第35卷 第18期 计 算 机 工 程 2009年9月
Vol.35 No.18 Computer Engineering September 2009
·工程应用技术与实现· 文章编号：1000—3428(2009)18—0234—03 文献标识码：A 中图分类号：TP303

基于SPM的多核SoC访存结构设计与优化

刘 磊，严 明，李思昆
(国防科技大学计算机学院，长沙 410073)

摘 要：针对“一个RISC主处理器核+几个专用协处理器核”结构的计算密集型SoC，设计一种以执行命令方式完成大块数据传输的高效访存结构。通过增加组传输和流水传输模式，对该结构进行优化。实验结果表明，该访存结构设计及优化方案的数据传输效率高、实现开销小，并且对同类SoC系统，该设计具有良好的适用性。

关键词：便签式存储器；多核SoC；访存

Design and Optimization of Memory Access Structure for Multicore-SoC Based on SPM

LIU Lei, YAN Ming, LI Si-kun
(School of Computer Science, National University of Defense Technology, Changsha 410073)

【Abstract】 Aiming at the structure of compute-intensive SoC that “one RISC main processor core + some dedicated co-processor cores”, this paper designs a memory access structure which supports a command-driven chunk data transfer. The structure is optimized with block-transfer and pipeline-transfer support. Experimental results show that the structure after optimization has high efficiency, low cost and well flexibility.

【Key words】 Scratchpad Memory(SPM); multicore-SoC; memory access

1 概述

多核访存是影响多核SoC系统性能的关键因素之一。便签式存储器(Scratchpad Memory, SPM)[1]在面积、功耗、访问效率等方面具有优势[2]。某些多核SoC中已用它来取代Cache，例如Cell处理器中的Local Store[1]。

对于面向嵌入式可视媒体处理的SoC，不仅要求高计算性能、高访存效率、低访存延迟，而且要求低功耗、低成本。此类SoC系统结构多采用“一个RISC主处理器核+几个专用协处理器核”的结构，其协处理器核的片上存储器适宜采用SPM。因此，多协处理器访存结构的设计，即多个协处理器如何访问片上存储器，片上存储器与共享的片外存储器之间怎样交换数据，成为影响系统性能的关键因素之一。对于此类SoC，共享总线结构存在总线竞争加剧导致访存性能变差的问题，而交叉开关结构则因互连节点较多致使面积开销过大[3]。本文针对此类应用，设计了一种基于SPM片上存储的高效访存结构，并对该结构进行了优化。

2 访存结构设计

本文以一个集成了32位RISC主处理器核和2个64位协处理器核的多核SoC为例进行分析。

对于采用SPM作为片上存储器的系统，数据传输与数据预取是由软件负责管理的[4]。但实现快速、高效的访存不仅是软件的工作，而且需要硬件提供有利于大块数据快速传输的支持机制。如果SPM与片外存储器之间每次传输数据都需要协处理器来控制，那么传输大量数据将会频繁地占用协处理器资源，降低协处理器的工作效率。并且，当多个协处理器同时访存时将会加剧竞争冲突，容易导致访存阻塞。

一种解决方法是引入专用数据传输管理单元，由它管理实现SPM与片外存储器之间的数据传输，多核之间以共享存储方式进行通信。该部件称作计算与通信调度器(Computation & Communication Scheduler, CCS)。

2.1 采用CCS的多核访存结构

采用CCS的多核访存结构如图1所示。其中，CPU是32位的主处理器，带有指令Cache和数据Cache；SPU0和SPU1是2个64位的协处理器，分别带有SPM类型的片上指令存储器(Instruction Scratchpad RAM)和片上数据存储器(Data Scratchpad RAM)。

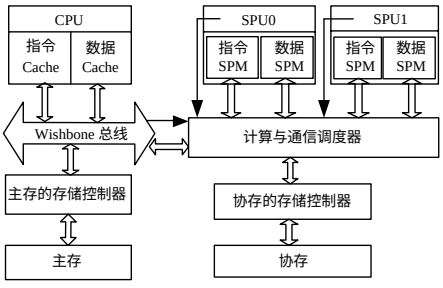

图1 采用CCS的多核访存结构

片外存储器分为2部分：32 bit宽的主存(Main Memory)，用作CPU的存储器；64 bit宽的协存(Synergistic Memory)，用作SPU0和SPU1的存储器。主存和协存都采用Mobile SDRAM存储芯片实现。图1中的空心箭头表示数据通路。

