

# 流体系结构抽象模型研究<sup>1</sup>

文梅 伍楠 张春元 李海燕 李礼

(国防科学技术大学计算机系 长沙 410073)

(wenmei8086@163.com)

**摘 要** 随着 VLSI 技术以及应用的发展, 现行的高性能处理器体系结构面临挑战。流应用以其重要性和覆盖领域的广泛性使得流体系结构在新兴的体系结构中备受关注。鉴于多种多样的流体系结构的存在, 本文提出了一个流体系结构的抽象模型, 包括硬件结构模型、程序设计模型和面向流体系结构的编译, 基于此在流体系结构模型上映射了一个流应用实例, 说明流体系结构的有效性。本项工作对开发应用在多种流体系结构上的可移植性具有重要意义。

**关键词** 流体系结构; 硬件结构模型; 程序设计模型; 流应用

**中图法分类号** TP332

## Research of Abstract Stream Architecture

WEN Mei, WU Nan, ZHANG Chunyuan, LI Haiyan, LI Li

(Computer School, National University of Defense Technology, Changsha, Hunan, 410073)

**Abstract** With the development of VLSI technology and applications, architecture of high performance processor is facing challenge. Stream architecture is outstanding in emerging architectures by stream applications' significance and universality. As many kinds of stream architecture exist, this paper presents an abstract stream architecture model, including hardware structure model, programming model and compiler. Based on the model, we map a stream application on the stream architecture model. Experience data shows that stream architecture is effective. The goal of this work is to improve interoperability of application on different stream platform.

**Keywords** stream architecture; hardware structure model; programming model; stream application

## 1 引 言

随着 VLSI 技术以及应用的发展, 现有高性能处理器体系结构面临一些重大难题:

处理器上集成的晶体管数目急剧膨胀, 随之而来的长线延迟问题日趋严重, 全局结构高速片内通讯难以实现;

尽管运算单元成本降低, 但集成效率不高, 难于获得足够的指令级并行;

设计专用化, 难以适应逐渐多样化的应用需求。市场需要灵活的可编程高性能处理器。

预计表明, 在处理器设计上如果没有创新, 微处理器性能将仅仅随制造工艺的速度增长, 约为每年 20%~25%。为了使微处理器性能继续持续每年 50%提升的速度甚至更高, 国际上正积极开展新型体系结构的研究, 涌现出 Imagine、RAW、Viram、TRIPS<sup>[1,2]</sup>流

处理器和 Score<sup>[1]</sup>流计算模型等多种新兴体系结构, 由于这些体系结构拥有面向计算密集型应用, 处理大量数据流的共同特点, 因此被统称为流体系结构, 但它们之间又有着明显的区别。Imagine 是向量技术的扩展; RAW 是 Tile 结构, 支持 MIMD; TRIPS 支持多态工作负载; Viram 采用向量流水线加 PIM 技术; Score 采用可重构机制。多种流体系结构必然造成各自流语言及编译器的开发, 已有的流语言及程序设计模型包括 StreamC/KernelC<sup>[3]</sup>、StreamIt、Brook、Cg<sup>[4]</sup>等, 以及应用在各平台间移植的困难。Labbonte 等提出了面向流虚拟机 (SVM) 的两级编译的方法<sup>[5]</sup>, 在一定程度上降低了流程序在不同流体系结构上编译的复杂度, 但 SVM 的体系结构模型覆盖面过于广泛, 反而弱化了流体系结构的特征。

本文提出了一个抽象的流体系结构模型, 具体安排如下: 第二部分说明硬件结构模型, 第三部分说明流语言及程序设计

<sup>1</sup> 基金项目: 国家自然科学基金 (60473080)、973 计划 (5131202)

模型，第四部分描述基于流体系结构模型的流编译，第五部分在流体系结构模型上映射了 Reed-solomon 解码应用，最后总结全文。本项工作对开发应用在多种流体系结构上的可移植性和深入理解流体系结构本质具有重要意义。

## 2 硬件结构模型

所谓流，就是不间断的、连续的、移动的数据队列。在众多的面向流的处理器之上，本文提取出一种抽象的流体系结构模型。该模型涵盖了流体系结构的本质的思想及其特征。

基于数据组织成流的思想，流体系结构的组成有别于传统的 von Neumann 结构——不仅分离了数据和指令，对数据访问、组织指令和运算操作指令的执行模块也进行了解耦合。整个流体系结构的抽象模型如图 1：

