RISC 微处理器中 I/O 子系统设计的一种优化方法*

冉计全，樊晓桠，孙华锦
(西北工业大学 航空微电子中心，陕西 西安 710072)

摘　要: 低效率的访存操作是限制微处理器性能提高的一个关键因素。提出了 I/O 子系统(IOSS)设计中一种优化的模型,阐述了该模型提高访存效率的机制,分析了这种模型协调微处理器与存储器之间速度差异的作用。Verilog 仿真、综合和静态时序分析的结果表明该设计达到了预定的要求。目前龙腾Ⅱ微处理器已经进入后端流程,不久将使用 0.18 μm 的工艺进行流片。

关键词: 微处理器; I/O 子系统; FIFO; IOSS
中图法分类号: TP303　　**文献标识码**: A　　**文章编号**: 1001-3695(2005)05-0045-03

An Optimal Approach in Design of Input/Output Subsystem of RISC Microprocessor

RAN Ji-quan, FAN Xiao-ya, SUN Hua-jin
(Aviation Microelectronics Center, Northwestern Polytechnical University, Xi'an Shanxi 710072, China)

Abstract: Low-efficiency memory access is an important factor which limits the high performance of microprocessors. This paper presents an optimal design model in design of input/output subsystem of RISC microprocessor. This paper also introduces the mechanism of improving the efficiency of IOSS accessing memory, analyses the function in balancing the speed difference of microprocessor and memory. The result of simulation, synthesis and static- time analysis show the design is successful. The Longten Ⅱ is now in back-end process and will tape out on 0.18 μm CMOS technology sooner.

Key words: Microprocessor; I/O Subsystem; FIFO; IOSS

输入/输出子系统始终是高性能计算机系统中的瓶颈[1]。当然随着 IC 制造工艺和 RISC 技术的发展,电路速度越来越快,指令执行时间越来越短,虽然连接微处理器(MPU)和存储器的 I/O 的带宽也不断增加,但它的增长速度远远赶不上微处理器和存储器本身速度的增长[2]。特别是先进的 RISC 技术,如超流水、超标量和 VLIW 在 MPU 设计中的广泛应用,使得这一矛盾更加突出。为了解决这一问题,常用的方法有:①优化 Cache 的设计。通过采用多级 Cache、牺牲 Cache,以及缺失下缺失和缺失下命中技术[2]等,减少了缺失率和缺失代价,从而满足了快速执行指令的要求[3]。②分支预测技术[4]。它目标是根据某些条件提前判断是否需要转移,若需要,则将预测的目标指令和数据预取到 Cache,从而避免转移引起的 Cache 失效,损失系统性能。这种技术能够从一定的程度上提高系统的性能。③存取式结构和大寄存器堆[5]。MPU 采用存取式结构,即在 MPU 内部设置大量的寄存器,为 Load/Store 指令以外的其他非访存指令提供广阔的运算空间,这样减少了 MPU 访存次数,从某种意义上来说也就提高了访存效率。

本文从另一个角度出发,提出了微处理器设计中输入/输出子系统的一种优化方法,它能有效地提高 MPU 的访存效率,并已成功地应用到了龙腾Ⅱ微处理器中。

收稿日期: 2004-05-31; 修返日期: 2004-07-15
基金项目: "十五"国防预研课题资助项目(41308010108)

1　I/O 子系统缓冲模型

I/O 子系统(IOSS)是内部 MPU 和外部总线数据传输的桥梁,因此为了减少 MPU 访存操作的等待时间,我们设想在 IOSS 中加入由三个 FIFO 组成的缓冲区,包括指令 FIFO、Store 数据 FIFO 和 Load 数据 FIFO。指令 FIFO 可以存放一定数量的访存操作,这样 MPU 只要将访存指令写到指令 FIFO 中,MPU 就可以执行其他的指令,从而避免了 MPU 因等待存取数据或取指令的等待时间。因而在高速 MPU 和低速存储器之间架起的这座桥梁使双方都能够按照自己原有的速度工作。

