

一种 VLIW 循环指令的预取和优化策略

琚 魁, 谢 憬, 毛志刚

(上海交通大学 微电子学院, 上海 200030)

摘 要: 本文提出了一种 VLIW 处理器的预取和针对循环指令的优化策略. 文中重点介绍了预取普通指令和处理循环指令的方法, 以及普通预取和循环预取这两种预取模式间的切换方式. 基于该设计和优化方案, 可以有效减小取指操作的功耗. 实验证明, 在针对不同的应用上, 减少的功耗从 40% 到 90% 不等, 优化了该 VLIW 多运算簇 DSP 处理器的性能.

关键词: DSP 处理器; VLIW; SIMD; 指令预取

中图分类号: TN402

文献标识码: A

文章编号: 1000-7180(2013)05-0019-04

Design and Optimization of a VLIW Loop Instruction Prefetch Structure

JU Kui, XIE Jing, MAO Zhi-gang

(School of Microelectronics, Shanghai Jiaotong University, Shanghai 200030, China)

Abstract: In this paper, we present a novel design of a structure for VLIW instruction prefetching and separating loop instruction from the other program. With this method, we can reduce the power consuming in fetching instruction, which range from 40% to 90% for different applications, and improve the performance of the cluster based DSP processor.

Key words: DSP processor; VLIW; SIMD; instruction prefetch

1 引言

现代的高性能微处理器通常都包括了取指模块和指令执行模块. 取指模块主要负责从指令储存器中读出指令并进行译码, 而指令执行模块则负责指令所表示的操作的执行以及相关操作数据的更新和保存^[1]. 在现代的处理器中, 取指已成为功耗的重要来源, 而在一些高性能计算处理器中 (如 DSP 芯片), 这一部分的功耗尤为明显. 现在较为流行的高性能计算处理器多采用超长指令字结构, 由于指令位数的增加, 这更加剧了取指功耗的问题^[2].

在嵌入式处理器中, 取指操作一般消耗大约 50% 的功耗, 而在 DSP 应用中, 这部分功耗又主要消耗在循环指令上. 在 DSP 应用中, 以循环形式出现的计算指令占了指令的大部分, 重复不断的从指

令储存单元中取址造成了巨大的功耗浪费.

针对上述问题, 通常采用预取指令的方式来解决该问题. 通用处理器采用的预取方式主要有 Next-N-line 预取, Wrong-Path 预取和 Markov 预取等, 而对于 DSP 处理器来说, 由于其指令具有并行性高, 可重用性好等特点, 主要采用的是 loop-based 的指令预取方式.

本文提出了一种针对采用流处理方式进行计算的 DSP 处理器的预取及循环指令一体化处理的结构, 可以有效减少处理器取指部分的功耗, 进而提升处理器的性能.

2 预取结构的目标和适用环境

2.1 指令预取的目标

在通用的嵌入式处理器中, 通常会用到很多的

跳转指令和分支指令,这给指令的预取带来了很多问题,同时也产生了很多针对不同应用需要的预取方式^[3].这些预取方式都有着相同的目标,即尽量保证预取的准确性.而所有的预取方法都应该关注三个问题,即:①什么时候开始预取;②取哪一行或哪些行;③什么时候去用预取的指令.从预取的角度来看,程序指令的顺序执行程度越高,则预取的方式就可以越简单,而其表现也就越好.

2.2 预取结构所在的 DSP 处理器结构

对于高性能 DSP 处理器来说,其主要是面向高密度计算应用的,因此其指令主要由顺序和循环两种形式出现,而其处理分支的方式是采用所有分支均并行执行,只在输出结果时进行判断.这给我们把预取和循环指令缓存结合起来提供了可行性.

