Matrix DSP 中多线程机制的研究与设计

邓 宇 孙永节 万江华
(国防科技大学计算机学院 长沙 410073)

摘 要 深入研究了 YHFT_Matrix 高性能 DSP 中的一种多线程机制,重点介绍了其循环指令缓冲的读写机制、单线程与多线程之间的模式切换机制。在基于 65nm 工艺下,经过综合,代码面积、功耗都有减少,关键路径优化 0.07ns。对程序的执行评估测试的分析结果表明:多线程工作模式相比单线程工作模式,其处理器性能 IPC(Instructions Per Cycle)平均提高了 9.64%。

关键词 多线程,循环指令缓冲,模式切换
中图法分类号 TP303 **文献标识码** A

Research and Design of Multiple Thread Mechanism in Matrix DSP

DENG Yu SUN Yong-jie WAN Jiang-hua
(School of Computer Science, National University of Defense Technology, Changsha 410073, China)

Abstract This paper presented the study of a multiple thread mechanism in YHFT_Matrix high performance DSP. It emphatically introduced the read and write mechanism of the loop instruction buffer and the mode switch mechanism between single_threaded and multithreaded. Based on the 65nm process, after being synthesized, both the code area and power consumption have been decreased, and the key path optimizes 0.07ns. The implementation of the program assessment test was analyzed, and the outcome shows that compared with the operation mode of single_threaded, the processor performance IPC(Instructions Per Cycle) has an average increasing of 9.64%.

Keywords Multiple thread, Loop instruction buffer, Mode switch

1 引言

现代数字信号处理器主要用于对大量的数据进行处理。在无线通信、图像视频处理等应用领域,涉及到一些固定的算法程序反复调用,这往往体现为代码中大量的循环。据统计,在一些媒体应用中消耗在循环代码执行中的时间可以达到70%以上。因此循环缓冲技术得到了很好的应用发展[1,2]。

在 YHFT_Matrix 高性能 DSP 的多线程模式下,配合相关控制机制实现的向量部件指令来源于循环指令缓冲部件,标量部件指令来源于取指部件。处理器能够根据应用程序中并行处理对串行处理的不同协作需求,动态选择不同的线程执行模式。当并行处理需要大量的串行处理进行配合时,处理器采用单线程执行模式,使标量、向量单元执行由标量、向量指令组成的同一个指令流。在单线程模式下,通过寄存器级的交互,实现标量、向量单元间灵活细粒度的交互。而当并行处理只需少量串行处理的支持时,处理器主要执行在多线程模式下[3],使标量、向量单元同时运行不同的指令流,只是在并行处理需要串行处理配合的时间点,才会回到单线程的执行模式下运行。在多线程模式下[4]向量单元执行独立的并行处理时,标量单元能够执行其他的串行任务。

本文首先介绍了 YHFT_Matrix 高性能 DSP 中支持多线程的超长指令字微体系结构[5,6],在此基础上提出了循环指令缓冲部件实现的多线程机制[7]。本文第 2 节主要介绍了支持多线程的微体系结构和循环指令缓冲结构;第 3 节主要介绍了循环指令缓冲部件的读写机制和线程模式切换机制,对循环指令缓冲读写以及单线程与多线程之间的切换机制进行了详细的阐述;第 4 节介绍了设计循环指令缓冲后在性能方面的评估分析;最后为结束语。