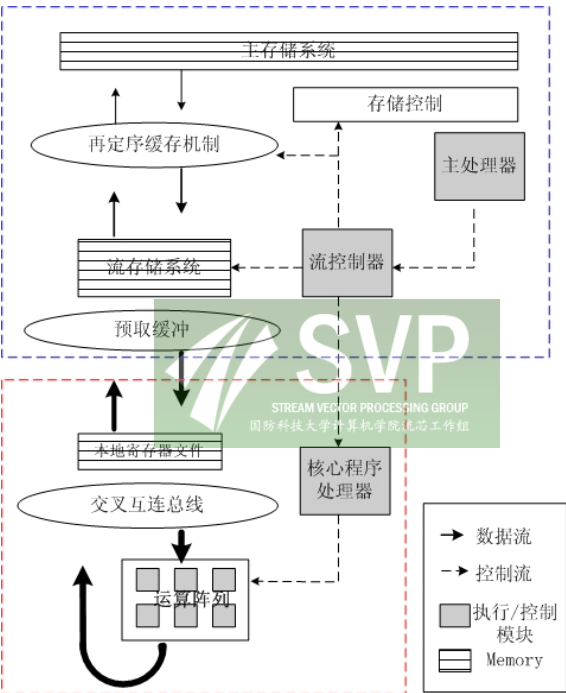


图1 抽象的流体系结构模型

基于流体系结构的处理器分为两大子系统：流的调度模块和流的计算模块。图1中上面方框为流的调度模块，下面方框为流的计算模块。

流的计算模块所能处理的数据和执行的指令都以流的形式加载。从排队论模型看，流体系结构中的数据和指令可以看作消费者和生产者之间的带索引的FIFO队列。计算模

块以分空间或分时的形式既充当消费者也充当生产者。当队列中质点的流出率达到较高值且点间距分布平稳时，组织成流的数据和指令能极大的提高运算和操作的效率。然而自然程序中的数据和指令不一定是按流的方式组织的，其在存储器中的分布很可能是无序的或者不是完整的流。模型中必须有一个转换或是标准化的过程，即流的组织调度，通常这个过程是同流的加载一起进行的。

从功能需求上看，流应用属于密集型计算，其计算操作的形式多样而复杂，对指令和数据的流出要求有极高的吞吐率，但是执行模式通常较为简单。相对而言，流的加载和组织的操作模式上较为简单，对指令的吞吐率要求也不高，但要求功能灵活而复杂。将两者紧密耦合在一起降低了机器的执行效率。

因此在流体系结构模型中，流的计算模块只负责密集的流上的计算并以极高吞吐率执行核心级指令。流的组织调度交由专门的控制模块，负责向计算模块提供数据流和指令流。

流的计算模块专注于密集计算，执行底层的仅包含运算信息核心级指令。通过大规模的计算阵列或是多功能功能单元获取极高的计算能力。图2列出了几种新型流处理器原型系统中计算模块达到的GOPS。

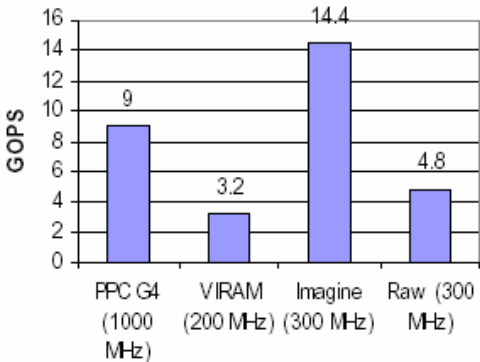


图2 几种流处理器实际达到的计算性能<sup>[6]</sup>

流的调度模块解决两个问题：流组织调度和带宽。数据通过索引和复杂的再定序机制在顶层流指令的控制下被组织成流（指令具有天然的流特性）。组织成流的数据排列规整对通信带宽要求较小，可以在运算阵列上以SIMD或MIMD的方式并行。由于分离了数

据的执行和load/store，流体系结构模型下的带宽结构是层次化的，充分开发局域性，配合预取和延迟隐藏流体系结构可以有效实现高速带宽通道。以基本符合流体系结构模型的处理器Imagine为例，其带宽层次明显，有很高的有效带宽（有效带宽指越靠近计算带宽越高），最内层的峰值达到544GB/s, 实际程序测试时的平均带宽也超过200GB/s, 如图3所示。