我们采用主控核和计算核结合的方式来实现一种高性能 DSP 处理器(图 1).其中主控核主要完成计算核数据的输入输出流的控制,以及遇到分支时判断条件的执行从而选择计算核结果输出的功能,而计算核只负责进行高密度的计算.计算核采用 16 个计算簇的形式构成,并使用 VLIW 的 SIMD 的方式进行工作.每个运算簇上由 4 个加法器和两个乘法器,数据交换单元和输入输出单元构成.计算核采用和主控核不同的指令体系,即 VLIW 来控制计算簇的运行.对于这样的处理器结构来说,所得到的计算核的 VLIW 指令的构成就是顺序指令和循环指令这两类指令,因此只要将这两类指令分开,就可以实现顺序指令部分预取的 100% 准确率,同时完成循环指令的缓存,减少了循环指令重复取指的功耗.

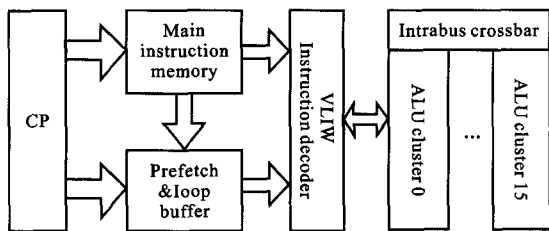


图 1 DSP 处理器总体架构

由于该 DSP 处理器的计算核采用 SIMD 的指令结构,其一条指令将会被群发至 16 个运算簇中,16 个运算簇执行相同的运算.指令的长度为 288 位,其中 11 位用来进行循环指令的控制,其余位用来控制 16 个簇上的功能单元.

这 11 位主要由三部分构成,如图 2 所示.

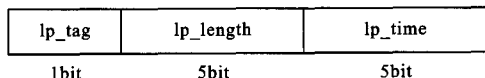


图 2 VLIW 指令循环控制部分

3 预取结构

根据以上的分析,适当的预取方法能在很大程度上节省取指单元的功耗.对于 DSP 处理器来说,循环指令占指令的大部分,因此如果我们可以把普通指令的预取和循环指令的处理结合起来,就可以最大程度的节省 DSP 应用的取指功耗和性能.以下我们将详细介绍一种将预取和循环指令优化结合在一起的取指优化方式.

3.1 预取单元的整体结构

预取单元主要由控制模块和缓存单元构成.预取控制模块(图 3)在主控核控制下的工作流程如图 4 所示.

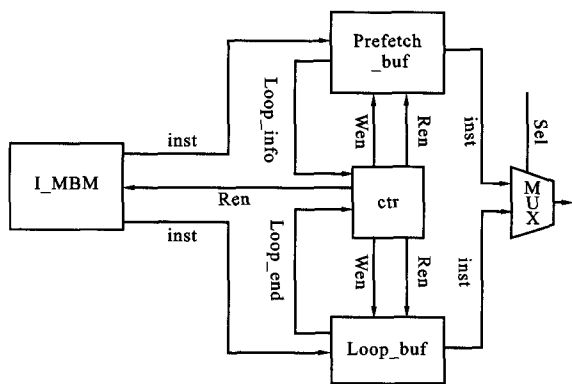


图 3 预取模块整体结构

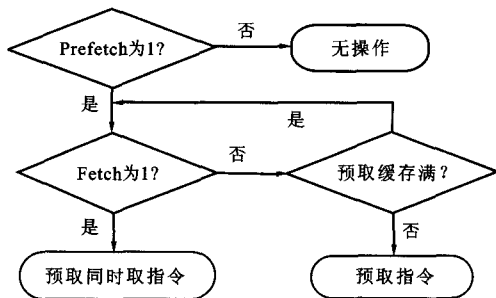


图 4 预取流程图

3.2 循环指令的预取

目前的结构主要支持简单的单层循环,即图 5 伪代码所示的循环结构.循环缓存主要对 for 循环内部代码转换而成的指令进行处理.这些指令中 $A[i]$, $B[i]$, $F[i]$ 均为一些无相关性的数据,可以并行处理.

```
//some sequential code
for(i=0; i<100; i++){
  C[i] = A[i] + B[i];
  D[i] = A[i] * B[i];
  E[i] = C[i] + D[i];
  G[i] = E[i] + F[i];
}
```

图 5 指令伪代码

对于循环指令的重用能够有效地减少取指的功耗^[4],经常采用的重用循环指令的方法有 cache 和 buffer 两种^[5],根据本设计的整体外部环境综合考虑,由于我们采用顺序的指令预取方式,不需要复杂的替换策略,因此这里采用 buffer 的做法.通过对一些常用应用的考察,将 buffer 的深度设定为 1024,其大小为 9KB.本设计中,buffer 的工作有以下几种状态(表 1).

