第34卷 第10期 计 算 机 工 程 2008年5月
Vol.34 No.10 Computer Engineering May 2008
·博士论文· 文章编号：1000—3428(2008)10—0022—03 文献标识码：A 中图分类号：TP313

RISC3200的MDS-II指令集扩展

姚英彪[1]，汪 斌[1]，章坚武[1]，刘 鹏[2]
(1. 杭州电子科技大学通信学院，杭州 310018；2. 浙江大学信息科学与工程学院，杭州 310027)

摘 要：通过利用媒体核心算法评估RISC3200的第一代媒体扩展指令集MDS-I的性能，发现MDS-I存在数据处理效率高但数据供应效率低的特点。基于该原因扩展了用于数据供应的第二代媒体扩展指令集MDS-II。实验结果表明，在扩展媒体指令集后，RISC3200的媒体核心算法的处理性能提高2~5倍左右。

关键词：微处理器；精简指令集；媒体应用；指令扩展

MDS-II Instruction Sets Extension of RISC3200

YAO Ying-biao[1], WANG Bin[1], ZHANG Jian-wu[1], LIU Peng[2]
(1. College of Telecommunication Engineering, Hangzhou Dianzi University, Hangzhou 310018;
2. College of Information Science and Engineering, Zhejiang University, Hangzhou 310027)

【Abstract】 By using the media kernel algorithms to evaluate the performance of MDS-I, the first generation media extension sets of RISC3200, it is found that MDS-I sets have powerful media data processing ability but awkward media data providing ability. The MDS-II sets for high efficiency media data provision is extended. Experimental results show MDS sets of RISC3200 can achieve about 2~5 speedups for media kernel algorithms compared with its MDF sets.

【Key words】 microprocessor; RISC; media application; instruction extension

1 概述

随着微处理器设计和加工技术的进步及其应用领域的不断扩展，静止图像、3D图形、动态音视频等媒体处理已成为任何一个微处理器都必须考虑的工作负载[1]。针对这个问题，通用处理器一般采用媒体指令扩展的方法，通过扩展面向媒体应用的指令，能以非常低的硬件代价换得非常高的性能增强。这方面代表性的工作有：基于CISC结构的媒体指令扩展，如Intel的MMX[2]指令集；基于RISC结构的媒体指令扩展，如HP的MAX和MAX-2[3]、Sun的VIS[4]指令集。

RISC3200是本课题组自主设计的面向嵌入式应用的32位RISC处理器[5]，其知识产权完全由本课题组拥有。为增强其媒体信号处理性能，本文也采用媒体指令扩展的方法。在它的媒体扩展指令集MDS设计中，由于以下原因决定自主设计MDS指令集：(1)CISC上的媒体扩展指令集继承了传统的CISC的2地址的指令结构，这与RISC3200的3地址RISC指令结构不一样。(2)RISC上的媒体扩展指令集种类繁多，各有特点，并且这些指令集都是与它们的应用范围和微结构特点相匹配，不能直接照搬使用。

在RISC3200的第一代媒体扩展指令集MDS-I设计过程中，主要是参考国际上成功的媒体指令集，从中挑选出一批能在中低端媒体处理中使用，并且适合在RISC3200微结构上扩展的指令。但是在实际的媒体算法评估中，发现MDS-I存在数据处理能力强但数据供应能力弱的缺点。针对这个问题，本文提出并设计了RISC3200的第二代媒体扩展指令集MDS-II。

2 MDS-I指令结构及其优缺点

在RISC3200进行MDS-II扩展前，其指令结构包括MDF和MDS-I两部分。MDF是与MIPS-I兼容的RISC指令集，MDS-I是第一代媒体扩展指令集，主要是按单指令多数据(SIMD)方式扩展的媒体运算指令，如表1所示(不同粒度、相同功能归为1条)。需要指出的是，MDS-I是在新的8x64位的媒体寄存器文件上进行扩展的。

表1 MDS-I指令集

指令类型	寻址模式	指令操作码举例
数据传输(4条)	INST MRd, Rs	PMTHI/LO, PMFHI/LO
数据转换(4条)	INST MRd, MRs, MRt	PACKSS, PUNPCKH
算术指令(21条)		PADD, PMAC, PAVG, PMAX
比较指令(2条)		PCMPEQ, PCMPGT
逻辑指令(4条)		PAND, PNOR
移位指令(3条)		PSLL, PSRL

在正式提出MDS-II前，先借助一个实际的媒体核心算法1 024点的32阶FIR滤波来分析MDS-I的优缺点。该算法采用16位的数据宽度，MDS-I指令集取得的性能加速比如表2第5列所示，此性能加速比低于预期的结果。为分析内在原因，根据指令实际作用将FIR汇编程序的指令分为3类：数据处理指令，程序控制指令和数据供应指令。数据处理指令是实际完成数据处理的运算指令；程序控制指令完成程序控制，包括分支指令和循环变量计算指令；数据供应指令完成内存访问，包括地址计算指令和Load/Store指令。这3类指令在实际运行中的执行周期数比例如表2中第2列~第4列所示。

