第34卷 第18期 计 算 机 工 程 2008年9月
Vol.34 No.18 Computer Engineering September 2008
·开发研究与设计技术· 文章编号：1000—3428(2008)18—0263—03 文献标识码：A 中图分类号：TP334.4

面向流应用的流寄存器文件

马驰远[1]，陈海燕[2]，齐树波[1]，陈书明[2]，肖 嵘[3]
(1. 国防科技大学并行与分布处理国家重点实验室，长沙 410073；2. 国防科技大学计算机学院，长沙 410073；
3. 廊坊广播电视大学，廊坊 065000)

摘 要：存储系统是通用处理器在处理流应用时的瓶颈。该文基于FT64流处理器体系结构，提出一种面向流应用的流寄存器文件结构设计方法和数据传输机制，分析它在FT64中的作用。通过采用大容量、高带宽、虚拟多端口的存储器，将大部分流数据存取操作限制在寄存器文件这一层次，减少了主存压力。实验结果表明，该结构能很好地适应流应用需求。

关键词：流寄存器文件；流处理器；FT64处理器；计算核心；存储

Stream Register File Oriented to Stream Application

MA Chi-yuan[1], CHEN Hai-yan[2], QI Shu-bo[1], CHEN Shu-ming[2], XIAO Rong[3]
(1. National Key Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha 410073;
2. School of Computer, National University of Defense Technology, Changsha 410073;
3. Langfang University of Broadcasting and Television, Langfang 065000)

【Abstract】 The memory system is the performance bottleneck while general processor handles stream applications. Based on the FT64 stream architecture, this paper presents a structure design method and data transfer mechanism of the stream register file which is oriented to stream application, and analyzes the effect on FT64. This structure with the adoption of big capacity, high bandwidth and multiple virtual ports can limit most of the stream data access operation at the register file level, greatly decrease the pressure of the main memory. Experimental result shows this structure meets the demand of stream applications.

【Key words】 stream register file; stream processor; FT64 processor; computing kernel; memory

1 概述

流应用正成为微处理器的主要负载，而通用微处理器体系结构不能适应流应用的要求。它们把大量的芯片面积用于隐藏存储延时，效果并不理想。因此，近年来出现了许多针对流应用的流体系结构[1-2]，它们在信号处理、多媒体及科学计算领域展现出明显优势。

典型流应用的数据访问有以下特点：

(1)大部分数据元素的操作比较独立，元素间的关联小，数据并行性好；

(2)数据重用少，片外存储器中的数据只被读入一次；

(3)流应用不同阶段产生的流数据具有生产者/消费者局部性，即前一阶段产生的计算结果会在下一阶段用到；

(4)计算密集，每组片外读入的数据会被执行100~200个算术操作。

处理流应用的主要思想是将应用组织成数据流与核心程序，利用数据局部性和计算访问的并行性来隐藏存储访问延时，提高整体性能。流体系结构设计的关键是在提供强大计算性能的同时，最大限度地隐藏存储访问延时。

为了隐藏存储访问延时，不同的流处理器具有不同的存储结构。VIRAM将DRAM放入片内，Imagine和Merrimac采用多级存储结构。本文设计了一款64 bit可编程流处理器FT64(Fei Teng 64)，其设计目的是开发信号处理、多媒体及科学计算领域应用程序中的并行性和局部性。它继承了其他流处理器的特点，比如，流编程模型和多级存储结构。FT64采用簇内局部寄存器文件(Local Register File, LRF)、4个运算簇共享的流寄存器文件(Stream Register File, SRF)、cache和DRAM多级存储层次来隐藏访问延时。SRF是FT64处理器核心片上的存储器，是针对流应用而特别设计的大容量、高带宽的片内存储部件。流数据在多个计算核心之间及流处理器之间的流动都要经过SRF，SRF的结构和性能对流处理器的整体性能有很大的影响。

2 设计背景

FT64采用流编程模型。在流编程模型中，一个流应用由一系列消耗及产生数据流的计算核心和负责组织数据流在多个计算核心之间流动的流操作构成。每个数据流都是由相同类型数据记录组成的序列。每个计算核心都是一段程序，对输入流元素进行相同的操作，产生一个或多个输出流。流级程序指定核心级程序执行的顺序，组织数据在核心程序间的流动，并负责流应用中的标量计算。核心程序是一个循环结构，它处理每一个输入流中的元素，产生输出结果。

FT64的主要组成部分有：流控制器，SRF，微控制器，4个ALU运算簇，流cache控制器，存储控制器，主机接口

基金项目：国家自然科学基金资助项目(60473079)
作者简介：马驰远(1972－)，男，副研究员，主研方向：计算机体系结构；陈海燕，副研究员；齐树波，博士研究生；陈书明，教授、博士生导师；肖 嵘，讲师
收稿日期：2007-12-10 **E-mail：**machiyuan2005@hotmail.com