2 总体结构设计

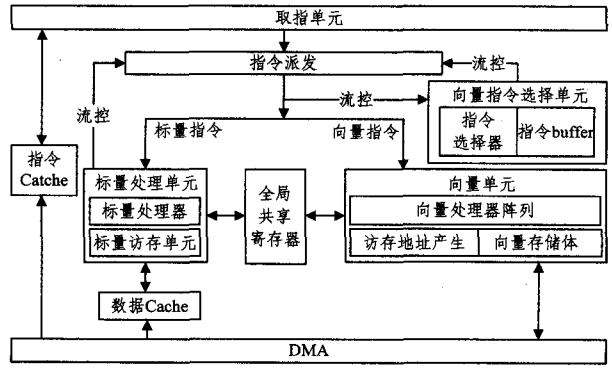

图 1 支持多线程的超长指令字微体系结构

YHFT_Matrix 高性能 DSP 体系结构的处理核心由标量处理单元和向量处理单元组成,其中标量处理单元由 1 个 PE

到稿日期:2012-06-23 返修日期:2012-09-29 本文受国家"核高基"重大专项(2009ZX01034-001-006)资助。
邓 宇(1987—),男,硕士生,主要研究领域为微处理器设计,E-mail:andyyu2010@163.com;孙永节(1962—),男,研究员,硕士生导师,主要研究领域为高性能微处理器设计、低功耗技术;万江华(1977—),男,助理研究员,主要研究领域为微处理器体系结构。

组成，向量处理单元由 16 个 PE 组成，标量单元和向量单元之间可以通过全局共享寄存器来进行数据通信，它们共享 DMA。正是鉴于这种标量单元和向量单元各自独立的如图 1 所示的体系结构，提出了多线程模式结构的设想。其主要是通过额外增加一个循环指令缓冲，配合相关的控制机制实现的。

循环指令缓冲的主要结构是一个拥有特定映射方式的指令缓存，用于存储部分或全部的循环指令。其结构如图 2 所示。

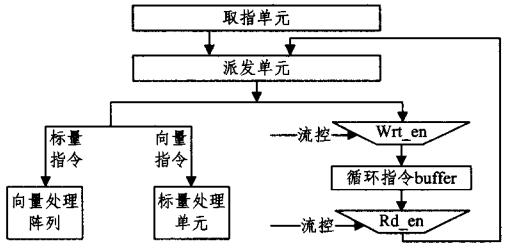

图 2　双指令流结构

该结构在单线程模式下，循环指令缓冲对派发单元没有反馈作用，它只是监控从派发单元派发出来的指令流，在写指令缓冲之前，通过标量单元配置指令流中循环体相应的起止地址以及循环次数、层次等参数。当写循环缓冲有效时，便开始写入循环指令；当读循环缓冲有效时，写缓冲是无效的，此时由单线程模式切换为多线程模式，循环缓冲向派发单元反馈指令。在多线程模式及读循环缓冲有效时，取指单元送出的指令都是标量指令，且与循环缓冲中的向量循环指令不存在相关性。本文增加的指令缓冲除了具有传统指令可以缓冲循环指令、支持软件流水调度等优点外，还具有以下优点：在进行向量操作时，标量部件也能利用其进行其它的操作，以充分解放标量部件，从而提高处理器资源利用率和并行处理性能，实现多线程。

3　循环指令缓冲读写机制

YHFT_Matrix 中循环指令缓冲的大小为 5.536kB，存储体的每个单元存放一个向量指令包和一个 TAG 位，一共可以存放 256 个(200 左右的指令条数基本可以满足大部分循环)指令包。循环指令缓冲部件存放循环体指令，包括需要在向量单元中处理的向量指令，以及流控类指令(NOP 指令)。循环指令缓冲部件负责在多线程执行模式下为向量单元提供指令。其读写机制如图 3 所示。

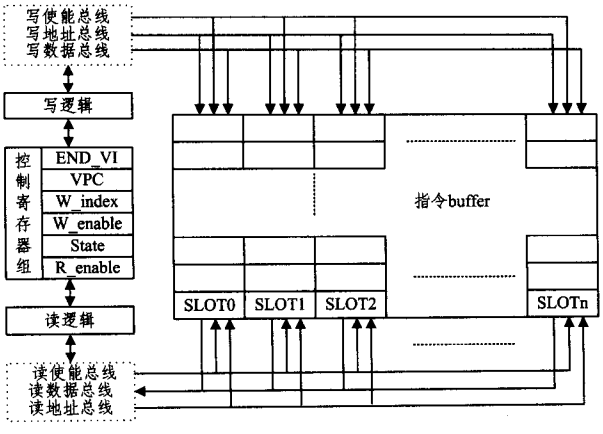

图 3　循环指令缓冲的读写机制

3.1　写循环指令缓冲机制

循环指令缓冲的写过程由标量单元通过配置写使能寄存器完成写过程的触发，在写使能有效的情况下，由控制逻辑将来自派发部件的循环体指令同步写入到循环指令缓冲的写指针所指示的单元中。