基金项目：国家“863”计划基金资助项目(2002AA1Z1140)
作者简介：姚英彪(1976－)，男，讲师、博士，主研方向：媒体信号处理，计算机体系结构；汪 斌，讲师、博士；章坚武，教授、博士；刘 鹏，副教授、博士
收稿日期：2007-05-27 **E-mail：**yaoyingbiao@163.com

表 2　FIR 算法的评估效果

程序类型	数据处理指令/(%)	程序控制指令/(%)	数据供应指令/(%)	加速比
MDF	60.2	13.3	26.5	1.00
MDF+MDS-I	7.6	18.7	73.7	1.47

从表 2 中可以看到，在进行 MDS-I 扩展后，程序控制指令执行周期数比例变化不大，但是数据处理指令和数据供应指令执行周期数比例发生很大变化。一方面，RISC3200 处理 FIR 算法需要的数据处理指令执行周期数减少很大，这说明它的数据处理能力大幅度提高，这是 MDS-I 扩展的优点；但是另一方面，数据供应指令执行周期数增加许多，它使得 MDS-I 数据处理能力强的优势无法明显体现出来，影响了 MDS-I 最后的性能加速比，这是 MDS-I 扩展的缺点。另外，在评估其他的媒体核心算法时，也碰到上面 FIR 算法同样的问题。基于此本文认为 MDS-I 扩展存在数据处理效率高、数据供应效率低的缺点，这也是在 RISC 构造上进行媒体指令扩展的通病。

3　MDS-II 指令扩展

MDS-I 的数据供应指令设计不合适，造成媒体寄存器的数据供应不方便，从而增加了许多对实际提升媒体处理性能无关的 overhead 指令。针对这个问题，本文从增强媒体寄存器数据供应能力的角度出发，根据实际的媒体数据的可能数据源，设计了 3 类数据供应指令。

3.1　媒体寄存器和内存的数据交换

在应用 MDS-I 进行具体的媒体算法的评估过程中，发现媒体寄存器数据供应效率低下最主要的原因在于不直接支持内存到媒体寄存器的数据通路。在 MDS-I 的扩展中，媒体寄存器和内存进行数据交换是按图 1 的方式进行的。

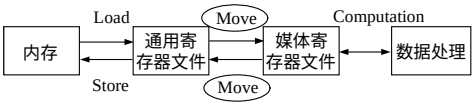

图 1　MDS-I 中媒体寄存器和内存的数据交换

从图 1 可以看到，内存中的数据必须经过通用寄存器中转才能完成和媒体寄存器的数据交换，显然这个过程全部是与提高媒体处理性能无关的 overhead 指令。因此，决定在 MDS-II 的扩展中提出以下 4 条指令完成媒体寄存器和内存的数据直接交换：

PLoadHi/Lo MRt, offset(base), PStoreHi/Lo MRt, offset(base)

在上面 4 条指令中，base 为基地址寄存器(通用寄存器)，MRt 为媒体寄存器。可以看到，本文只扩展从内存中取 32 位数据到媒体寄存器的高或低 32 位。这是因为：(1)RISC3200 微结构里数据 cache 的行大小为 32 位。如果扩展 64 位的取数，那么它就对应 2 个 cache 行，这对 RISC3200 的微结构改动非常大。(2)在实际的媒体应用算法中，规整的按双字对齐的数据访问并不多见，经常是按半字、字对齐的连续的、甚至非连续的内存访问。

3.2　媒体寄存器和通用寄存器的数据交换

虽然扩展了内存和媒体寄存器的直接数据交换，但是由于指令编码和指令结构兼容的限制，MDS-II 指令集的 Load/Store 指令的功能没有 MDF 指令集的 Load/Store 指令的功能齐全(如不支持字节、半字等操作)，有些复杂的内存访问还必须通过通用寄存器中转。另外，有些情况下媒体寄存器的数据直接就来源于通用寄存器。因此，灵活的媒体寄存器和通用寄存器的数据交换对增强媒体数据供应能力也是至关重要的。

在 MDS-I 的扩展中，PMTHI/LO 和 PMFHI/LO 这 4 条指令的功能就是完成通用寄存器和媒体寄存器的数据交换，但是它们进行数据交换的粒度是 32 位，即它们只能完成通用寄存器和媒体寄存器的高或低 32 位的数据交换。在实际评估 MDS-I 的性能中发现，32 位的数据交换只是媒体寄存器和通用寄存器进行数据交换的许多形式中的一种，实际应用中的数据交换比这复杂，如上面提到的 FIR 算法的内存访问就是按图 2 的方式进行的。

