

面向多任务的片上簇处理器流访存控制系统设计

胡 凯, 谢 憬, 毛志刚

(上海交通大学 微电子学院, 上海 200240)

摘 要: 为了解决分簇流处理器在进行多任务处理时访存系统效率不高的问题, 提出了一种新的流访存系统. 针对多任务处理情况下, 各簇间工作的差异性与独立性, 设计了新的流访存机制, 包括流访存系统结构、访存流控制、簇间流顺序调度和流信息配置等. 实验结果表明, 改进后的多簇流处理器能够工作在并行与多任务两种模式下, 同时在并行处理性能基本不变的情况下, 获得平均 20% 的多任务处理性能提升.

关键词: 流处理器; 多任务; 流访存系统

中图分类号: TN402

文献标识码: A

文章编号: 1000-7180(2014)06-0040-04

A Stream Based Memory Access Control System of an On-Chip Clustered Processor for Multi-task Application

HU Kai, XIE Jing, MAO Zhi-gang

(School of Microelectronics, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract: In order to solve the low efficiency of the memory access system in a clustered stream processor while doing multi-task processing, a new stream memory access system is proposed. Considering the diversity and independency between each cluster under multi-task condition, a new stream memory access mechanism is designed, including system architecture, stream control, stream schedule, and stream information configuration. Result shows that after modification, the processor can work in both parallel and multi-task mode, remain almost the same parallel performance, and achieve an average improvement of 20% in multi-task mode.

Key words: stream processor; multi-task; steam memory access system

1 引言

数字信号处理具有数据规律、重复运算以及密集计算等特性, 所以随着数字信号处理的需求越来越高, 出现了许多针对数字信号处理的处理器. 而这些处理器通常都拥有大量的运算单元, 并且将运算单元分组, 采用 SIMD 的方式, 对数据进行并行操作^[1-2].

近年来产生的流体系结构面向这类应用, 其工作方式更适用于大量规律性的密集计算^[3]. 同时流体系结构将运算与访存分离, 能较好地隐藏访存延迟, 并通过合理的流组织提高访存效率^[4]. 然而对于

现有的流访存结构, 其流访存方式适合于并行处理, 但并不满足多任务处理的需求, 会造成运算性能的浪费^[5].

本文针对流体系结构的访存特性进行研究分析, 对原有多簇处理器的流访存结构进行改进, 使其能够适应并行计算以及多任务处理两种需求.

2 流处理器结构

如图 1 所示为本文流处理器的结构图, 主要包括 4 个运算簇, 其中包含 4 个加法器、2 个乘法器以及输入输出单元, 每个簇分别对应一个 PE2SB 和 SB2PE 数据缓存, 采用 FIFO 的形式传输数据. 图中

收稿日期: 2013-08-27; 修回日期: 2013-10-18

基金项目: 国家“六八三”重大专项子课题(2012AA012702); 国家自然科学基金面上项目(61176037)

虚线圈出的部分是本文重点分析及改进的部分,主要负责访存流的生成、控制与数据传输,而余下部分为 L1 Cache 与下层 Cache 以及片外存储器的交互模块。

该结构采用 SIMD 和多任务处理两种工作模式。在 SIMD 模式下,应用中相同运算的部分将被分配到 4 个簇中同时进行,适合并行处理^[6]。而此时,图 1 中虚线包围的流访存结构会将数据按顺序分配给各簇,以满足并行运算的数据需求。

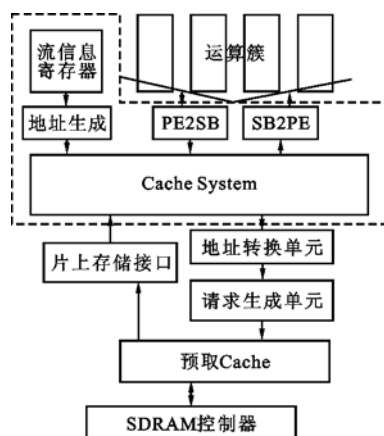


图1 基本流处理器结构图

而在多任务处理情况下,各簇需要独立运行应用或者是同一应用的不同部分,所以相互间的数据并不相关,而且各簇的执行顺序也不确定,所以,通过现有的流结构将数据同时分配给各簇,很难适应各簇对数据的不同需求。