对于每层循环体，在第一次执行完毕时，该层循环体以内包括的循环体指令都已经全部写入指令缓冲中，此时程序员可以根据以该层循环为最外层的循环体执行时间来决定是否开始执行多线程模式。如果运行时间较长，则以该层循环为最外层循环开始执行多线程模式，否则仍然采用单线程模式完成后续循环次数的执行。对于最外层循环，在循环体指令全部写入指令缓冲后，开始执行多线程模式。

对单层循环而言，在循环体的第一次执行过程中将指令写入指令缓冲，而对于嵌套循环的情况，外层循环体的第一次执行包含了内层循环体的多次执行，这样就需要避免内层循环体多次执行时对指令缓冲的多次写入。为了支持嵌套循环体的写入，在线程切换装置的专用寄存器中添加一个上一指令包 PC 值寄存器，记录上一个写入指令缓冲的执行包 PC 值，每当写入来自派发部件的指令包时，都要判断该指令包的 PC 值是否大于上一指令包 PC 值，如果大于，则完成写入，同时更新上一指令包 PC 值为当前写入指令包的 PC 值。否则不予写入。这样的机制可以保证嵌套循环的内重循环体只有第一次运行时才写入指令缓冲，避免了内重循环体的重复写入。

当有多个循环体需要写入循环指令缓冲时，采取的写入机制是，替换策略，即循环体永久都是从循环指令缓冲的首地址开始写入。采用这种机制的好处是控制非常简单，缓冲的剩余空间大小永远都是 5.536kB。程序员可以方便地根据循环大小决定是否采用多线程模式配置。若采用 FIFO 的写入机制，则需要时刻检测循环指令缓冲的剩余空间大小，判断是否会发生写入溢出，大大增加了控制的复杂度，有些得不偿失，故采用替换的写入机制。采用替换方法，对于多层循环来讲，由于缓存大小固定，当完成目标循环指令的初始化设置后，可严格控制目标循环体的大小。若是实际的应用程序循环体大于指令缓冲大小(尽管可能性很小)，可采取拆分循环体的方式解决。执行目标循环体完毕后向标量单元发送中断信号，标量单元根据中断信号切换到单线程模式，不会产生抖动。

当循环体写到某层循环结束地址时，通过提前配置好的指令，将写缓冲有效切换成读缓冲有效，至本层循环执行完成后，硬件会自动完成读到写的切换，即多线程模式向单线程模式的切换，然后继续写入剩下的循环指令，如此反复直到整个循环体全部写入缓冲，此时写循环指令缓冲工作完成。

3.2　读循环指令缓冲机制

循环指令缓冲的读过程由标量单元通过执行 SWITCH 指令完成写使能的取消与读使能的配置。在读使能有效的情况下，控制逻辑将指令缓冲的读指针所指向的存储单元内的指令取出并发送到向量单元进行处理。同时根据各层循环的起止地址和次数在寄存器中的信息以及支持各层循环次数固定的多重循环，在相应的循环次数执行完毕后，由指令缓冲的读逻辑向标量单元发送中断，使执行模式重新变为单线程模式。

多重循环的控制过程：根据当前循环层指针的指示，在从指令缓冲中读出指令的同时，比较当前读指针与循环层指针

所指的循环尾地址寄存器中的值是否一致，如果一致，则修改相应循环层次数寄存器的值，当寄存器的值不为 0 时，将指令缓冲的读指针修改为对应于该层循环的初始(首)地址寄存器的值。相当于完成了程序的跳转。当寄存器的值为 0 时，判断当前循环层数是否为最外层循环，如果不是则改变当前循环层指针的值，并继续进行指令缓冲读指针的循环递增过程。如果已经是最外层循环，则向标量单元发送中断，同时修改向量单元的状态寄存器为空闲。但这里要特别注意一点，即由于写入循环缓冲时，循环体已经执行过一遍，因此在读开始时当前循环层的循环次数要在原来的基础上减 1，当进入下一层循环时，又要恢复最先配置的循环次数。

