多维可扩展流体系结构研究与评测

吴 伟, 文 梅, 伍 楠, 何 义, 杨乾明, 管茂林, 荀长庆, 任 巨, 柴 俊, 张春元
(国防科学技术大学计算机学院并行与分布重点实验室, 湖南长沙 410073)

摘 要: MASA(Multiple-dimension scalable Stream Architecture)是一种可在多个维度扩展的流体系结构. 本文对该体系结构的扩展性进行了深入探讨, 分析了簇内、簇间和多核扩展的 VLSI 资源开销, 并通过一组测试程序评测了 MASA 的性能. 结果表明, 三个扩展维度形成有利互补, 使得 MASA 流体系结构可支持扩展到单片内集成上千个 ALU.

关键词: 流处理器; 流体系结构; 扩展性; 并行处理
中图分类号: TP303 文献标识码: A 文章编号: 0372-2112 (2008) 05-0899-07

Research and Evaluating of a Multiple-dimension Scalable Stream Architecture

WU Wei, WEN Mei, WU Nan, HE Yi, YANG Qiang-ming, GUAN Mao-lin,
XUN Chang-qing, REN Ju, CHAI Jun, ZHANG Chun-yuan
(*National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha, Hunan* 410073, *China*)

Abstract: Multiple-dimension scalable Stream Architecture (MASA) can be scalable in multiple dimensions. This paper discussed the scalability of MASA, and analyzed VLSI resource while scaling in inner-cluster, intra-cluster and multicore. The evaluating model was built by a set of kernels to test the performance of MASA. As presented in this paper, the efficiency of the stream processor architecture enables scaling to thousands of ALUs in a chip by the complementarities of the three dimensions.

Key words: stream processor; stream architecture; scalability; parallel process

1 引言

流体系结构是一种基于并行体系结构发展而成的对数据流进行处理的新型微体系结构, 它拥有大规模的运算单元和多级带宽存储层次来支持对应用的密集计算[1], 主要应用于信号处理、音视频、图像处理、科学计算等领域, 近年来涌现出了 Imagine[2]、TRIPS[3]、CELL[4]、FT64[5]多种流处理器体系结构.

随着人们的需求不断提高, 应用对处理器提出了越来越高的要求, 如天基雷达(SBR)[6]信号强调数据并行处理性能, 在芯片上集成数百个计算单元, 计算需求达到 1TFLOPS[7]. 另一方面, VLSI 工艺的发展使得片上可集成的晶体管数目增加. 面向未来片上十亿支晶体管的容量和应用需求, 流体系结构的扩展能力和扩展效率是体系结构设计人员面临的重大挑战. 多维可扩展流体系结构 MASA 是我们提出的一种可在多个层面上进行扩展的流体系结构, 其扩展维度包括: 簇内扩展、簇间扩展和内核扩展.

本文着重对 MASA 每个扩展维度的资源成本及性能进行分析与评估, 探讨未来流体系结构扩展的扩展方向, 并给出相对扩展成本较小和性能较优的扩展方案.

2 MASA 微体系结构

MASA 流体系结构[8]如图 1 所示, 整个体系结构从内到外可以划分为 3 个硬件层次, 每个层次有其专门的计算、控制、通信模块和特有的存储层次结构.

2.1 运算簇

MASA 流体系结构的最内层采用基于 VLIW 的运算簇(Cluster)结构[9]作为最基本的执行单元, 运算簇内包含多个逻辑运算单元和一些特殊的功能单元, 每个功能单元的每个输入端口都有自己的存储本地数据的本地寄存器文件(LRF), 同一个运算簇的 LRF 之间通过内部的高速共享开关连接互连. 这些本地寄存器文件存储核心程序(Kernel)中的常数、参变量和本地变量, 用以减少对流寄存器文件的带宽要求. MASA 核心级体系结构采用了分布式寄存器文件结构. 分布式寄存器结构降低了面积、延迟和功耗, 有更好的模块化设计特性, 但是以额外的交叉开关和通信总线为代价. 另外当数据被不同的 ALU 使用时, 必须在寄存器文件中放置多个副本, 因此分布式结构需要的寄存器单元数量一般为 SIMD 方式

收稿日期: 2007-04-20; 修回日期: 2008-01-16
基金项目: 国家自然科学基金(No. 60673148, No. 60703073); 国家 863 高技术研究发展计划(No. 2007AA01Z286); 教育部博士点基金(No. 20069998025); 国防科技大学优秀研究生创新资助项目(No. S070604)