3 适用于多任务处理的流访存系统结构

3.1 硬件结构

如图 2 所示为改进后的流访存结构,与图 1 虚线框内结构对应。图中为一个 4 簇结构,每个簇都与一个输入 FIFO 和输出 FIFO 相连,数据会通过簇内的输入输出单元完成簇与 FIFO 的数据交换。

流信息寄存器中保存着需要被执行的输入输出流的信息。流地址序列生成单元负责将流信息,包括流首地址、流跨步、流长度等,生成一串流访存地址序列。待输出流寄存器主要是为了解决各流之间的数据相关性。以上模块与并行处理工作方式相同,故可以进行复用。

为了适应多任务处理,本文设计、修改了流访存控制单元,并增加了流数据缓冲寄存器。控制单元根

据系统工作模式与访存状态,发出信号来控制整个流访存结构的工作。流数据缓冲寄存器是一个暂存正在访存的输入流的空间,当一个流的所有地址都访存完成后,控制单元会将相应数据从该暂存空间送入对应簇的输入 FIFO 中。

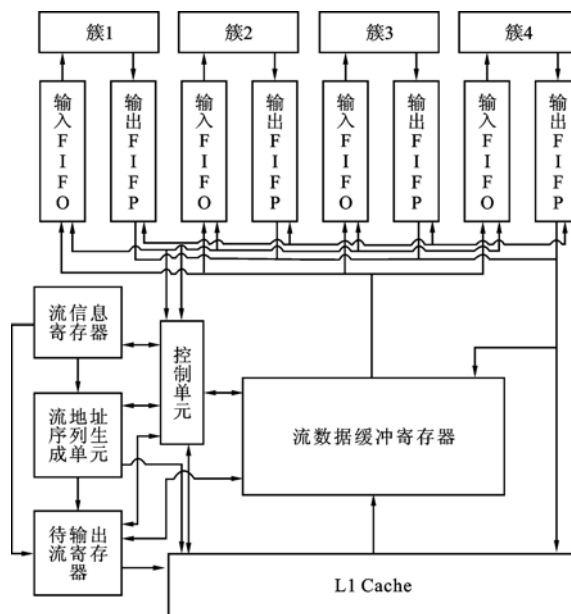


图2 改进后的流访存系统结构

3.2 控制及工作方式

改进后,访存结构能同时支持并行处理以及多任务处理,并通过配置控制单元中的模式寄存器设定其工作状态。

如图 3 所示为多任务情况下流访存结构的工作方式。定义流信息即从外部接口对流信息寄存器进行配置,控制单元根据当前的流状态决定下一步的操作。如果缓冲区中有输入流已访存完毕,或输出 FIFO 数据已准备好,则开始该流的数据传输,将数据送入输入 FIFO 或写入 Cache。如果没有需要被传输的数据,则选择下一个需要被开启的流,生成流地址序列,进行访存或者等待输出。

如图 4(a)所示为一个输入流在多任务处理中的访存流程,流信息从流信息寄存器进入到序列生成单元,产生的地址序列进入 Cache 访存,将返回的数据写入缓冲区。如果存在数据相关,则数据会在相应的输出流传输数据时获得,最后缓冲完成的数据将送入对应簇的输入 FIFO 中。