3.3 线程模式切换机制

线程模式的切换既是难点也是重点。在本文结构中单线程模式向多线程模式的切换是通过配置指令完成的，多线程模式向单线程模式的切换是硬件自动完成的。在单线程的模式下完成循环执行的初始化操作以及循环体的第一次执行，并将循环体写入指令缓冲部件，此时指令缓冲仅进行写入操作没有读出操作。标量单元对循环体进行相关配置后，选择来自指令缓冲部件的指令发送到向量单元，完成单线程到多线程的切换，此过程为软件切换。向量单元从指令缓冲处获取循环体指令，进行循环处理，标量单元仍然从指令派发部件获取标量指令(此时，指令派发部件仅派发标量指令)。标量单元与向量单元执行不相关的指令流。当目标循环体执行完毕后，执行模式转换为单线程继续执行，此时缓冲模块检测到目标循环体，执行完毕，状态机自动跳转到结束状态，读使能信号无效，完成多线程向单线程的切换过程。此过程为硬件切换。指令缓冲停止派发工作，而派发部件继续为标量单元与向量单元发射指令。在模式切换的时候，为了支持软件流水，采用了旁路技术来保证无缝切换。

在模式切换中，若单线程向多线程模式的切换也采用自动切换，则控制难度较大，且在时序上难以满足。此过程的切换开销主要看标量配置指令占整个循环体指令的比重。在本设计中，当写入每层循环后立刻进行读需要较大开销。也就是说频繁地在单线程与多线程之间进行切换代价开销较大。而当把循环体全部写入后再进行读，开销较小。多线程模式向单线程模式的切换采用自动切换，即循环体的一部分或者全部执行完后，双指令流模式自动切换为单指令流模式，是基于切换开销的考虑。因为通过指令配置完成则需要发出模式切换的中断信号，由于中断开销较大，故采用硬件自动切换模式。为了更加直观地了解多线程模式的执行过程以及循环缓冲的读写流程，本文给出二重循环指令缓冲的读写流程图，如图 4 所示。本设计中的循环指令缓冲执行支持三重嵌套循环，一重循环和三重循环的读写机制与二重循环读写机制相同。

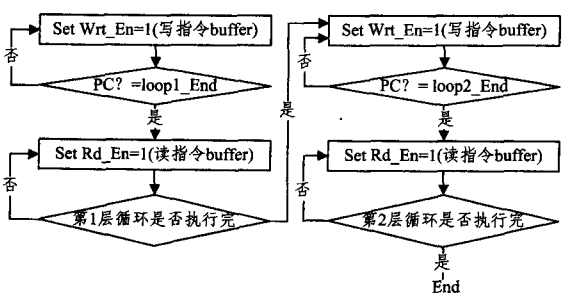

图 4 二重循环指令缓冲读写流程图

4 性能评估

4.1 功能验证

在 Modelsim 和 NC Verilog 模拟环境下对 RTL 级代码分别进行了模块级与系统级验证，主要验证其逻辑功能的正确性。验证结果显示，设计逻辑和功能均符合要求。图 5 示出了多线程模式下的读写循环缓冲过程及单线程与多线程模式切换过程的波形。