基金项目：国家自然科学基金资助重点项目“半导体重大研究计划”(90707003)
作者简介：刘 磊(1984－)，男，硕士研究生，主研方向：SoC设计方法学；严 明，博士研究生；李思昆，教授、博士生导师
收稿日期：2008-12-24 **E-mail：**lyye03@163.com

各个片上、片外存储器经由 CCS 实现任意两者之间的数据传输。CCS 作为专门负责数据传输的部件，须主动发出访存信号，因此，其数据端口属于主端口类型。CCS 有 3 个命令接收端口，可分别接收来自 CPU, SPU0 和 SPU1 的命令，如图 1 中的黑线箭头所示。

2.1.1　采用 CCS 结构进行访存时的工作过程

CPU, SPU0 和 SPU1 发送命令给 CCS，CCS 执行命令，完成数据在主存、协存和片上存储器任意两者之间的传输。CCS 内有多个通道，每个处理器对应一个或多个通道，处理器发送的命令寄存在通道内。CCS 对各通道的命令进行仲裁，选出优先级最高者作为当前命令，执行该命令，完成一块(chunk)数据的传输。每条命令的内容包括源地址、目标地址、块大小、总传输大小、优先级等。

对协处理器而言，SPM 片上存储器是局部私有的。协处理器访问自己的 SPM 时，通过执行 load/store 指令实现；当需要访问其他处理器 SPM 中的数据时，通过使用 CCS 命令实现。

2.1.2　采用 CCS 结构进行访存的特点

(1)将主存与协存分离，减少了 SPU0 和 SPU1 访存对主处理器系统总线的打扰。

(2)当协处理器的片上数据存储器有多个存储体时可通过交替访问来隐藏访存延迟。

(3)CCS 内部用数据组装与拆分逻辑实现 32 位与 64 位数据域之间的数据传输。

(4)CCS 专门负责数据传输，减少了数据传输对协处理器资源的占用。

2.2　CCS 内部结构

CCS 内部结构主要包括 4 个部分：命令端口，数据端口，通道命令寄存器及通道选择，传输调度引擎。CCS 内部结构如图 2 所示。其中，sp slave 模块从协处理器接收传输命令；wishbone slave 模块从 wishbone 总线上接收来自主处理器的传输命令。接收到的命令寄存在通道模块中。这里设置了 4 个通道：通道 0 和通道 1 由 CPU 配置，通道 2 和通道 3 分别由 SPU0 和 SPU1 配置。通道选择模块从各通道命令中选出优先级最高者作为当前命令。

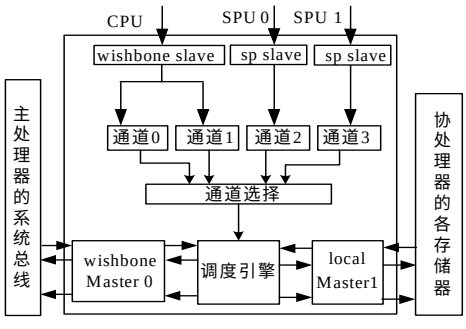

图 2　CCS 内部结构

CCS 通过 Master0 数据端口访问 wishbone 总线上的设备，通过 Master1 数据端口访问协存以及协处理器的片上存储器。调度引擎模块根据当前命令，产生访存操作信号，发送给数据端口。调度引擎模块内部包含有状态机、数据通路、地址生成、数据组装与拆分、读写控制等逻辑部件。

调度引擎以“读后写”的方式工作，首先从源端口读出数据，然后写到目标端口。其状态机共有 6 个状态，迁移过程如图 3 所示，其中，序偶<src, dst>表示数据源和目标端口号，例如<01>表示从 Master0 端口往 Master1 端口传输数据；start 信号表示启动一次传输；done 信号表示一次传输已完成。

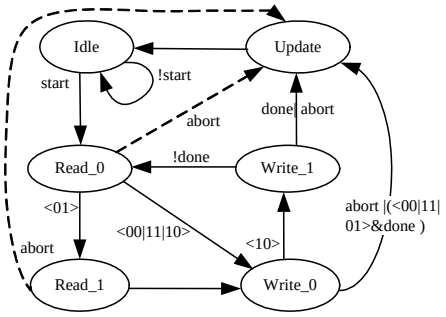

图 3　状态迁移过程

由于 Master0 与 Master1 位宽不同，从 32 bit 端口往 64 bit 端口传输数据时，需要连续 2 次读操作，如图 2 中 Read_0 和 Read_1 状态所示。从 64 bit 端口往 32 bit 端口传输数据时，需要连续 2 次写操作，如 Write_0 和 Write_1 状态所示。在 Update 状态时 CCS 更新通道命令寄存器的值。