和互连网络控制器，如图 1 所示。

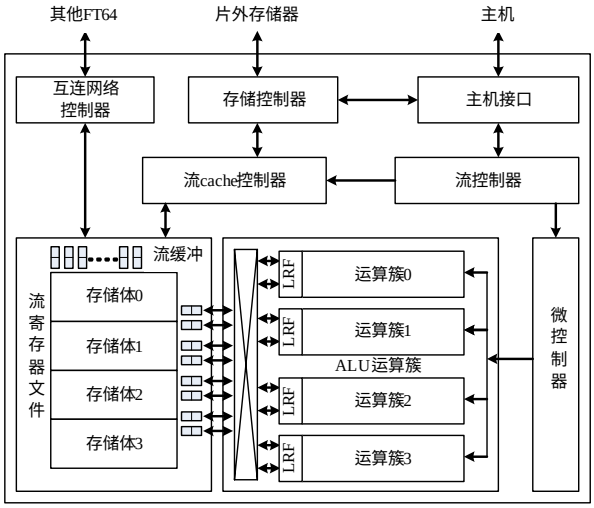

图 1　FT64 总体结构

主处理器执行流级程序，加载流数据和核心级程序到 FT64 的主存中(在流传输过程中，核心级程序代码可认为是一个普通的流数据)，然后发送流级指令给流控制器。流控制器将流级指令发射到相应的功能单元中执行。流级指令执行流程如下：

(1)通过 SRF，从主存加载核心程序代码到微控制器的指令存储器中；

(2)流控制器将流数据从主存加载到 SRF 中，当核心程序和流数据都准备好时，流控制器将启动微控制器指令存储器中核心程序的执行；

(3)微控制器使 4 个运算簇以 SIMD 方式处理 SRF 中的流数据，并产生输出流；

(4)计算结束时，流控制器将存在 SRF 中的输出数据传到主存中或通过网络接口传到其他 FT64 中；

(5)当 FT64 完成运算处理后，主处理器通过主机接口，读出 FT64 主存中的数据。在 FT64 中，计算核心所处理的流数据为生产者/消费者关系，计算核心间的流数据传递是在 SRF 中进行的，若要加快计算核心的处理速度，减少访问 cache 及外部存储器的频率，就需设计一个高效的 SRF。

3　SRF 设计与实现

计算核心处理的流数据一般较大，且同一时刻可能有多个流处于活跃状态，也就可能需要多组流数据，因此，SRF 的结构和容量需要仔细设计，以保证处于活跃状态的流数据不会在 SRF 中溢出。

3.1　SRF 结构

SRF是面向 4 个运算簇设计的，因此，SRF分为 4 部分。此外，SRF还可以为包括运算簇在内的多个部件提供数据传输服务。SRF主要由流描述符寄存器(Stream Descriptor Register, SDR)、流控制寄存器(Stream Control Register, SCR)、SRF存储器、流缓冲(Stream Buffer, SB)和仲裁器构成[3]。SRF 基本结构如图 2 所示。

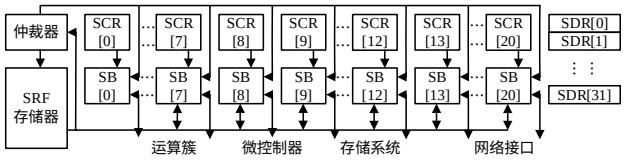

图 2　SRF 基本结构

SDR 是一个集中式寄存器文件，由 32 个 28 bit 的寄存器组成。SDR 用来描述流在 SRF 中的位置和长度信息。在流数据传输启动时，对应于 SDR 中的内容会被读到相应的 SCR 中去。

SCR 由 21 个 52 bit 的寄存器组成，每个 SB 对应一个 SCR，SCR 在物理上是分布寄存器文件。SCR 控制流在 SRF 与各部件之间的传输，包括流传输的启动、进行及终止。SCR 与 SB 是一一对应的，即只能控制它所对应 SB 的流传输，而不能控制其他 SB 的流传输。

SRF 存储器采用单端口 SRAM 结构，大小为 256 KB，共分为 2 048 个块，每块大小为 16 Byte，字长 64 bit；存储器有 4 个存储体，对应 4 个运算簇。存储器的数据位宽是 1 024 bit，即一个块的大小。每次对 SRF 存储器的访问都会读出或者写入一个块的数据。软件为每个流分配的空间是以块为单位的，因此，SRF 存储器的地址是块地址。