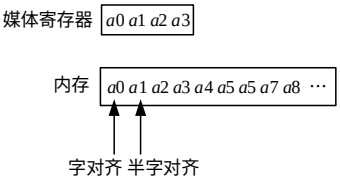

图 2　媒体寄存器非字对齐的内存访问

在图 2 中，媒体寄存器访问内存是按半字对齐的方式进行的。具体说，就是先访问 *a*0, *a*1, *a*2, *a*3 这 4 个半字；然后访问 *a*1, *a*2, *a*3, *a*4 这 4 个半字；依次类推。对图 2 所示的内存访问，可以将 *a*4 先 Load 半字到通用寄存器，然后从通用寄存器 Move 到媒体寄存器。因此，在 MDS-II 中设计了 2 条媒体寄存器和通用寄存器进行数据交换的指令：

PMTMRF/PMFMRF MRd, Rs, Sa

其中，MRd 为媒体寄存器；Rs 为通用寄存器；Sa 为 8 位立即数。这 2 条指令的功能如图 3 所示。可以看出，本文设计的通用寄存器和媒体寄存器是按 16 位的粒度进行数据交换的，并且通用寄存器中的 2 个 16 位数和媒体寄存器中的 4 个 16 位数可以进行任意位置的数据交换。从指令功能上说，它包括了原有的 PMTHI/LO 和 PMFHI/LO 指令。

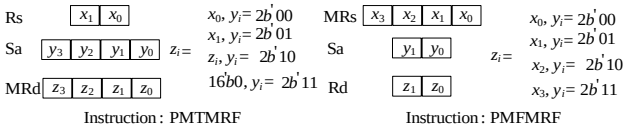

图 3　PMTMRF/PMFMRF 指令的功能

3.3　媒体寄存器和媒体寄存器的数据交换

媒体寄存器最后的数据源是媒体寄存器。以图 2 为例，媒体寄存器第 1 次使用 *a*0, *a*1, *a*2, *a*3 这 4 个半字，第 2 次使用 *a*1, *a*2, *a*3, *a*4 这 4 个半字。可以看到 2 次之间需要用 *a*4 替换 *a*0，另外 *a*1, *a*2, *a*3 的位置要换成以前 *a*0, *a*1, *a*2 的位置。*a*4 替换成 *a*0 可以用上面设计的 PMTMRF 完成，但还必须设计调整 *a*1, *a*2, *a*3 位置的指令。基于以上分析，在 MDS-II 中设计了这样一条媒体寄存器和媒体寄存器进行数据交换的指令：

PSHUFD MRd, MRs, Sa

其中，MRd, MRs 和 MRt 为媒体寄存器；Sa 为 8 位立即数。PSHUFD 指令完成媒体寄存器与媒体寄存器的按 16 位宽度进行的数据交换，其指令功能如图 4 所示。

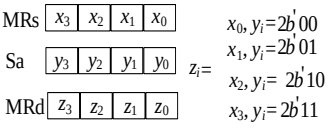

图 4　PSHUFD 指令功能

4　媒体指令扩展的性能评估

在完成 MDS-II 指令集设计后，本文从媒体处理性能提

升程度和硬件实现代价 2 个方面评估了 RISC3200 媒体扩展指令集 MDS 的性能。

4.1 性能评估

为评估 MDS 的性能，本文从实际的媒体应用程序中选择了表 3 所示的核心媒体测试程序。本文对以上所有算法都开发了相应的用基本指令和用媒体扩展指令实现的测试程序(全部采用 16 位的运算)。另外，为消除算法变化和编程人员能力不同引起的性能损失，测试程序都是参照同一个 C 程序由同一个编程人员人工开发完成，并根据 RISC3200 微结构的特点进行了性能优化。最后，本文用以上测试程序在 RISC3200 的 RTL 级仿真平台上进行仿真，统计测试程序需要的指令周期数，结果如表 4 所示。

表 3 核心媒体测试程序列表

测试程序	程序内容描述及内存访问特点
dotp	64 点的一维数组内积，按字对齐的内存访问
fir	1 024 点的 32 阶的 FIR 滤波器，按半字对齐的内存访问
idct	8×8 点的二维 IDCT 变换，按半字对齐的内存访问
dct	32 点的一维 DCT 变换，按字对齐的内存访问
cfft	512 点的复数 FFT 变换，不规整的内存访问
subband synthesis	32 个音频样点的合成，不规整的内存访问

表 4 核心媒体测试程序性能加速比

测试程序	MDF 执行周期/个	MDF＋MDS 执行周期/个	加速比
dotp	582	150	3.9
fir	327 685	146 940	2.2
idct	1 112	464	2.4
dct	4 346	940	4.6
cfft	133 702	66 884	2.0
subband synthesis	11 520	6 560	1.8