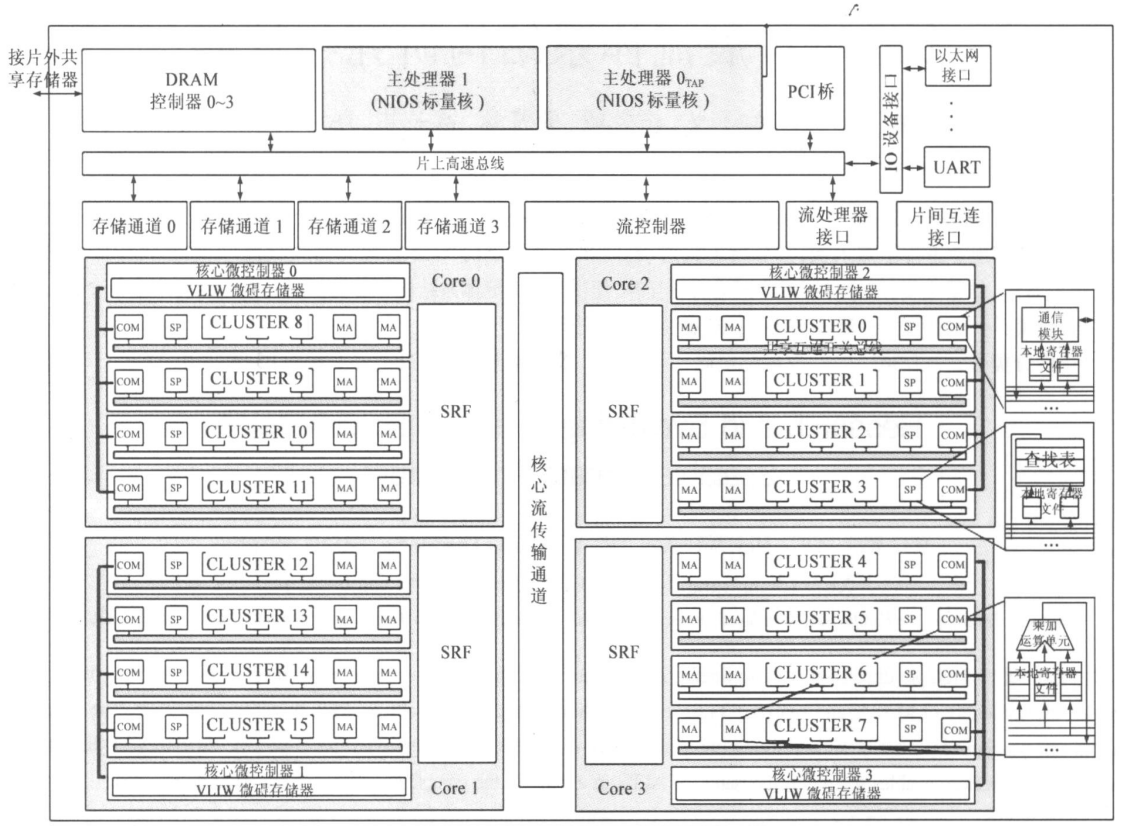

图 1 MASA 流体系结构框图

的 3 倍. 但是寄存器的面积却可以降低一个数量级, 这对于流应用的背景是值得的[10]. 每个运算簇内的所有 ALU 通过可编程的交叉开关互连, 具有快速、灵活的通信能力.

2.2 流内核

每个流内核(Core)主要由 4 个部分组成: 运算簇组, 每个内核包含一定数量的运算簇, 运算簇之间以 SIMD 模式开发数据级并行, 同时簇间设计有一个互连开关用以完成少量的簇间通信. 微控制器(Microcorr troller, UC), 存储核心程序的程序微码, 发射并译码核心指令. 当核心程序启动时, 微控制器逐条广播核心程序微码的 VLIW 指令到运算簇组, 控制运算簇组以 SIMD 方式执行. 流寄存器文件(SRF), SRF 作为外存和 LRF 之间的中间存储层次, 负责存储执行核心需要的输入流和产生的输出流, 可以通过分段输送有效隐藏访存延迟[11]. 流寄存器文件(SRF)隔离了硬件的流级和核心级, 处在 MASA 流体系结构带宽层次的中心, 允许流体系结构的流级和核心级独立扩展. 流缓冲(SB), 能够支持多个流的同时传输, 峰值瞬时带宽可达上百 GB/s, 并可根据应用领域的需求进一步扩展带宽.