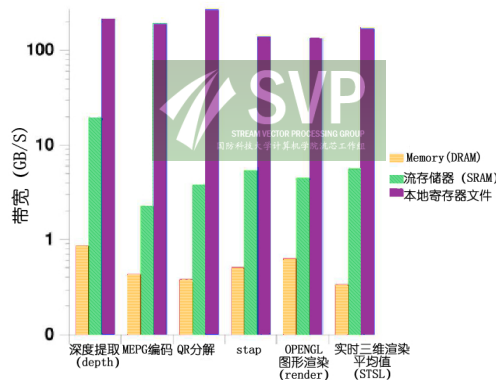


图3 应用在Imagine实现的带宽层次<sup>2</sup>

3 程序语言及程序设计模型

现在已有或正在开发的基于流的语言很多,包括StreamIt、Brook、StreamC/KernelC等。在这些流语言的的基础上，结合流体系结构的硬件结构模型，我们抽象出流程序语言及设计模式的基本模型，与传统的通用语言有较大差异，见表1。

流体系结构的编程模式应是一种层次化的编程模式。本节以我们在流语言StreamC/kernelC上实现的Reed-Solomon(RS)解码过程为例说明，如图4。首先是确定顶层应用的功能和输入输出流。接着是流级程序设计，对计算过程进行分解，由程序员将一个应用分解成一系列的计算核心，产生结构化的数据流图，形成明确的生产者-消费者模型。基本上是根据功能来划分核心，每个核心有明确的输入流和输出流，例如我们将RS解码过程分为图4示的4个核心程序。流级程序对应第二节中的流调度模块，在主处理器和流控制器上执行。分解的一个个核心程序（kernel或filter），采用专门的核心级语言描

表1 流语言与传统非流语言比较

传统的非流语言	流语言
描述能力强，覆盖所有应用	作为通用语言的扩展子集，对流应用有极高的描述能力和效率
以数组或指针描述类似于流的数据	可以直接描述各种流(非/变长，不/带索引，一维/多维)
单一层次	分层的程序设计模式
隐藏数据的访问和通信	数据流加载和通讯对程序员可见
可交叉调用、嵌套的函数和过程，数据流以图表示	简单的 Kernel 或 Filter，数据流结构化

述，负责对流的密集结算，对应第二节中的流计算模块，在核心程序处理器执行。简单的说，流程序就是将处理过程描述成一个个的流并按序流过各个kernel。

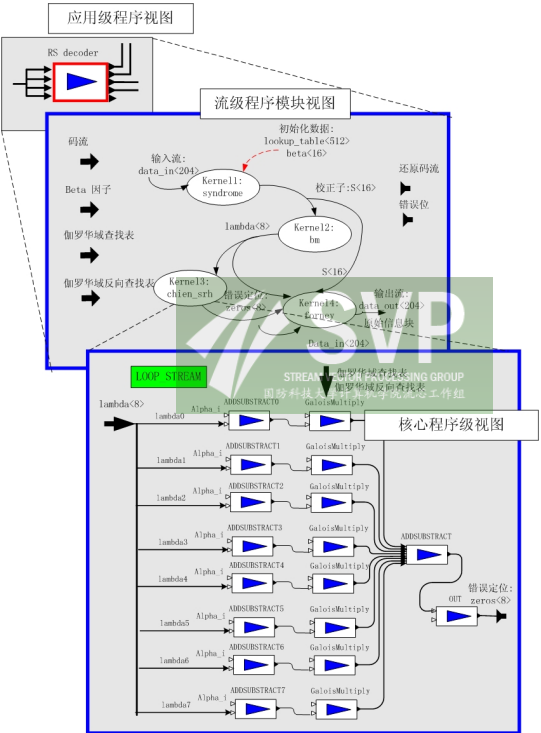


图4 RS解码在流体系结构上的映射

核心程序并不等于传统语言中的函数，如图5所示。传统的函数是非结构化的，可以互相嵌套、交叉应用。而核心程序是结构化的，由于其主要是处理密集的数据流且指令也以流的形式加载，不允许复杂的控制流存