2.3　关键逻辑部件

(1)地址生成

对于每次数据块传输，起始源地址和目标地址存放在通道的命令寄存器中；而访问每个数据字时的源地址和目标地址由调度引擎模块内部寄存器提供。传输启动时，把起始源地址和目标地址从命令寄存器加载到调度引擎模块内部的寄存器。传输结束时，用调度引擎模块中的地址值来更新命令寄存器的相应内容。

地址按字节编址，每次读操作后源地址累加 1 次，每次写操作后目的地址累加 1 次。对于 32 位读写，累加值是 4；对于 64 bit 读写，累加值是 8。

(2)数据通路

多个数据端口之间的数据交换由数据通路逻辑实现，它包括 3 个部分：数据寄存器，输入端口选择和输出端口选择。调度引擎模块根据当前命令选出源和目标，从源端口输入的数据被寄存到数据寄存器，然后从数据寄存器转发到目标端口。

3　传输模式的优化

3.1　传输时序分析

协存的存储控制器与协处理器的片上存储器都连接在 Master1 端口。当两者之间传输数据时，以先读后写方式进行。以数据从协存传输到片上存储器为例，CCS 工作时序分析如下(假设块大小是 *chunk_size* 个字)：

(1)CCS 通过 Master1 端口从协存存储控制器读入一个数据字到内部数据寄存器。此处是读 SDRAM 操作，占用 6 个时钟节拍，其中，5 拍为潜伏期；1 拍为采样数据。

(2)CCS 通过 Master1 端口把数据写到片上存储器。此处是写 SRAM 操作，占用 1 个时钟节拍。

(3)判断是否读了 *chunk_size* 次，如果不是，则跳到第(1)步继续；如果是，则本次传输完成。

在上述过程中，每传输 1 个数据字，需要 7 个时钟节拍，其中 5 拍处于潜伏期，传输延迟较长，效率较低。

SDRAM 芯片的特性决定了潜伏期是必需的。为提高访存效率，SDRAM 芯片提供了 burst 读写模式。在 burst 模式中，过了潜伏期后一旦数据有效便能以每拍一个字的速率连续读/写 *burst length* 个字。为利用 SDRAM 的 burst 模式，需

要做 CCS 优化，以使数据端口支持 burst 读写。

3.2 组传输优化方案

对于上文所述协存与片上存储器之间的数据传输，数据源和目标连接在同一个端口上，只能以先读后写的方式进行。在这种情况下，对 busrt 读写的支持需要用存储转发方式实现，即组(Block)传输。在 burst 读写时，连续读入/写出的 *burst length* 个数据字临时存放在 CCS 内部。

若要支持组传输，需要对 CCS 内部的数据通路逻辑、状态机、通道命令寄存器等处做修改。

CCS 数据通路逻辑中需要增加数据寄存器，以构成数据寄存器队列，用于寄存 burst 读写时连续读进/写出的数据。burst length 的取值决定了数据寄存器队列的深度。

对状态机的修改有 2 种方法：(1)引入计数判断逻辑，每次读/写到一个字，累加计数器，判断是否已读/写到 *burst length* 个字。(2)为 burst 中每次字访问增加读状态与写状态。

组传输优化方案所需修改比较简单，但缺点是组的大小不可配置，并且组大小的选择对硬件开销影响很大。这是对同一个数据端口分时复用，采用存储转发方式很难避免的。如果数据在 2 个端口之间传输，则可以同时进行读写，避免分时复用的问题。

3.3 流水传输优化方案

当数据源和目标连接在不同端口时，数据传输以边读边写的方式进行。这时，对 burst 读写的支持可以用一端进一端出的流水方式实现，即流水传输。在 burst 读写时，连续的数据字从一个端口读入，另一个端口写出，2 个端口同时操作，无须把数据临时寄存在 CCS 内部。

增加对流水传输的支持，需要对 CCS 内部的数据通路逻辑、状态机、通道命令寄存器等处进行修改。对数据通路的修改主要是对输入端口选择逻辑和输出端口选择逻辑进行扩展。在状态机中须增加一个状态以表示 2 个端口同时操作。并且产生读写控制信号的逻辑单元也应做调整。