表 1 循环状态

循环状态寄存器	描述
LOOP_ONCE	Loop once
LOOP_BEGIN	One loop begins
LOOP_CONFLICT1	Another loop before one loop over
TWO_2_ONE	Two loop to one loop
LOOP_CONFLICT2	Another loop before one loop over on its end
DEFAULT	Other situations

与循环指令预取相关的寄存器有以下四个:looptime,lp_length,lp_time_reg,lp_length_reg.其中 looptime 和 lp_length 用来储存当前循环体的循环信息,lp_time_reg 和 lp_length_reg 用来在产生冲突时存储后面一个循环的相关信息.

在表 1 所表示的几种状态中,LOOP_ONCE 表示当前循环体被执行完一次,对应的寄存器 looptime 减 1;LOOP_BEGIN 表示一次循环的开始,此时将从 VLIW 中得到的循环信息赋值给 looptime 和 lp_length 寄存器;LOOP_CONFLICT1 表示当前的循环体尚未被执行完成时,后一个循环体又被检测到,此时将从 VLIW 中得到的循环信息赋值给 lp_time_reg 和 lp_length_reg 寄存器;TWO_2_ONE 表示在 lp_buf 的等待队列中存在两个循环体时,有一个完成时的状态,此时将保存在 lp_time_reg 和 lp_length_reg 寄存器的循环信息分别赋值给 lp_time 和 lp_length 寄存器,开始新的循环体指令获取;LOOP_CONFLICT2 表示和 LOOP_CONFLICT1 近似的状态,只是在这种状态下,下一个循环体正好出现在当前循环体的某一个单次执行完成处,因此多了一些寄存器操作.

3.3 循环指令和普通指令模式的切换

在整个计算核开始工作之前,普通指令预取单元先开始从指令主存中预取指令.

在计算核开始工作后,普通指令预取单元会检测预取到 VLIW 指令的 lp_tag 位,一旦为 1,则其后的指令将会转由循环指令预取单元进行预取.对于指令主存中拿取指令的切换关系如图 6 所示.

图 6 中的 loop_end 信号是由一系列信号经过组合逻辑运算获得,其表示循环状态结束.而译码单元将会从普通预取和循环预取的缓存中选择一个作为得到的指令.而作为代价,编译器需要在每次循环体开始前一条指令自动插入循环的标志和相关信息.

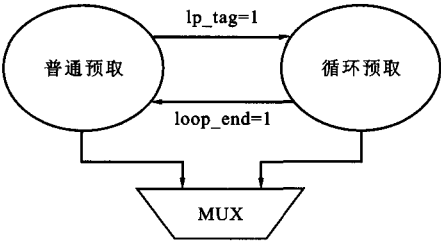


图 6 普通和循环预取状态切换

4 实验结果和分析

设计采用了 SMIC0.13um 工艺库,使用 synopsys 的 Design Compiler 和 Astro 工具对预取模块和译码模块进行综合和后端设计,同时针对一些典型的 DSP 应用使用 PTPX 进行功耗分析,与不采用该预取结构的设计进行比较.

我们分别对未优化的结构和优化后的结构进行了功耗测试.图 7 是针对不同应用时两种结构的存储单元功耗比较.图 7 中共有 100 对 4×4 矩阵进行乘累加,每次执行四对矩阵的运算;向量为 16 长度的 32 位整形数据.表 2 是面积的增加与 memory 容量的变化.