<sup>2</sup>本文中有关 Imagine 的数据是在其时钟精确模拟器 ISIM (500MHz)上获得。



在。核心程序只有三种结构：流水、分叉或

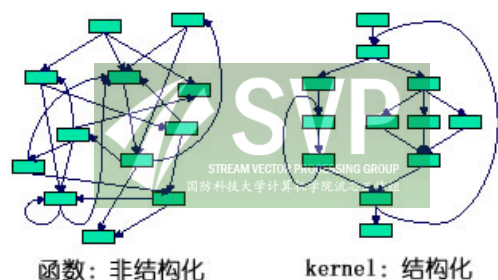


图5 函数与核心程序的比较

聚合、反馈环。这种简单的结构，对程序员来说更容易编写和分析，鲁棒性更强。当然增加了对编程的限制，对某些应用不适合。从机器和编译的角度，对控制流的限制使得数据和指令的执行更容易预测，局域性和并行性更强，适合高吞吐率的密集计算，同时也更有利于编译的调度和优化。

在流的程序设计模型中，流的访存和通讯对程序员是可见的，主要是为了减少对应的硬件结构的复杂性，以尽量减少长线延迟的影响。在流级程序设计时程序员需要考虑流的调入和组织，会带来一些问题比如短流，kernel或filter的划分等<sup>[7]</sup>。

## 4 面向流体系结构模型的编译

抽象流体系结构开放硬件资源以及层次化流程序执行模式给编译带来了新的要求。同时，流访问模式可知使得应用对编译完全透明，扩展了编译优化的空间。一个值得考虑的重要问题是分时或是分空间执行数据流图中的核心程序。分时执行的好处是无需考虑负载均衡，调度简单，缺点是如果出现短流，核心程序调入调出抖动，装载核心程序的开销大；由于输出流不能及时消耗因此对本地存储的容量要求高；延时较长，不适于实时性强的应用。分空间执行则正好与之相反。

编译要解决的另一个问题是数据的分块（blocking）调度。理想情况是数据流在计算核心程序之间匀速传递，且所有运算核心程序同时执行，数据流以被消费的速度产生，那么无须分块，核心程序一次可处理的数据流可以任意长。但由于硬件限制以及核心程序的计算量的不均衡，造成数据流分块分批处理的必要。由于流式数据之间的弱相关性，

分块较容易实现。

对应于层次化的编程模式，流编译也是层次化的。

顶层编译器结构如图6示，由若干阶段构成。首先语言预处理，转换顶层程序指令成流控制器可执行的操作形式；接着根据操作执行资源分配，包括几种资源：核心程序处理器里的程序存储单元、流存储系统、主存储系统、流控制器的相关控制寄存器，为每个核心程序分配程序存储空间，为每个输入输出流分配流存储空间，为每次访存操作分配主存空间，流控制器寄存器指定流地址和长度。

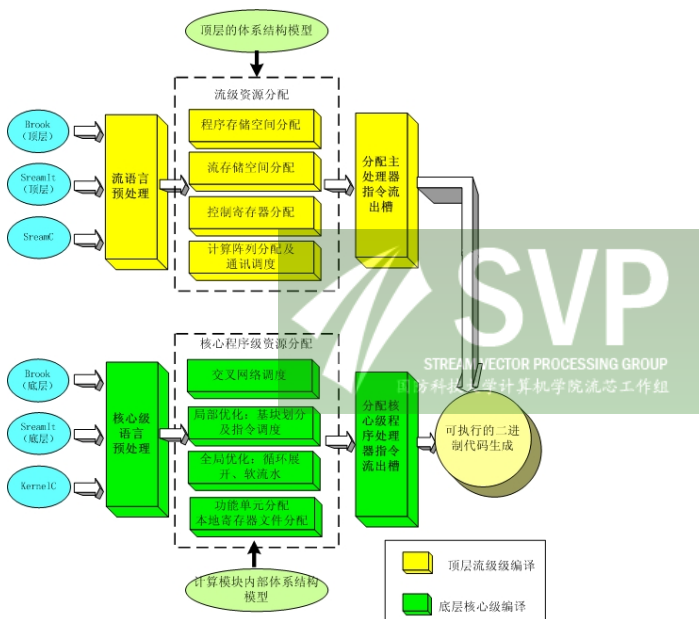


图6 两层流编译器

核心程序存储空间的分配尽量给核心程序分配不同的空间，如一次空间分配不能满足则优先替换不再需要执行的核心程序。流存储空间资源有限，分配策略应依据以下原则：首先满足当前执行核心程序的需要，同时考虑下一个执行核心程序的预取问题。流过长时，要采用double-buffer的策略<sup>[8]</sup>以减少等待开销。如将多个核心程序分空间执行还依据流图重新划分空间执行的核心块，并分配它们到相应的计算单元，以及计算单元之间的通讯调度。最后生成主处理器指令流出器的操作相关依赖图，分配流出槽派发操作给流控制器。在此基础上可做进一步优化，包括长流处理的分块分批操作，核心程序之间的软件流水，以最大限度利用硬件资源，隐藏