1.1　I/O 子系统缓冲模型的结构

I/O 子系统的缓冲模型结构如图 1 所示。

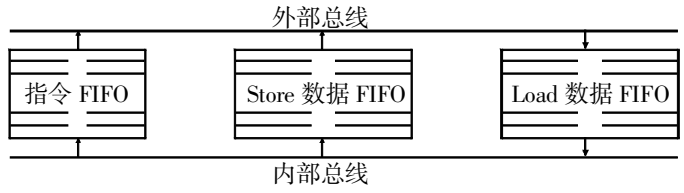


图 1　IOSS 缓冲模型

IOSS 主要由指令 FIFO、Store 数据 FIFO 和 Load 数据 FIFO 三部分组成。指令 FIFO 可以同时存储 n 个访存操作,即由 n 个单元组成,每个单元由标志位、访存地址和访存属性三部分组成。其中标志位用于指示该单元是否被占用;访存地址指示访存操作的内存地址;访存属性表示访存操作的属性信息。Store 数据 FIFO 可以同时存储 n 个访存的存储数据,位宽为

256 位, 主要用于支持 64 位数据总线的突发操作。Load 数据 FIFO 可以同时存储 m 个从内存中取回的加载数据或指令, 由 m 个单元组成, 每个单元由标志位、地址位和数据位三部分组成。数据位存放从内存中取回的数据。此结构支持三级流水操作, 流水级可分为三部分, 即取指级、地址总线级和数据总线级。

1.2 I/O 子系统缓冲模型数据通路描述

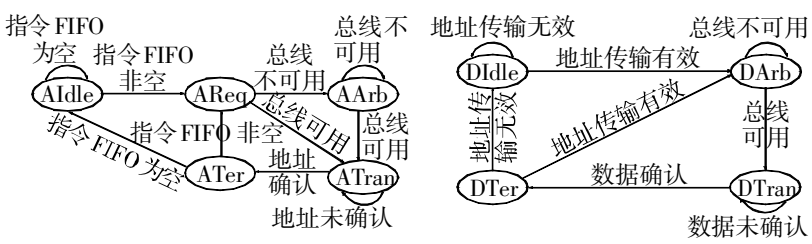
当内部 MPU 需要进行访存操作时(假定为指令 x) , 只需要将访存操作的相关信息以指令 FIFO 的数据格式写到指令 FIFO 中指令队列的尾端, 流水线就可以继续执行后续指令, 从而减少了 MPU 因外部访存长时间等待而使流水线断流的几率, 这在一定的程度上提高了系统的性能。对于 Store 指令, 在写指令 FIFO 的同时将要存储的数据写到 Store 数据 FIFO 中数据队列的尾端。

当指令 x 位于指令队列的首位时, 启动总线控制状态机进行外部总线的请求。在总线授权的情况下, 若为 Store 指令时, 则分别从指令 FIFO 和 Store 数据 FIFO 中取出地址、地址属性信号, 以及要存储的数据送到外部总线上去, 当外部确认信号有效时, Store 指令执行完毕。若为 Load 指令, 则将从指令 FIFO 中取出地址和地址属性信号送到外部总线上, 在外部确认信号有效时, 从外部数据总线取回数据或指令暂存在 Load 数据 FIFO 中, 此时指令 x 还没有真正的完成, 还需要在 FIFO 排队等待送往内部 MPU, 在取回的数据或指令被送到内部 MPU 后, Load 指令才被真正的执行完毕。

1.3 I/O 子系统缓冲模型控制通路的描述

I/O 子系统控制通路由地址状态机和数据状态机组成。当地址状态机检测到指令 FIFO 非空时, 表示指令 FIFO 中有访存指令等待排队, 此时启动地址状态机和数据状态机进行外部总线的请求。在外部仲裁器授权的情况下, 控制缓冲模型执行相应的操作。

地址状态机由五个状态组成: AIdle 空闲状态、AReq 地址总线请求状态、AArb 地址总线仲裁状态、ATran 地址总线传输状态和 ATer 地址总线终止状态; 数据状态机由四个状态组成: DIdle 空闲状态、DArb 数据总线仲裁状态、DTran 数据总线传输状态和 DTer 数据总线终止状态, 如图 2 所示。