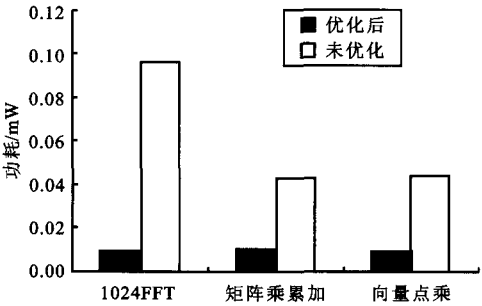


图 7 不同应用下的存储单元功耗比较

表 2 面积消耗与 memory 容量

	面积(μm^2)	Mem(KB)
优化前	6132547	130
优化后	6669924	135

可以看出,我们以多花费 8%左右系统面积和不到 4%的存储单元为代价,获得了减少 50%—90%取指功耗的效果.优化后的结构对 DSP 应用中的指令存储单元的功耗有显著的减少.

5 结束语

本文提出了一种针对 DSP 处理器的指令预取策略,其中针对 DSP 应用的特点又对循环指令采用了特殊的预取策略,提高了 DSP 应用的执行效率,减少了指令储存单元的功耗,优化了 DSP 处理器的性能。

参考文献:

- [1] 沈立,戴葵,王志英.以基本块为单位的非顺序指令预取[J].计算机工程与科学,2003,25(4):94-99.
- [2] Zhenqi Wei, Peilin Liu, Ji Kong, et al. Low-power microarchitecture of zero-overhead nested loops in embedded processors[C]// International Symposium on Intelligent Signal Processing and Communication, China, Chengdu, 2010.
- [3] Yiqiang Ding, Wei Zhang. Loop-based instruction pre-fetching to reduce the worst-case execution time[J].

IEEE transactions on computers, 2010, 59(6): 855-864.

- [4] Jian Wang, Bogong Su, Erh Wen Hu. A scalable loop optimization approach for scalable DSP processors[C]// ICASSP '00 Proceedings of the Acoustics, Speech, and Signal Processing, 2000. on IEEE International Conference USA: Salt Lake City, 2000.
- [5] Vladimír Guzmán, Teemu Pitkäräinen, Jarmo Takala. instruction buffer with limited control flow and loop nest support[C]// International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, Lebanon, Beirut, IC-Samos, 2011.

作者简介:

琚 魁 硕士研究生.研究方向为大规模集成电路设计.

谢 憬 博士研究生.研究方向为大规模集成电路设计.

毛志刚 博士生导师,教授.研究方向为大规模集成电路设计.

(上接第 18 页)

- [7] Kuon I, Rose J. Automated transistor sizing for FPGA architecture exploration[C]// Proceedings of the 45th annual conference on Design automation. New York, NY, USA: ACM, 2008: 792 - 795.
- [8] Zhao W, Cao Y. New generation of predictive technology model for sub-45 nm early design exploration[J]. IEEE Transactions on Electron Devices, 2006, 53(11): 2816 - 2823.
- [9] Apache. Xerces-c[EB/OL]. (2010-03-07)[2012-05-16]. <http://xerces.apache.org/xerces-c/>.
- [10] Kuon I, Rose J. Area and delay trade-offs in the circuit and architecture design of FPGAs[C]// Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays. New York, NY, USA: ACM, 2008: 149 - 158.
- [11] Mutlu A, Yamacli S. An improved elmore delay model for VLSI interconnects[J]. Mathematical and Computer Modelling, 2010, 51(7/8): 908-914.

- [12] Sheldon D, Vahid F. Making good points: application-specific pareto-point generation for design space exploration using statistical methods[C]// Proceedings of the 17th ACM SIGDA International Symposium on Field-Programmable Gate Arrays. Monterey, CA, USA: ACM, 2009: 123-132.
- [13] Xiaowei Han, Lihua Wu, Yan Zhao, et al. A radiation-hardened SOI-based FPGA[J]. Journal of Semiconductors, 2011, 32(7): 075012-6.

作者简介:



张 峰 男, (1987—), 博士研究生. 研究方向为 FPGA-EDA 工具的设计与开发.