访存延迟。

底层编译器结构如图6示,也由若干阶段组成。首先预处理,转换指令为流计算模块可执行操作;接着分配计算模块资源,交叉网络调度,划分基本块,做操作相关分析,根据调度策略调度基本块;执行全局优化,由于核心程序主体都是循环,需要采用循环展开、软件流水等技术实现优化;最后将每一块内的操作在合适的Cycle分配给具体的功能单元,分配时应遵循可以在多个功能单元执行的单元尽量不分配给使用率高的功能单元原则,以及进行寄存器分配。

## 5 应用映射

适合流体系结构处理的应用我们统称流应用。基于抽象流体系结构模型,流应用应具有以下几个典型特征:具有自然的输入输出数据流,密集的计算需求、大量的数据并行性、鲜明的生产者-消费者局域性。我们对RS解码在流体系结构模型上进行了研究,RS解码是对制式码块进行解码并校正错误码元的过程,需解码码块和校正后的码块就是数据流的形式,码块之间无相关性,而且数量众多,可以大量开发数据并行。对于6Mbytes/sec的ADSL通道,仅RS解码要占用TMS320C64x(主频600-800MHz)计算能力的4.75%–6%<sup>[9]</sup>,显然随着网络带宽流量的提高,计算量还将显著增加。图7显示RS解码应用的过程,可以明显发现生产者-消费者局域性。基于RS解码的上述特征,可以认定这个应用是流应用。

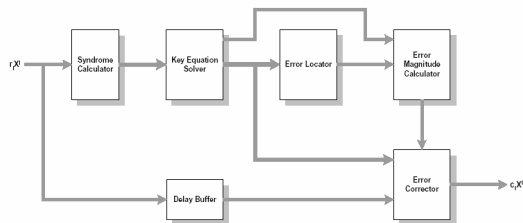


图7 RS解码过程<sup>[11]</sup>

我们将图7所示RS解码过程,依据PGZ算法<sup>[9]</sup>并针对流体系结构模型作修改。修改后的算法包括4步:1、输入码流经过syndrome核心程序的处理产生校正子流;2、BM核心程序对校正子进行操作生成错误定位多项式

系数lambda流;3、chien\_srh核心程序求解错误定位多项式,得到错误码元位置zero流;4、forney核心程序纠正错误码元,输出校正后的码块流。数据流图见图4。

下面讨论应用在流体系结构模型的执行过程。主处理器流出流操作由流控制器动态调度执行,在资源可用并且满足相关性时流控制器主要流出以下操作:

load数据流,从外存储入数据流到流存储系统。数据流包括两类:运算所需查的表,beta因子,流长固定,整个解码过程都需要,只需加载一次,称为初始流;需要解码的码块流,流长不固定,每个码块一旦解码完毕无须再用,新的码块到来需要不断的加载。由于核心程序的执行可以与load/store流和load核心程序并发执行,流控制器可以保证在核心程序执行时,所需要的数据流已经在流存储系统中。

Load核心程序,从外存储入核心程序到核心程序处理器,以流的方式加载。

执行核心程序。核心程序由核心程序处理器控制,在流的计算模块执行,所有操作数从本地寄存器文件读写,输入流和输出流需要从流存储系统读写。执行4个核心程序的操作依顺序流出,对于多个RS码块的解码,核心程序需循环执行。

Store数据流。校正过的码块流由流存储系统存回外存。流控制器监控流存储系统,一旦填满,则更换执行的核心程序以消耗流存储系统中的数据,因此不会发生流存储系统溢出而需要将数据流存回外存的情况。

4个核心程序的加载和执行的时机与执行模式相关。如分时执行,每一次核心程序加载后的运行时间由流存储系统的大小决定,如已经填满则流控制器发出执行其他核心程序的操作。整个执行过程类似核心程序间的软件流水;如分空间执行,核心程序类似于流水线的不同阶段,流控制器需要控制核心程序之间的流在流存储系统的传输。