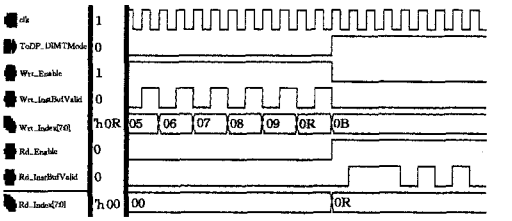

图 5 NC 仿真验证模拟波形图

其中 ToDP_DIMTMode 信号指示执行模式。当信号为低电平时表示单线程模式，高电平表示多线程模式。Wrt_Enable 表示写指令缓冲使能信号，Rd_Enable 表示读指令缓冲使能信号。Wrt_InstBufValid 表示写循环指令缓冲有效信号，Rd_InstBufValid 表示读循环指令缓冲有效信号。写使能为高时，写指令缓冲有效，读使能为高时，读指令缓冲有效。从图中可以看出写使能信号无效时，读使能信号立即有效，从而实现了从单线程到多线程的无缝切换。

4.2 性能分析

基于 65nm 标准单元库采用 Synopsys 公司的 Design Complier(DC)工具对 RTL 级代码进行综合优化。综合结果如表 1 所列，在保证功能正确的同时使性能提高，面积(um)^2 和功耗(mw)减少。

表 1 DC 综合结果

参数	派发	指令 Buffer	CPU	Buffer/CPU
面积	448104.1	65599.5	15927991.7	0.41%
功耗	24.1	4.4	1518.6	0.29%

由表 1 可以看出，在单线程模式下，直接由派发而得出的面积是 448104.1，功耗为 24.1，所占 CPU 比例为 0.41%。而在多线程模式下，循环指令缓冲中面积只需要 65599.5，功耗为 4.4，所占 CPU 比例为 0.29%。多线程模式下开销所占 CPU 的比例是很小的。

点积算法程序是一个两层循环嵌套程序，矩阵乘算法程序为一个一层循环嵌套程序，针对该两个算法循环程序分别在两种不同的模式下执行 10 次。程序执行到第 3 次时，基本已达到稳定。本文仅记录了其前 3 次结果作为依据，得到点积算法程序 IPC 分析结果，如表 2 所列，矩阵乘算法程序 IPC 结果如表 3 所列。结果以 IPC 作为评估参数。

表 2 点积算法程序 IPC 分析结果

次数	单线程(ns)	多线程(ns)	Δ	百分比(%)
1	2388.0	2168.0	220.0	9.21
2	1886.0	1678.0	208.0	11.03
3	1886.0	1678.0	208.0	11.03

从表 2 看出，第一次由于发生取指缺失等原因，导致所耗时间不稳定，第 2 次以后循环程序执行时间都稳定，从而可以得出结论：多线程模式下，派发效率提高 11.03%。

表 3　矩阵乘算法程序 IPC 分析结果

次数	单线程(ns)	多线程(ns)	Δ	百分比↑(%)
1	26710.0	16216.0	10494.0	39.29
2	728.0	668.0	60.0	8.24
3	728.0	668.0	60.0	8.24

从表 3 可以得出，当程序派发执行稳定后，在多线程模式下的派发执行效率比单线程模式下提高了 8.24%，多线程模式下 IPC 的性能提升了 8.24%。

结束语　本文详细介绍了多线程模式下循环指令缓冲的读写机制以及单线程与多线程的切换机制。使用 NC Verilog 对 RTL 级代码进行了系统级仿真验证，用 65nm 标准单元库对 RTL 级代码进行了 DC 综合，在保证了功能正确性的同时，综合后的运行频率可达 714.286MHz，通过程序验证了在多线程模式下，处理器在性能上得到了提升。

参 考 文 献

[1] Merten M C, et al. Modulo Scheduling Buffer[C]// Proceeding of the 34th annual ACM international symposium on Microarchitecture. Texas, Dec. 2001

[2] Hennessy J L, Patterson D A. Computer Architecture: A Quantitative approach(4th Edition)[M]. Elsevier, 2007: 274-363

[3] Ramakrishna B, Schlansker M S, Tirumalai P. Code Generation Schema for Modulo Scheduled Loops[C]// Proceeding of MICRO-25. 1992: 297-326

[4] Grunewald W, Ungerer T. Towards Extremely Fast Context Switching in a Block Multithreaded Processor[C]// Proceedings of the 22nd EUROMICRO Conference. September 1996

[5] Gwennap L. MAJC Gives VLIW a New Twist[J]. Microprocessor Report, 1999, 13(12)