图 3 给出 2 个流在 SRF 中和片外存储器中的排列情况(假定片外访问的模式是跨步的)。

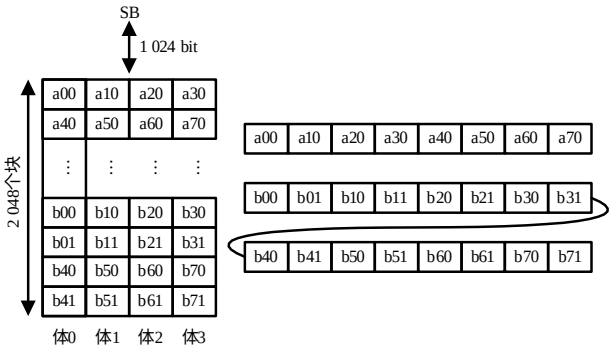

图 3　流数据在 SRF 和片外存储器中的放置

图 3 左半部分是 SRF 存储器的结构及 2 个流在 SRF 中的放置；右半部分是 2 个流在片外存储器中的放置情况。其中，流 a 的长度是 8 bit；流 b 的长度是 16 bit；流 a 的记录大小是 1 Byte；流 b 的记录大小是 2 Byte。*Pmn* 表示流 *P* 的第 *m* 个记录的第 *n* 个字。

从图 3 可以看出，记录在片外存储器中是以记录本身为单位连续放置的；在 SRF 中，同一个记录放置在同一个体中。因为流 a 的长度不是块的整数倍，所以为流 a 分配的空间的最后一个块中含有 8 个无效数据。

SB 调整 SRF 存储器与各功能部件之间的数据带宽不匹配现象，为多个功能部件访问 SRF 提供接口。在 SRF 的设计中，使用 21 个 SB(用 SB0~SB20 来表示)，通过 21 个 SB，将只有一个物理端口的 SRF 存储器映射为具有 21 个逻辑端口的 SRF 存储器，21 个 SB 通过动态仲裁，访问单端口存储器。SB 可使通过 SRF 的峰值带宽达到 156 GB/s。每个 SB 由 2 个半缓冲和 1 个流缓冲控制器组成，每个半缓冲的大小是 16 Byte，即一个块的大小。半缓冲也分为 4 部分，每部分包含 4 Byte。

这样的设计能够允许 SB 与功能部件之间的数据传输和 SB 与 SRF 存储器之间的数据传输同时进行，即一个半缓冲与功能部件传输数据，而另外一个半缓冲与 SRF 存储器传输数据。

图 4 给出了服务于运算簇的 SB 结构，并给出图 3 中，流 b(在此假设流 b 的长度是 32 bit，有 16 个记录)被运算簇读取时，在 SB 中的放置情况。

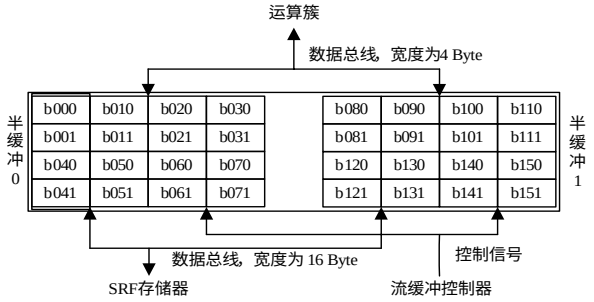

图 4　数据在 SB 中的排列

21 个SB在访问单端口的SRF存储器时，需要动态仲裁。仲裁器采用近似LULS(Last Used Last Serve)仲裁策略[4]，将最近被仲裁到的SB优先级设置为最低。这种算法可以保证每个SB较公平地获得总线使用权，硬件开销小。

3.2　实现结果

在完成对全芯片的RTL描述和模拟验证后，用逻辑综合软件Design Compiler对全芯片进行综合，用物理设计软件SOC Encounter进行后端物理设计，采用Artisan 0.13 μm工艺库。时序分析结果表明，SRF的最长路径时间为 1.9 ns，频率超过 500 MHz。芯片中SRF的面积为 13 mm^2。

4　SRF 性能评估

用多个多媒体及科学计算核心程序进行实验。这些程序包括：2 个 NPB 测试程序(CG, MG)，2 个 SPEC CPU2000 测试程序(Swim, Lucas)，3 个重要的科学计算核心程序(FFT, Jacobi 和 NLAG-5)。用流模型重新编写这些程序，使它们能在 FT64 上运行，使用 FT64 的时钟精确模拟器来获得运行参数。有些程序的数据集较小，改变这些程序的数据集，并在模拟器上多次执行，获得准确结果。