与传统的非流体系结构的执行的区别在于以下几点:严格分离数据流组织存取和计算,整个应用只在读输入码流和输出码流时访存,中间流完全在片内流转;充分利用片上存储捕捉生产者-消费者局域性,初始流一

且存入在整个应用解码过程中不换出，其开销随着码流的增长几乎可以忽略；计算层次充分利用大规模的计算资源最大限度地开发数据并行；计算可以和数据流存取并行，便于延迟隐藏。

Imagine是一个基本符合本文流体系结构模型的流处理器，根据我们在Imagine上的实现数据<sup>3</sup>（见表2）可以看出：流体系结构在RS解码这种典型的DSP应用问题的处理能力方面可与专用芯片相比，而且流体系结构的扩展性和灵活性明显要优于专用芯片。

表2 RS(204,188)解码（单位为cycles）

	Imagine	C64x <sup>[9]</sup>	
	KernelC StreamC	C代码	优化 汇编
校正子	1122	16635	538
BM算法	360	982	389
钱式搜索	558	2704	594
Forny算法	220	966	234

目前在各种流体系结构上已获证明得到较好实现的应用领域包括嵌入式信号处理、传感器网络、智能软件路由器、高性能图形图像处理等多媒体处理以及部分科学计算<sup>[10]</sup>。

## 6 结论

传统的工作负载需操作的数据零散没有规律，数据的存取和运算只能紧密耦合，访存开销大。根据程序局域性原理增加存储层次和预取操作可减小一定开销，但存在失效等状况。流体系结构模型的操作对象为流元素，同构有序且依序流动，因此将数据运算和数据存取分离，便于延迟隐藏，极大的减少访存开销，这也是流体系结构模型的本质所在。由此带来硬件模型分为流调度模块和流计算模块，层次化的程序设计模型和层次化的编译。

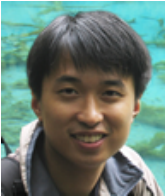
我们将在未来工作中实现该抽象模型的原型系统。

## 参考文献

- [1] 王湘新,文梅等.媒体处理器体系结构.微机发展 Vol14, 2004.6:120~123
- [2] Karthikeyan Sankaralingam et al., Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS architecture, 30<sup>th</sup> Annual International Symposium on Computer Architecture, 2003.5
- [3] Peter Mattson et al, Imagine Programming System Developer's Guide, <http://cva.stanford.edu>, 2002
- [4] Saman Amarasinghe et al, Stream Languages and Programming Models, PACT 2003, September 27, 2003
- [5] Francois Labonte et al, The Stream Virtual Machine, PACT 2004, 2004.9
- [6] Saman Amarasinghe, William. Stream Architectures, PACT 2003, September 27, 2003
- [7] Nan Wu, Mei Wen, et al, Programming design patterns for the Imagine stream architecture, 13<sup>th</sup> National Conference on Information Storage Technology, Xi'an, 2004
- [8] Peter Mattson, Ph.D. Thesis, A Programming System for the Imagine Media Processor, Dept. of Electrical Engineering, Stanford University.2001
- [9] TI company, Reed Solomon Decoder: TMS320C64x Implementation. Application report, 2000
- [10] PACT 2003 Tutorial, Architectures, Languages, and Compilers for the Streaming Domain. <http://catfish.csail.mit.edu/streamit/talks/pact03tutorial/> September 27, 2003
- [11] Reed Solomon Decoder, [www.lateral.com](http://www.lateral.com), 2002.11

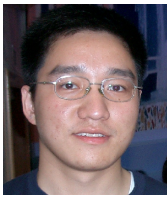


文梅 女, 1975 年生, 博士研究生, 主要研究方向为高性能微处理器体系结构。



伍楠 男, 1981 年生, 硕士研究生, 主要研究方向为高性能微处理器体系结构。

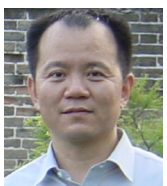
<sup>3</sup> 具体实现参见[7]。



李礼 男，1981 年生，硕士研究生，主要研究方向为高性能微处理器体系结构。



李海燕 女，1981 年生，硕士研究生，主要研究方向为高性能微处理器体系结构。



张春元 男，1964 年生，教授，博士生导师，主要研究方向高性能微处理器体系结构，并行处理技术，嵌入式系统。