2.3 流层

流层是 MASA 体系结构的最顶层, 包含若干流处理核和标量核、流控制器、片外存储器、片上总线和 IO 接口等模块. 其中流处理核是整个结构的核心部件, 流处理核以标量核的协处理器的方式工作, 异构核间通过片上总线互连. 流处理核负责对计算量最大的流应用核心程序加速, 一个 MASA 上可以包含多个这样的核. 每个流处理核可以处理不同的核心程序, 相当于可以同时执行多个计算任务, 开发了流应用的计算型任务级并行. 流控制器保证流指令的正确流出, 并对流指令进行译码. 流控制器通过计分牌技术动态调度指令乱序流出, 最大化处理器执行效率.

MASA 流处理核之间采用了分布式的流寄存器文件, 每个 Core 拥有 1 个本地流寄存器文件, 通过若干流缓冲与其相连. MASA 的核间流传输通路采用了可编程专用数据通道的方式. 这种设计基于两点考虑: (1) 能够提供足够的峰值带宽(由调度机制负责带宽负载平衡和延迟隐藏), (2) 减少开关和不必要的仲裁逻辑以节省硬件.

总之, MASA 流体系结构的层次化结构, 确保内核程序永远不能直接访问主存, 访存延迟通过缓冲加以隐藏, 所有内核的输入输出都是通过储存在 SRF 中的数据流通信. 同时内核的局部数据限定在内部的高速 LRF 中, 减小了对 SRF 的带宽压力, 每个操作数的存取时间都是确定的. 对于这种结构, 可以产生一个精确的静态的 VLIW 时间表, 额外的寄存器更名、相关性检测、乱序流出等硬件都可以大大简化或是取消. MASA 流体系结

构重点解决了高带宽和多 Kernel 执行的问题, 同时结构模块化和局域化特征明显, 有利于提高芯片利用率和降低功耗. MASA 流体系结构根据不同的并行性和带宽层次, 明确划分了体系结构的不同层次, 可以根据不同应用的特点, 在任务级、指令级以及数据级并行等多个维度上进行选择性的扩展, 具有较大规模且有效的扩展能力. 目前正在进行的 ASIC 实现显示频率可以达 500MHz 甚至更高, 单核即能提供接近 18GOPS 的运算性能.

3　扩展资源分析

在 MASA 流体系结构的三个扩展维度上, 设置其每个流处理核内部的部件数量和大小都相同, 则流处理器的资源、延迟和能耗可建模成一个由一些基本单元组成的处理器: 运算单元 ALU、微控制器、运算簇内部互联开关和线、运算簇之间互联开关和线、片上存储. 片上存储包含三类: 流寄存器文件 SRF、本地寄存器文件 LRF(包含便签寄存器 SP)以及流缓冲 SB. 流控制器、存储系统和网络接口等模块实现比较固定, 不会随 ALU 数量的扩展明显扩张, 因此本文不在此讨论. 本文主要讨论各主要模块的资源占用情况, 包括片上逻辑资源(ALUT)和片上存储资源(RAM). 表 1 列出构建运算簇的一些主要基本单元的参数, 其中有关资源数据根据综合和布局布线结果获得. 本节给出了基于 FPGA 实现的多维扩展方案下的资源开销, 并根据实验数据拟合得到资源趋势图. 有研究表明, 流体系结构下功耗、延迟同资源的变化趋势类似[12].

表 1　构建运算簇的基本单元参数

参数	描述	数值
A_{LRF}	每个 LRF 的配置	32 项, 读写端口各 1
A_{ALU}	每个 ALU 的配置	置 32 位乘、加、逻辑操作
A_{SP}	每个便签寄存器文件配置	2048 项 32 位 word
b	流处理器的数据字长	32
b_1	簇内方向增加一个 ALU, VLIW 增加的位数	40bit
b_2	微控制器存储单元中放置 VLIW 条数	2000
N	每个运算簇内 ALU 数量	可扩展
W_{intra}	簇内交叉配置	可扩展
W_{inter}	簇间交叉配置	可扩展
W_{corr}	簇间通道配置	可扩展
N_{SP}	每个运算簇内便签寄存器文件数量	1
N_{COMM}	每个运算簇内通讯单元数量	1
$N_{I/O}$	每个运算簇内 I/O 单元的数量	8
N_{CORE}	单片流处理器上流处理核的数量	可扩展
I_{CISB}	每个流处理核所需的流缓冲数目	8
I_0	除面向流处理核之外的流缓冲数目	5
m	运算簇内每个功能单元需要的 LRF 个数(平均值)	2
C	每个流处理核内运算簇的数量	可扩展

3.1　实验结果

表 2 给出了主要模块在不同维度扩展时的资源占用情况.