流处理器的设计目的是提供强大的计算能力，并最大限度地隐藏存储访问时间。从图中可以看出，多数程序中，SRF 的停顿占全部停顿的 72%。这表明，在 FT64 流处理器中，运算簇的计算功能非常强大，由于运算簇忙而使停顿较少，尤其在以 SIMD 方式进行运算时，一条指令同时对多个数据进行操作，这时，数据的提供成为关键，此时，执行瓶颈为 SRF 读写访问；另外，在 FT64 的多级存储结构中，SRF 有效命中了大部分数据访问，尤其是 SRF 的双缓冲机制，屏蔽了多数访问 cache 及片外存储器的延时，使数据访问和调度都在 SRF 内部进行，而 SRF 的单位访问时间要比 cache 和外存少，使数据访问速度加快，提高了系统性能，这说明 SRF 能够达到设计目标。在运行测试程序时，运算簇、SRF 和外层存储的执行停顿比例如图 5 所示。

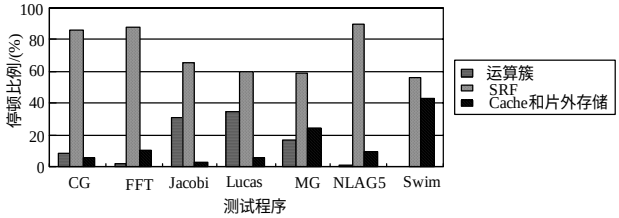

图 5　程序执行时运算簇、SRF 和外层存储的停顿比例

5　结束语

本文介绍一种面向流应用的 SRF 结构设计方法和数据传输机制，分析它在 FT64 流处理器中所起的作用。实验结果表明，这种结构能够适应流应用的需求。本文中的 SRF 对流数据的访问是顺序完成的，可以称它为顺序 SRF，而顺序 SRF 在某些带索引的流应用中，效率不高，下一步工作将重点研究在带索引的流应用中如何设计 SRF。

参考文献

[1] Kapasi U, Dally W, Rixner S, et al. The Imagine Stream Processor[C]//Proceedings of the 20th IEEE International Conference on Computer Design. [S. l.]: IEEE Computer Society, 2002.

[2] Dally W, Hanrahan P. Merrimac: Supercomputing with Streams[C]// Proceedings of Supercomputing Conference'03. USA: [s. n.], 2003.

[3] 齐树波, 陈海燕, 张民选. 流寄存器文件的研究与硬件实现[C]// 第十届计算机工程与工艺全国学术年会. [S. l.]: 中国计算机协会, 2006.

[4] 汪东军. 基于动静态混合算法的仲裁器模块设计[J]. 安庆师范学院学报, 2001, 7(2): 35-37.

(上接第 262 页)

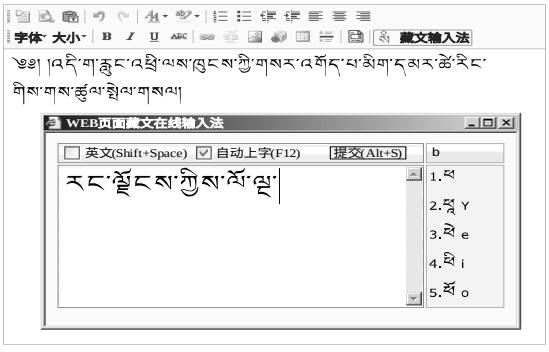

图 5　外挂法和网页编辑器 FCKeditor 的整合效果

5　结束语

Web 页面藏文输入法作为 Web 页面下藏文信息处理的基础技术，可以实现在浏览器中脱离本机输入法而实现藏文的输入，这为构建基于藏文的电子商务、企业信息管理(MIS)、网站内容管理(CMS)、电子政务以及藏文网络社区、聊天室、BBS 等网络系统提供了跨平台的藏文输入解决方案。

参考文献

[1] 朱巧明, 李培峰, 杨季文, 等. 基于 Windows9x/2000/NT 平台汉字输入法的设计[J]. 小型微型计算机系统, 2000, 11(3): 1217-1220.

[2] 西藏语委. GB/T 17543 1998 中华人民共和国国家标准信息技术藏文编码字符集(基本集)键盘字母数字区的布局[S]. 国家质量技术监督局, 1999.

[3] 于洪志. 藏文 Win32 平台的输入技术[J]. 西北民族大学学报: 自然科学版, 2001, 22(38): 5-8.

[4] 西藏语委. GB16959 1997. 信息交换用藏文编码字符集基本集[S]. 国家质量技术监督局, 1998.

[5] 于洪志. Web 环境下藏文信息处理技术[J]. 西北民族大学学报: 自然科学版, 2005, 26(3): 5-8.