面积的比，即每单位面积的平均数据吞吐率，ζ 为数据吞吐率与功耗的比，即每单位功耗的平均数据吞吐率。设传统 DMAC 的

数据吞吐率，功耗和面积都为 1，则 η_t=1，ζ_t=1，其中下标 t 表示传统 DMAC。本 MDMAC 当 3 个通道同时工作时其数据吞吐率达到最大，为 3，面积为 104322.6 /46037.4 = 2.3，则其 η_c = 3/2.266 = 1.3，下标 c 表示本文设计的 MDMAC，功耗为 13.1/6.5 = 2.0，则其 ζ_c = 3/2.0 = 1.5，两种 DMAC 的比较结果如表 2 所示。可见，本 MDMAC 比几个传统 DMAC 协同使用更有效率，这是因为省去了仲裁单元并且某些控制逻辑可以复用。

表 2 Crossbar DMAC 与传统 DMAC 的比较

	吞吐率	面积	功耗	吞吐率/面积	吞吐率/功耗
MDMAC	3	2.3	2.0	1.3	1.5
传统 DMAC	1	1	1	1	1

5 结论

本文给出了一种基于 Crossbar 的多通道 DMA 方案，并用硬件实现了这种 DMA 控制器，它可以同时服务多个通道，接口采用了全 Crossbar 的拓扑结构。所设计的方案在 virtex5 开发板上通过了验证，其最大的数据吞吐率可达 348MB/s，经采用标准 180nm 工艺综合，其吞吐率和面积比是传统 DMA 控制器的 1.3 倍，吞吐率和功耗比达到 1.5 倍。

参考文献

[1] Yong Dou; Lin Deng, Jinhui Xu. DMA Performance Analysis and Multi-core Memory Optimization for SWIM Benchmark on the Cell Processor[A]. Yong Dou. Parallel and Distributed Processing with Applications [C].IEEE CONFERENCE PUBLICATIONS.2008.170-179.

[2] Texas Instrument, TMS32OC6000 Peripherals Reference Guide.

[3] Kuan Jen Lin, Chuang Hsiang Huang, Cheng Chia Lo. Design and Implementation of a Schedulable DMAC on an AMBA-Based SOPC Platform[A]. Kuan Jen Lin . Circuits and Systems[C]. IEEE CONFERENCE PUBLICATIONS.2006.279-282.

[4] Peng Yu, Li Bo, Liu Datong. A High Speed DMA Transaction Method for PCI Express Devices [A]. Peng Yu .Testing and Diagnosis [C]. IEEE CONFERENCE PUBLICATIONS. 2009. 1-4.

[5] Nikola Vujic , Felipe Cabarcas, Gonzalez Tallada, M. DMA++: On the Fly Data Realignment for On-Chip Memories [J]. IEEE TRANSACTIONS ON COMPUTERS, VOL. 61, NO. 2, FEBRUARY 2012.237-250.

[6] Guanghui He, Ningyi Xu, Wei Yu.A Single Receiving Chip for DVB Data Broadcasting System [J]. IEEE Transactions on Consumer Electronics, Vol. 52, No. 3, AUGUST 2006.1084-1091.

[7] Tumeo, A., Monchiero, M., Palermo, G.. Lightweight DMA Management Mechanisms for Multiprocessors on FPGA [A]. Tumeo,A.. Application-Specific Systems, Architectures and Processors[C]. IEEE CONFERENCE PUBLICATIONS.2008.275-280.

[8] Minje Jun, Deumji Woo, Eui-Young Chung. Partial Connection-Aware Topology Synthesis for On-Chip Cascaded Crossbar Network[J]. IEEE TRANSACTIONS ON COMPUTERS, VOL. 61, NO. 1, JANUARY 2012. 73-86.

[9] 曹亚菲,王大伟,李思昆.Crossbar 总线与共享总线相结合的 SoC 系统级通信综合方法研究. 计算机研究与发展. ISSN 1000-1239/CN 11- 1777/TP.2008. 1439-1445.

作者简介

陈伍敏，厦门大学在读硕士，主要研究方向为数字集成电路设计。

刘荣生，福建新大陆自动识别技术有限公司，主要研究方向为自动识别。

罗闵闾，厦门大学在读博士，主要研究方向为嵌入式处理器。

郭东辉，厦门大学教授，主要研究方向为集成电路设计，人工智能。