4.2 硬件实现代价

在RISC3200 微结构上实现媒体扩展指令的主要硬件代价包括：媒体指令译码模块，功能扩展的执行单元(ALU, MAC)，媒体寄存器文件等。基于SMIC的 0.18 μm标准单元库，使用Synopsys Design Compiler将设计综合到门级，在最坏工作情况下(1.62 V, 125 ˚C)，本文得到表 5 所示的时延和面积结果。可以看到其时延满足RISC3200 的 200 MHz的时钟要求，而媒体扩展的硬件开销只占整个核的 3.83%。

根据表 4 和表 5 的实验结果，得出如下结论：(1)媒体指令扩展是一种非常有效的提升处理器媒体处理性能的方法。以后的 RISC3200 的指令结构设计中，媒体指令扩展应进一步完善。(2)本文设计的媒体扩展指令是成功的，它以 3.83%的硬件代价换取 2~5 倍的媒体关键处理的性能改善。(3)在进行媒体扩展指令设计时，不仅要注重直接用于媒体处理的运算指令的设计，也要注重辅助数据供应指令的设计。只有这 2 方面的指令都设计好，才能大幅度提高媒体处理的性能。

表 5 媒体扩展单元的面积与时延

扩展部件	面积/μm^2	时延/ns
媒体译码部件	5 300	1.23
ALU	147 780	4.30
MAC	458 920	4.89
媒体寄存器文件	97 430	2.78

5 结束语

本文提出一种 RISC3200 的第二代媒体扩展指令集 MDS-II。在该指令集中扩展了媒体数据供应指令。它包括媒体寄存器和内存、媒体寄存器和通用寄存器、媒体寄存器和媒体寄存器的数据交换指令。实验结果表明，其核心算法的处理性能有较大幅度的提高。

参考文献

[1] Dasu A, Panchanathan S. A Survey of Media Processing Approaches[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2002, 12(8): 633-645.

[2] Peleg A, Weiser U. MMX Technology Extension to the Intel Architecture[J]. IEEE Micro, 1996, 16(4): 42-50.

[3] Lee R B. Multimedia Extensions for General-purpose Processors[C]//Proc. of IEEE Workshop on Signal Processing Systems. [S. l.]: IEEE Press, 1997: 9-23.

[4] Tremblay M. VIS Speeds New Media Processing[J]. IEEE Micro, 1996, 16(4): 10-20.

[5] Xiao Zhibin, Liu Peng. Optimizing Pipeline for a RISC Processor with Multimedia Extension ISA[J]. Journal of Zhejiang University Science, 2006, 7(2): 269-274.

(上接第 21 页)

高消息公平率。bucket 算法的公平率只受用户单向时延抖动的影响，所以在相同条件下的公平率要高于 cslag 算法。但是 bucket 算法要求同步客户端电脑时间。

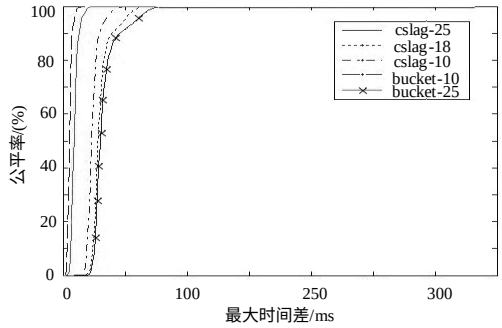

图 3 不同最大客户平均时延下的公平率

4 结束语

本文算法一方面考虑到了用户的反应时间，使得反应快的用户具有较高的获胜概率，另一方面又避免了文献[3]中对某些消息的不平等处理。并且可以通过分布多台服务器，使用户连接到离自己较近的一台服务器来提高游戏的公平性。

参考文献

[1] Mauve M, Vogel J, Hilt V, et al. Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications[J]. IEEE Transactions on Multimedia, 2004, 6(1): 47-57.

[2] Lin Yow-jian, Guo K, Paul S. Sync-MS: Synchronized Messaging Service for Real-time Multi-Player Distributed Games[EB/OL]. (2002-10-20). http://citeseer.ist.psu.edu/lin02syncms.html.

[3] Guo K, Mukherjee S, Rangarajan S, et al. A Fair Message Exchange Framework for Distributed Multi-player Games[C]//Proc. of ACM NetGames'03. California, USA: [s. n.], 2003: 21-33.

[4] Ferretti S, Palazzi C E, Roccetti M. Buscar el Levante Por el Poniente: In Search of Fairness Through Interactivity in Massively Multiplayer Online Games[C]//Proc. of the 3rd Consumer Communications and Networking Conference. [S. 1.]: IEEE Press, 2006.

[5] Carson M, Santay D. NIST Net: A Linux-based Network Emulation Tool[J]. ACM SIGCOMM Computer Communication Review, 2003, 33(3): 111-126.