(a)地址状态机状态迁移图 (b)数据状态机状态迁移图

图 2 IOSS 控制状态机状态迁移图

控制通路的操作可以分为如下几步:

(1) 当指令 FIFO 为空时, 地址和数据状态机均停留在 Idle 状态; 若指令 FIFO 非空, 表示有新的访存请求, 此时地址状态机进入 AReq 状态。

(2) 当外部授权地址总线使用时, 地址状态机进入 ATran 状态, 同时地址状态机启动数据状态机进入 DArb 状态。

(3) 当外部授权数据总线使用时, 数据状态机进入 DTran 状态; 对于 Store 指令将 Store 数据 FIFO 的数据送到数据总线上; 对于 Load 指令, 则从数据总线上取回加载的数据或指令, 存储到 Load 数据 FIFO 中去。

(4) 当外部地址确认信号有效时, 地址状态机转到 ATer 状态。

(5) 当外部数据确认信号有效时, 数据状态机转到 DTer 状态。

(6) 当指令 FIFO 继续非空, 则转到(2) ; 若指令 FIFO 为空时, 地址状态机转到 AIdle 状态, 数据状态机转到 DIdle 状态。

如上所述, 取指令、传地址、传数据是同时进行的, 在数据总线进行数据传输时, 可以同时启动地址状态机, 同样, 在地址总线进行地址及其访存属性信号传输时, 可以从指令 FIFO 中取出数据, 这样就很好地实现了访存操作流水线的设计, 提高了 MPU 访存的效率。

在这里有一点需要说明的是: 当指令 FIFO 为满时, 必须阻塞内部 MPU 继续向 I/O 子系统发访存请求; 当 Load 数据 FIFO 为满时, 若指令 FIFO 队列的队首为 Load 指令, 则 Load 数据 FIFO 必须阻塞指令 FIFO 进行新的总线请求操作。

2 I/O 子系统缓冲模型在龙腾Ⅱ中的实现

龙腾Ⅱ微处理器是西北工业大学“十五”预研项目中自主开发的新一代高性能的机载微处理器。它在指令系统级上与 PowerPC[6] 的体系结构相兼容。

龙腾Ⅱ微处理器实现了 117 条定点指令和 69 条浮点指令, 可以进行 8, 16, 32 位整数和 32, 64 位浮点运算。其体系结构包含八个功能部件: 取指部件、指令译码部件、定点部件、浮点部件、存取部件、存储管理部件、高速缓存部件和 I/O 子系统, 如图 3 所示。

I/O 子系统(IOSS) 为微处理器内核与处理器外部总线提供了一个数据交换通道, 由指令预处理逻辑、输出 FIFO(包括指令 FIFO、Store 数据 FIFO) 、输入 FIFO(Load 数据 FIFO) 和数据后处理逻辑四部分组成。主要完成 Load/Store、取指和预取指操作, 同时进行地址、数据的校验检查和总线监听, 实现了外部总线的流水、乱序执行和总线重传操作。既提高了总线接口的效率, 又使得龙腾Ⅱ可以应用于多处理机系统。

根据龙腾Ⅱ体系结构的定义, I/O 子系统可以从数据 Cache、指令 Cache、DMMU 和 IMMU 同时接收访存请求。其中数据 Cache 有取数和预取数、指令 Cache 有取指和预取指操作, 但同时也考虑到芯片的面积不予过大, 由此我们设计指令 FIFO 的深度为 6。与此同时, 通过对机载计算机应用程序的研究发现, MPU 不会连续发送三个 Store 或 Load 访存请求, 故选择 Store, Load 数据 FIFO 的深度为 4 较宜。

该设计中, 出于对芯片设计面积和成本的考虑, 我们把输出 FIFO 拆分为指令 FIFO 和 Store 数据 FIFO, 这是因为: 若在指令 FIFO 的单元中加入数据表项, 则对于 Load 指令和指令 Cache 的取指和预取指操作而言, 其数据区为空, 这样既损失了芯片的设计面积同时也不符合低功耗的设计要求。