表 2　扩展资源占用

簇内扩展方案下部分模块的资源(C= 8)						
		N= 2	N= 4	N= 8	N= 16	N= 32
运算簇(单个)	ALUT	5727	7940	19762	41341	93586
	RAM	9600B	10112B	11136B	13184B	17280B
SRF	ALUT	1812	2876	4656	8414	19530
	RAM	128KB	256KB	512KB	1024KB	2048KB
流缓冲	ALUT	4046	8074	10614	15246	26510
	RAM	1408B	2816B	5632B	11264B	22528B
微控制器	ALUT	1673	1850	2014	2216	2580
	RAM	120KB	140KB	180KB	260KB	420KB
簇间扩展方案下部分模块的资源(N = 4)						
	$C_{Cluster}$	C= 4	C= 8	C= 16	C= 32	C= 64
运算簇(合计)	ALUT	30360	63520	123312	248104	498321
	RAM	40KB	79KB	158KB	316KB	632KB
SRF	ALUT	1812	2876	4656	8414	19530
	RAM	128KB	256KB	512KB	1024KB	2048KB
流缓冲	ALUT	4419	9834	12654	21566	38910
	RAM	1408B	2816B	5632B	11264B	22528B
interswitch	ALUT	410	1296	4724	19424	67624
微控制器	ALUT	1833	1850	1945	2016	2028
	RAM	138KB	140KB	144KB	148KB	156KB

内核扩展方案下部分模块的资源(N = 4, C= 4)								
N_{CORE}		2	3	4	8	10	12	16
Core		29559* 2	29559* 3	29559* 4	29559* 8	29559* 10	29559* 12	29559* 16
SRF	ALUT	2713	4277	6126	13127	16998	20218	24218
	RAM	256KB	392KB	512KB	1024KB	1280KB	1536KB	2048KB
通道	ALUT	864	1824	3280	6528	10147	14756	17040

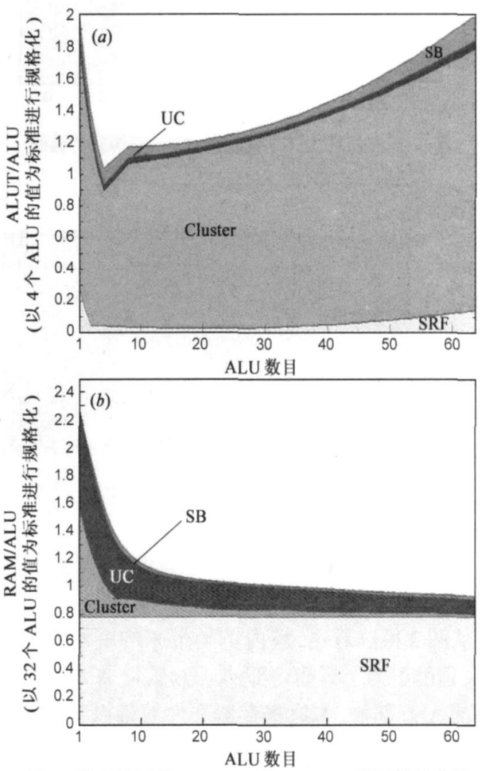

图 2　簇内扩展的 ALUT(*a*)和 RAM(*b*)资源趋势曲线

为了更直观的显示多维扩展的资源成本趋势, 本文

根据的 FPGA 实验结果和已有的部分成本模型按最小二乘法拟合得到资源曲线图,如图 2、图 3、图 4 所示. 其中将各模块资源数据求和除以 ALU 数目求得每个 ALU 所占的各项资源,并做了标准化的工作,图中各线分割的区域为流处理器主要模块对该值的贡献.