图 4(b)所示为一个输出流在多任务处理中的访存流程,在生成了地址序列以后,流地址会进入待

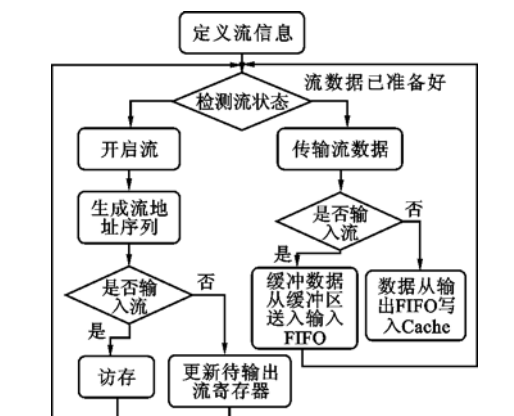


图3 多任务工作方式

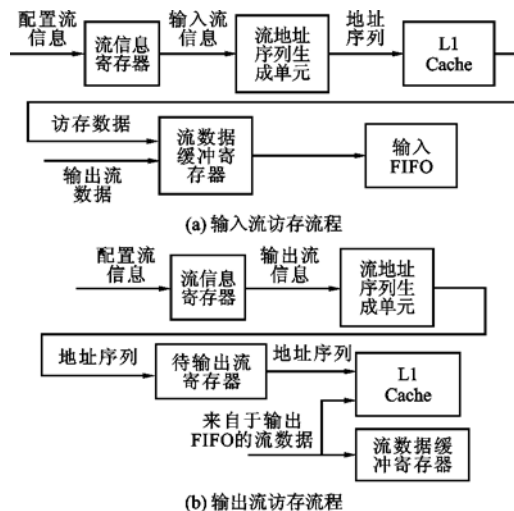


图4 流的访存流程

输出流寄存器,当数据在输出FIFO中准备好时,数据和地址会一起被送往Cache执行写入操作.当存在数据相关时,相应数据会被写入流数据缓冲寄存器中.

3.3 簇间流优先级

因为在多任务处理的情况下,每个簇独立工作,所以通过合理调度簇间的流顺序,能减少运算簇等待数据的情况.

本设计的簇间流优先级定义如下:

- ①簇内流水线因等待数据而停滞;
- ②输入FIFO已空;
- ③流信息寄存器中下一条为输出流;
- ④输入FIFO数据量最少;

每个运算簇自己的流采用顺序执行的方式,而多个簇间的开启顺序由当前状态和优先级决定.如果某个簇的运算因等待数据而停滞,则该运算簇的流将作为下一个流优先被开启.在没有运算簇符合

第一个条件时,如果有输入FIFO为空的簇,则该运算簇的流将被优先开启.同样如果所有运算簇输入FIFO内均有数据,则选择下一条为输出流的运算簇.如果均不满足,则开启输入FIFO数据量最少的运算簇的流.

3.4 流信息配置

如图5(a)所示为流信息寄存器的配置情况,本文将原本多簇共用1个寄存器组,改为了4个寄存器组.并行处理时流信息寄存器组仍为多簇共有.而多任务情况下,各簇分别对应一个寄存器组.其中流编号为流的标识,流首地址定义了该流要访问的地址序列的第一个地址,流跨步对应流地址序列中相邻两个地址之间的偏移量,流长度定义了该流地址序列的长度,即地址的个数,输入/输出位用来表示该流是输入流还是输出流.

流编号	流首地址	流跨步	流长度	输入/输出
...	...	...	...	...

(a) 流信息寄存器

有效位	流地址	流编号	缓冲寄存器号
...	...	...	...

(b) 待输出流寄存器

图5 流信息的配置

流地址序列的生成原理如式(1)所示, i 为0到(流长度-1)之间的整数,通过流首地址、流跨步和流长度,可以得到一组流地址,个数由流长度决定.

$$\text{Addr}[i] = \text{HeadAddr} + i \times \text{Stride} \quad (1)$$

如图5(b)所示为待输出流寄存器的配置情况.其中每条记录包含4项内容,流地址保存流地址序列中的地址信息,流编号保存流的标识,缓冲寄存器号保存存在先写后读冲突的输入流地址所在的缓冲区寄存器号,这样该输出流开启后,就可以将数据直接写入该缓冲区.有效位表示的是该地址是否存在数据相关性.