3 FIFO 的实现

在 IC 设计中, 对于存储体的实现, 国内外普遍采用的方法

是, 采用芯片后端生产厂商提供的同步或异步的双端口(有的设计用三端口甚至多端口) RAM 模型, 按照其设计要求, 将所采用的 RAM 模型内嵌到设计单元中。因此在本设计中, 出于对设计周期和成本因素的考虑, 也将采用该设计方法。

由于 I/O 子系统中包含内部系统时钟和外部总线时钟, 因此如何匹配内部和外部时钟的多时钟域的问题也成为了本设计的一个难点。在此, 我们选择了异步 FIFO 作为总线接口的缓冲模型。其结构包括读、写地址产生逻辑, 空满标志产生逻辑和双端口 RAM, 如图 4 所示。

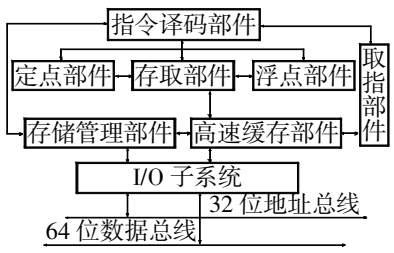


图 3　龙腾Ⅱ微处理器体系结构框图

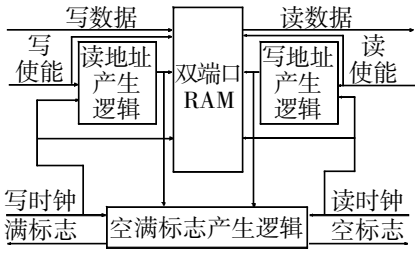


图 4　异步 FIFO 逻辑框图

整个 FIFO 分为两个完全独立的时钟域——读时钟域和写时钟域。FIFO 的存储介质为一块双端口 RAM, 可以同时进行读写操作。在写时钟域部分, 由写地址产生逻辑产生写控制信号和写地址; 读时钟部分由读地址产生逻辑产生读控制信号和读地址。在空/满标志产生部分, 由读写地址相互比较产生空/满标志。

在设计异步 FIFO 时, 需要解决亚稳态的问题。在本设计中, 我们采用如下两种措施来解决异步 FIFO 的亚稳态问题:

(1) 对写地址/读地址采用格雷码。对多个触发器的输出所组成的写地址/读地址可以采用格雷码。由于格雷码每次只变化一位, 采用格雷码可以有效地减少亚稳态的产生。

(2) 采用触发器来同步异步输入的信号, 这样比较好地减少亚稳态出现的几率, 但同时这种方法会给输入信号带来一级延时。

4　结论

I/O 子系统的缓冲模型的设计方案已在 Synopsys 工具的支持下用 VerilogHDL 语言实现了 RTL 级电路设计并成功地用 VCS 通过了仿真。通过对该模型的仿真发现, 该模型很好地缓解了高速 MPU 和低速外部存储器之间的速度差异, 提高了整个微处理器的访存效率。目前该处理器已经进入后端流程, 不久将使用 0.18μm 的工艺进行流片。实验证明, I/O 子系统缓冲模型在任何微处理器设计中都是一种很好的设计方法, 具体应用时只要根据具体的要求进行相应的修改来选择最优的设计方案。

参考文献:

[1] 孙华锦, 高德远, 樊晓桠, 等 . 32 位微处理器总线接口部件的设计[J] . 西北工业大学学报, 2003, 21 .

[2] 马婉良, 高德远, 张盛兵. 微处理器设计中提高访存效率的一种方法[J] . 西北工业大学学报, 1999, 17(3) : 338-343.

[3] M Hennessy, Patterson. Computer Architecture: A Quantitative Approach (3Edition) [M] . Tsinghua University Press, 2001.