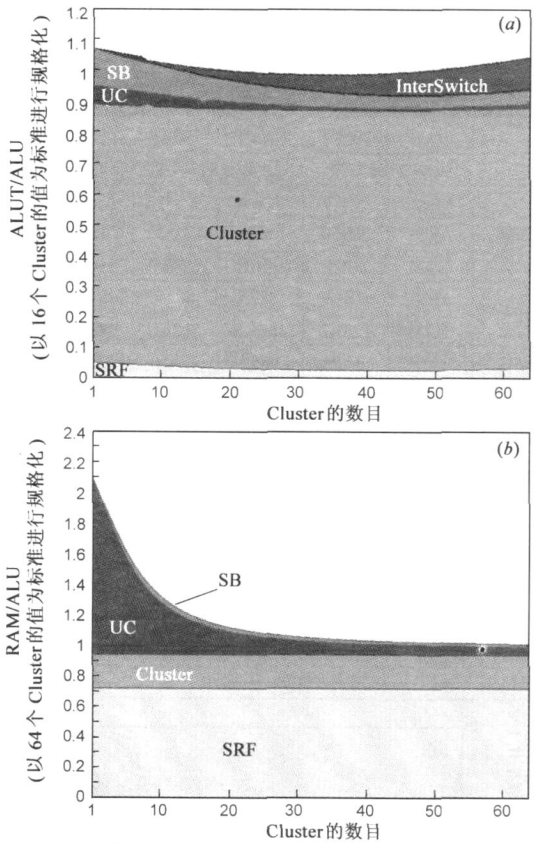

图 3 簇间扩展 ALUT(*a*)和 RAM(*b*)资源趋势曲线

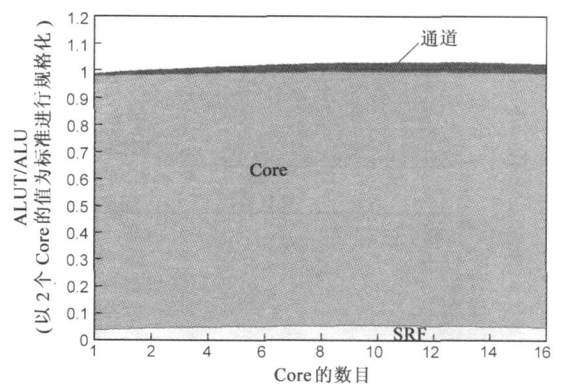

图 4 内核扩展 ALUT 资源趋势曲线

3.2 扩展成本讨论

从图 2 可以看出,簇内扩展带来的资源/ALU 开销最大,是由于当运算簇内部的 ALU 数量 N 增长时,运算簇内部 ALU 部件、本地寄存器文件数量以 $O(N)$ 增长,而簇内互联开关资源以 $O(N^2)$ 增长,因此簇内扩展时资源/ALU 基本成线性增长趋势,N 为 4 至 8 时效率最高,继续增长 ALU 数目将降低效率,这是由于簇内互联开关的影响增加造成的. RAM/ALU 的资源效率呈微降的趋势,这是由于权重最大的流寄存器文件大小随 ALU 数量线性增长,而其他存储部件不变.

与簇内扩展趋势相比,簇间扩展的资源/ALU 开销相对缓和,如图 3 所示,因为簇间通讯单元比簇内互联开关增长缓慢. 从表 2 可以看出 ALUT 资源占有的比重仍是运算簇最大,运算簇间互联开关增长较为明显,在 C 为 8 以下时,其所占资源小于流寄存器文件,但由于其以 $O(C^2)$ 增长因此随 C 的增加很快超过流寄存器文件、流缓冲,在 $C>32$ 时,成为第二大模块. 流寄存器文件与流缓冲的变化与簇内扩展类似,而微控制器变化小于簇内,因为簇间扩展时 VLIW 字长几乎不变. RAM 资源权重最大为流寄存器文件,最小为流缓冲. SRF 和运算簇内 RAM 的容量随运算簇数量扩展线性增长,而微控制器存储由于簇间扩展时 VLIW 几乎不变而变化较小,因此在 C 为 16 以上时本地寄存器文件大小超过了微控制器存储,成为影响 RAM 的第二大因素. 从微控制器所使用的 RAM 大小比较可以发现,当 ALU 数量在 64 以上时,相同数量的 ALU,簇内扩展的要大于簇间扩展,并且随着 ALU 数量的增长差距扩大.

与簇内扩展和簇间扩展的资源利用率比较,多核扩展时资源利用率基本保持水平不变,如图 4 所示,较簇内扩展明显较优,略低于簇间扩展的资源利用率. 并且由于微控制器分布在多个核内,不会出现簇间扩展导致的微控制器与运算簇、运算簇间、微控制器与流控制器间通讯的长延时问题. 因此从 VLSI 成本角度考虑,多核扩展具有优势.