[6] Aa T V, Jayapala M, Barat F, et al. Instruction Buffer Exploration for Low Energy VLIWs with Instruction Clusters[C]// Proceedings of the 2004 conference on Asia South Pacific design automation. Yokohama, Japan, Jan. 2004: 163-247

[7] Tullsen T D, Eggers S, Emer J, et al. Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor[C]// Proceedings of the 23rd Annual International Symposium on Computer Architecture. Philadelphia, May 1996

(上接第 40 页)

适的条目数量。

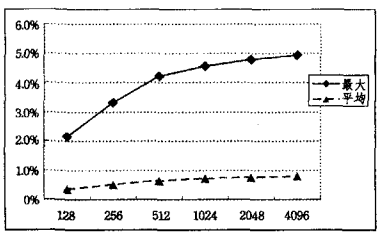

图 5　不同条目数量下性能变化趋势

以 4096 条目数量的性能提升作为参考值，将其他数量的提升情况与 4096 情况进行对比。图 6 详细给出了部分 spec2000 课题在不同 DPT 条目数量下的性能提升对比情况。当条目数量为 1024 时，性能提升约为 4096 时的 80%；当条目数量为 512 时，降为 4096 的 60%左右。4096 情况比 1024 硬件开销增加 4 倍，但性能提升只增加 20%。综合考虑，在实际使用中，1024 条目数量是一种合适的选择。

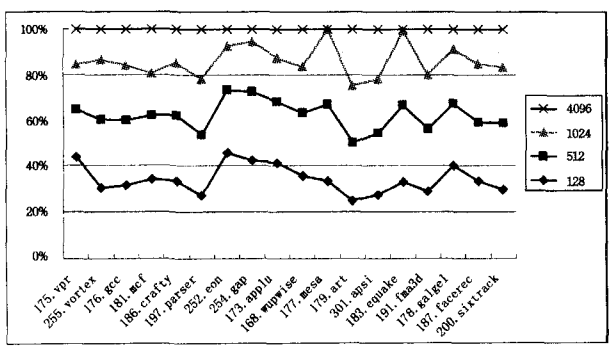

图 6　spec2000 各课题在不同条目数量下的性能提升对比

结束语　本文设计了一种简单、高效的存储相关性预测器 SMDP。SMDP 易于实现，硬件开销小；能够根据程序运行动态情况自适应更新预测信息表；能够充分利用指令级并行性，使得顺序发射的 load 指令在保证正确率的前提下，能尽早乱序发射。本文详细介绍了 SMDP 的预测及处理过程、预测信息位更新策略，并分析了 SMDP 的实现难度与开销。实验表明，SMDP 能够有效提升处理器性能，在贴近实际处理器参数配置情况下，平均性能提升达到 0.7991%，最大可达 4.9225%。

参 考 文 献

[1] Doweck J. Inside Intel® Core™ Microarchitecture and Smart Memory Access[M]. Intel White Paper, 2006

[2] Moshovos A, Breach S E, Vijaykumar T N, et al. Dynamic Speculation and Synchronization of Data Dependences[C]// the Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA). 1997

[3] Chrysos G Z, Emer J S. Memory dependence predictor using store-sets[C]// Proceedings of the 25th Annual International Symposium on Computer Architecture (ISCA). 1998

[4] Loh S S G H. Store Vectors for Scalable Memory Dependence Prediction and Scheduling[C]// the Proceedings of the 12th International Symposium on High-Performance Computer Architecture(HPCA). 2006

[5] Onder S, Gupta R. Dynamic memory disambiguation in the presence of out-of-order store issuing[C]// Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture (MICRO). 1999

[6] Yoaz A, Erez M, Ronen R, et al. Speculation Techniques for Improving Load Related Instruction Scheduling[C]// Proceedings of the 26th Annual International Symposium on Computer Architecture (ISCA). 1998

[7] Kessler R E. The Alpha 21264 Microprocessor Architecture[C]// IEEE MICRO. 1999