[4] J Yeh, *et al*. Increasing the Instruction Fetch Rate via Multiple Branch Prediction and aBranch Address Cache[C] . Tokyo: Conference Proceedings '93 International Conference on Supercomputing, 1993 . 67-76.

[5] Motorola. PowerPC Microprocessor Family: The Programming Environments for 32-Bit Microprocessors[R] . Motorola Inc. , 1997.

[6] Motorola. MPC750 RISC Microprocessor Family Users Manual [R] . Motorola Inc. , 2001.

作者简介:

冉计全(1979-) , 硕士研究生, 主要从事计算机体系结构、微处理器设计等方面研究; 樊晓桠(1962-) , 教授, 博士生导师, 主要从事计算机体系结构和 VLSI 系统设计研究工作; 孙华锦(1975-) , 博士研究生, 主要从事计算机体系结构、微处理器设计等方面研究。

(上接第 36 页) 合、强扩展、适用于复杂协同工作环境的业务集成模型。模型实施中最大限度重用原有应用, 使得原有系统软件仅作简单包装即可平滑纳入新的业务集成系统。目前, 我们的业务集成系统已经成功部署, 但仍然存在不足。下一步的主要工作将集中在提高系统响应速度, 优化企业逻辑库设计等方面。

参考文献:

[1] illiam A Ruh, *et al*. Enterprise Application Integration[M] . New York: John Wiley & Sons, Inc. , 2001. 30-41.

[2] Graham S, *et al*. Building Web Services with Java: Making Sense of XML, SOAP, WSDL, AND UDDI[M] . Sams, 2001.

[3] Kreger H. Web Services Conceptual Architecture[R] . IBM Technical Report WCSA 1. 0, 2001.

[4] 黄双喜, 范玉顺, 等 . 基于 Web 服务的企业应用集成[J] . 计算机集成制造系统—— CIMS, 2003, 9(10) : 864-867.

[5] W3C. SOAP version 1. 2: Messaging Framework [EB/OL] . http: //www. w3. org/TR/soap12-part1 /, 2003-06-24.

[6] W3C. Web Services Description Language (WSDL) version 1. 2: Core Language[EB/OL] . http: //www. w3. org/TR/2003/WD-wsdl12-20030611/, 2003-06-11.

[7] CAI Xiaolu. Whitebook of UDDI[EB/OL] . http: //www. uddi-china. org, 2001-11-04.

[8] Debenham J. Who Does What in a Multi-Agent System for Emergent Process Management, Engineering of Computer-based System[C] . Proc. of 9th Annual IEEE International Conference and Workshop, IEEE, 2002. 35-40.

[9] Maurizio Lenzerini. Data Integration: A Theoretical Perspective [J] . ACM PODS, 2002, (6) : 3-6.

[10] 李慧芳, 范玉顺 . 工作流系统时间管理[J] . 软件学报, 2002, 13 (8) : 1552-1558.

[11] M Muhlen. Evaluation of Workflow Management Systems Using Meta Models[C] . Proceedings of the 32nd Hawaii International Conference on System Sciences, 1999. 121-131.

[12] 尹建伟. 基于 Web 构架智能分布式柔性工作流管理系统——WideFlow 研究及实现[D] . 杭州: 浙江大学计算机系, 2001.

[13] Bernstein A, *et al*. The Process Recombinator: A Tool for Generating New Business Process Ideas[C] . International Conference on Information Systems, 1999.

[14] Curran T, *et al*. SAP R/3 Business Blueprint[M] . Prentice Hall, 2000.

作者简介:

张涛(1979-) , 男, 山东泰安人, 助理工程师, 硕士, 主要从事工作流、网络中间体、信息集成等方面的研究; 尹建伟(1974-) , 男, 江苏徐州人, 副教授, 博士, 主要从事网络中间件、CSCW、信息集成等方面的研究; 陈刚(1973-) , 男, 浙江宁波人, 教授, 博士, 主要从事 CIMS、网络安全、工程数据库等方面的研究; 董金祥(1945-) , 男, 浙江温州人, 教授, 博士生导师, 主要从事 CIMS、操作系统、工程数据库等方面的研究。