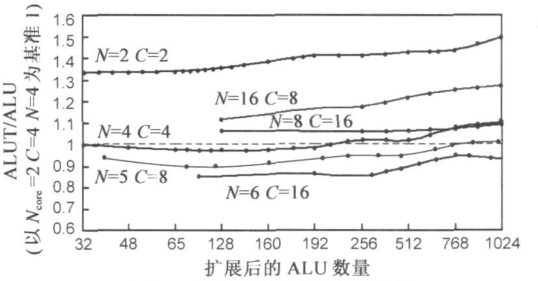

图 5 合并扩展资源效率趋势

综合簇内、簇间和多核这三种扩展维度扩展的 VLSI 开销分析,簇内扩展带来的资源/ALU 开销最大,随运算簇内 ALU 扩展呈线性增长趋势,运算簇的扩展与多核扩展带来的资源/ALU 开销相对缓和. 簇内和簇间扩展后要保持 ALU 高的利用率都会对流应用提出较高的要求,这些不足可以通过多核的扩展来弥补. 综合这三个扩展方向,可以得到如图 5 所示的扩展趋势(图中以 N_{CORE}= 2, C= 4, N= 4 为基准). 实验结果说明,在未来更大规模 VLSI 芯片上,流体系结构可以在合理的开销下在三个维度上综合扩展到上千ALU. 在本文的几种

N_{CORE}= 2 的实验方案中可以看出, C= 16、N= 6 时资源/ALU 是最佳的.

4　扩展性能评测*

MASA 在硬件资源扩展后, 能获得多大的性能提升是我们关心的问题, 本节分别评价三个维度扩展时的性能, 表 3 是评价使用的测试程序, 包括 Kernel 和应用.

表 3　MASA 测试程序

核心程序/流应用	数据类型	简要说明
BlockSearch1(BS1)	Byte4	Mpeg2 中的运动搜索算法, 主要是计算图像残差和
BlockSearch2(BS2)	Byte4	H. 264 中的运动搜索算法, 采用的是 UMHexagonS 算法
RLE	Byte4	Mpeg2 中的熵编码
MatrixMul8(MM8)	Int	块大小为 8× 8 的矩阵乘法
MatrixMul16(MM16)	Int	块大小为 16× 16 的矩阵分块乘法
RS decoder	Int	reed solomon 解码器(204, 188, 8), 包括四个核心程序
DCT	Half2	块大小为 8× 8 的离散余弦变换
Convolve(CV)	Half2	卷积运算, 每次处理 320 个(8× 40) 像素点
MatrixUpdate(MU)	Float	QR 分解中的矩阵变换(转换成上 Hessenberg 矩阵)
QRD	Float	对大小为 256× 256 的上 Hessenberg 矩阵 QR 分解
FFT1024C	Float	快速傅立叶变换(1024 个点)
Mpeg2	Byte4	三帧大小为 360× 288 24b 的视频图像 Mpeg2 标准的图像编码
Ygx2[14]	DFloat	二维拉格朗日和欧拉耦合法求解二维爆轰流体力学问题
Depth	Float	512× 384 像素图像的立体深度提取
QRD	Float	256× 256 矩阵分解

4.1　簇内扩展

对于典型流应用如深度提取等, 超过 80% 的执行时间消耗在核心程序的内层循环上. 而核心程序对运算簇的扩展又非常的敏感, 核心程序的程序特征对指导扩展也有非常重要的作用. 本文将核心程序在不同结构下分别重新编译(C= 8, N 从 2 变化到 14), 并考虑簇内的通讯延迟在 N 增大后会加长的影响. 簇内扩展后核心程序的性能受到 VLIW 编译器开发 ILP(指令级并行) 的影响, ILP 分为几类: 核心程序一遍循环迭代中真实的 ILP, 以及通过软件流水或循环展开将 DLP(数据级并行) 转化而来的 ILP. 而后面的两类通过簇间扩展也可以开发并行.