4 性能分析

本节拟通过在改进前后的结构上映射多个DSP应用来验证本设计方案的可行性,以及对实际性能的影响.实验中改进前的流访存结构包含深度为32的输入、输出FIFO,下层连接16kB的4-bank L1 Cache,及256KB的L2 Cache,改进后增加了256B的流缓冲区和相应控制电路,并使用上文讨

论的互联方式.

表 1 所示为对两种流访存结构进行综合的结果,采用了 65 nm 的工艺库,从表中可以看出,改进后面积增大了 6%左右.

表 1 综合结果

	时序/ns	面积/ μm^2
改进前	2.13	246 704
改进后	2.25	262 246

实验中,并行处理情况下,单个应用会采用 SIMD 的方式被分配到四个运算簇进行执行.而多任务处理情况下,四个运算簇会分别执行不同的应用.测试应用包括 128 阶 FIR、1 024 点 FFT、1 024 点 IFFT 和 32×32 矩阵乘^[7].这几个应用是数字信号处理中经常用到的算法,有很强的代表性和特点,所以本文选择这几个程序来进行比较.

表 2 和表 3 分别为并行和多任务情况下各应用的运行周期数.从中可以发现,对比改进前后,在多簇并行处理的情况下,四个应用的性能几乎没有受到影响.而由于对流访存系统进行了修改,在多任务处理情况下,四个应用可以获得 10%到 40%不等的性能提升.

表 2 并行处理实验结果

应用 \ 周期数	改进前	改进后	性能比
128 点 FIR	3 275	3 477	0.94
1 024 点 FFT	88 122	91 700	0.96
1 024 点 IFFT	95 586	101 082	0.95
32×32 矩阵乘	103 073	107 186	0.96

表 3 多任务处理实验结果

应用 \ 周期数	改进前	改进后	加速比
128 点 FIR	5 089	4 223	1.205
1 024 点 FFT	259 072	235 520	1.100
1 024 点 IFFT	295 698	262 144	1.128
32×32 矩阵乘	510 018	363 520	1.403

5 结束语

本文提出了适用于多任务处理的多簇处理器流访存结构,通过对流信息寄存器、控制单元进行修改,并增加流数据缓冲寄存器,来实现对多任务处理的支持,使系统同时兼容并行和多任务处理两种工作模式.通过两者硬件模块的复用,减少了额外的面积开销,并在保持原有并行处理性能的情况下,获得了较好的多任务处理性能.

参考文献:

[1] Scott Rixner, Stream processor architecture[M]. USA: Kluwer Academic Publishers. 2001:53-59.

[2] Munshi A, Wong A, Clinton A, et al. A parameterizable SIMD stream processor[C]//Electrical and Computer Engineering. Canadian, 2005:806-811.

[3] 伍楠.流处理器 MASA 内核的研究及实现[D].长沙:国防科学技术大学,2005.

[4] Peter Mattson. A programming system for the imagine media processor[D]. Stanford University, 2002.

[5] Ji K, Peilin L, Yu Z. Atomic streaming: a framework of on-chip data supply system for task-parallel MPSoCs[J]. Computer Architecture Letters,2012(11):5-8.

[6] Khailany B, Dally W J, Kapasi U J, et al. Imagine: media processing with streams[J]. Micro, IEEE,2001 (21):35-46.

[7] Alan V Oppenheim, Ronald W Schafer. Discrete-time signal processing[M]. 3rd ed. NJ:Prentice Hall, 2009.

作者简介:

胡凯男,(1989-),硕士研究生.研究方向为大规模集成电路设计.

谢憬男,(1981-),助理研究员.研究方向为大规模集成电路设计、片上处理器体系结构、先进集成电路设计方法.

毛志刚男,(1962-),教授,博士生导师.研究方向为大规模集成电路设计、可重构体系结构设计、集成电路与元器件可靠性等.