流水传输的传输效率比较高，传输时的连续数据字个数可由软件配置。但是 CCS 的控制变得复杂，并且增加数据端口会增加相应的硬件开销。

4 实验结果分析

本文的设计用 Verilog 代码实现，在 Modelsim SE6.1f 中进行了模拟，使用 Xilinx ISE9.1 工具综合，下载到 Xilinx Virtex4-XC4VLX160 开发板上通过了验证。

4.1 性能分析

以传输数据所花费的时钟节拍数作为衡量传输效率的计算指标。假设数据块大小为 *chunk_size*，组的大小为 *block_size*，且 *chunk_size* 是 *block_size* 的整数倍。

(1)当在片上存储器之间传输数据时，由于访存没有潜伏期，因此优化前后的传输效率相同。

(2)当从协存往片上存储器传输数据时，完成一个数据块传输需要的时钟节拍数如下：

1)优化前：

T = (5 潜伏 + 1 读 + 1 写)×*chunk_size*

2)组传输：

tb = (5 潜伏 + *block_size* 读 + *block_size* 写)

T_b = (*chunk_size*/*block_size*)×*tb*

3)流水传输：

T_p= 5 潜伏 + *chunk_size* 读写

对 *chunk_size* 和 *block_size* 取不同的值，各种情况下的时钟节拍数如表 1 所示。其中，T_{b4} 表示 *block_size* 取 4；T_{b8} 表示 *block_size* 取 8；T_{b16} 表示 *block_size* 取 16。

表 1 各种情况下的时钟节拍数

Chunk_size/字	T	T_p	T_{b4}	T_{b8}	T_{b16}
16	112	21	52	42	37
32	224	37	104	84	74
64	448	69	208	168	148
128	896	133	416	336	296

可以看出，优化后传输所用时钟节拍数减少很多。以 *chunk_size* 取 32 为例，流水传输的节拍数只占优化前的 16.51%，组传输的节拍数只占优化前的 46.43%~33.04%。*block_size* 的取值在 4~8 时效率提高 19.23%，在 8~16 时效率提高了 11.90%。随着 *block_size* 逐渐增大，效率提高逐渐变慢。*chunk_size* 取值越大传输效率越高，但是占用 CCS 时间也越长，容易出现多通道命令之间调度不均衡的现象。

(3)当从片上存储器往协存传输数据时，情况与(2)相似，其潜伏期为 2，优化后传输效率的提升比(2)中略低。

4.2 开销分析

优化前后的硬件开销对比如表 2 所示。

表 2 优化前后的硬件开销对比

部件名称	等效逻辑门数	增加开销/(%)
CCS(优化前)	5 358	0.00
CCS(组传输,每组 8 个字)	6 196	15.64
CCS(流水传输)	7 017	30.96

可以看出，CCS 结构的硬件开销比较小。相对于优化后所提升的传输效率，开销增加较少。

4.3 可扩展性分析

在本文的设计方案中，增减协处理器时，对 CCS 的修改主要是增减相应的命令端口。增减数据端口时，主要是修改数据通路逻辑。此外，片上存储器的容量和存储器个数同样易于扩展。因此，对基于 SPM 的多核 SoC，当处理器核数量不多时，本文设计方案具有良好的适用性。

5 结束语

本文设计了一种通过执行命令来完成大块数据传输的高效访存结构。该设计采用计算与通信调度器管理数据传输，克服了多核访存无序竞争导致冲突加剧的缺点，减少了访存操作对协处理器的占用，提高了访存效率。分离主存与协存的设计，与共享总线结构相比，减少了总线占用。通过增加组传输和流水传输模式，对传输模式进行优化，提高了传输效率。

参考文献

[1] Wikipedia. Scratchpad RAM[Z]. (2008-07-05). http://en.wikipedia.org/wiki/Scratchpad_RAM.

[2] Banakar R, Steinke S, Lee B, et al. Scratchpad Memory: A Design Alternative for Cache On-chip Memory in Embedded Systems[C]// Proc. of the 10th International Symposium on Hardware/Software Codesign. Estes Park, Colorado, USA: ACM Press, 2002.

[3] 高 翔. 多核处理器的访存模拟与优化技术研究[D]. 合肥: 中国科技大学, 2007.

[4] Ozturk O, Kandemir M, Kolcu I. Shared Scratchpad Memory Space Management[C]//Proc. of the 7th International Symposium on Quality Electronic Design. Washington D. C., USA: IEEE Computer Society, 2006.

编辑 顾姣健