为了单独评价应用在簇内扩展时的影响, 图 6 所示的数据是核心程序内层循环中每个运算簇的平均 IPC(指令/Cycle), IPC 从 N= 2 时的 1 左右, 到 N= 12 时的 1 至 5, 根据核心程序而不同. 核心程序 CV 的加速情况最好, 说明在内层循环中有大量的 ILP 存在. 其他核心程序的加速不理想, 说明最小循环长度受限于内层循环流元素处理的依赖链长度, 使得这些核心程序调度受限于关键路径而不是运算单元. 核心程序 RLE、MU 的 IPC 较低的原因在于核心程序内层循环的性能不是受限于 ALU 而是其他单元例如通讯, 因此增加运算单元对性能没有明显加速. IPC 的数据表明多数核心程序在簇内扩展超过 6 个 ALU 后, 加速不是很明显.

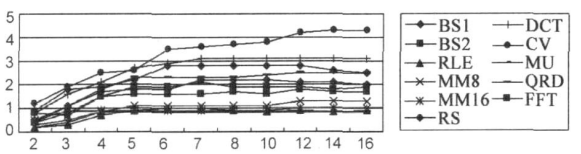

图 6　簇内扩展对核心程序性能的影响: 计算型指令的 IPC

采用软件流水和循环展开等优化技术后, 核心程序内层循环中每个运算簇的平均 IPC 和计算单元利用率, 与未采用前相比有一定改善, 但是由于 ILP 是通过 DLP 转化而来, 在扩展到一定规模后, 会受到短流问题的影响, 这一点与簇间扩展遇到的问题是相同的.

4.2　簇间扩展

运算簇数量增加后, 可以更充分的开发应用的数据并行性, 而对应用来说, 并不一定都能提供. 例如: 核心程序 *BS*1 的运行效率并不会因为运算簇数目的增多而提高. 另外, 运算簇数量增加后, 可能出现短流的问题[13]. 如果运算簇的计算能力与流长度不相匹配就会造成运算性能效率的下降. 运算簇增加会减少核心程序内层循环执行时间, 相对就会增加其他部分的时间比率, 降低执行效率. 然而, 有的应用有自然固定的数据集合, 例如 Mpeg2 输入图像的一行元素较为方便, 更长的流元素则不便于利用; 又例如 QR 分解应用有固定的数据流长, 对这样的流应用, 簇间扩展带来的加速不明显, 反而 ALU 执行效率降低. 图 7 显示表 3 所示应用在簇间扩展后的性能加速比, 考虑了扩展后通讯延迟增加的影响.

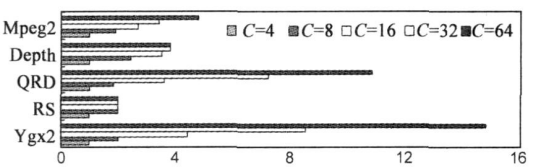

图 7　簇间扩展的应用加速比

由图可以看出, 加速最明显的是 *Ygx*2, 说明 *Ygx*2 有丰富的数据并行, 而且处理器处理规模扩大减少边界处理, 在 C 为 8 和 16 时出现了超线性增长, 但继续增加运算簇数目就带来短流问题, 加速未能达到线性增长. 加速效果最差的是 RS 解码, 因为绝大部分应用中的 RS 码块解码的数据并行度为 8, C> 8 以后的扩展显得没有意义. Mpeg2 和 Depth、QRD 中或者 8× 8 块搜索占相当

* 本节的评测数据通过时钟精确的模拟器 MSIM 获得, MSIM 介绍参见文献[13], 并可从我们的学术网站 masa. nudt. edu. cn 下载.

比重, 或者有固定数据集, 因此加速效果不理想.

4.3 多核扩展

多核系统为应用的实现提供了更灵活的模式, 但是由于流处理器是一种面向部分专用领域的体系结构, 因此应用的并行化上没有强调模式的通用性, 而由程序员针对特定应用更精巧的"量身制作". 本节应用在多核结构上的映射算法也因应用而有较大差异, 下面分别介绍映射的思路. 需要注意的是如果映射算法不同, 结果会有较大的差异.

RS 码块之间数据没有相关性, 但是码块内部数据并行性有限. 因此 RS 在多核上采用 SIMD 并行模式, 同时执行多个 RS 码块解码. 例如每个流处理核解码一个码块, 可以同时进行4路 RS 的解码. MPEG2 数据的生产者消费者局域性较好, 多个流处理核上执行的核心程序间构成流水关系. 因此 MPEG2 在多核上以 MIMD 模式并行, 可以有效降低流寄存器文件中中间结果大小, 并减少核心程序切换开销. Ygx2 数据并行性非常好, 但是流有一定的不规则访问特性, 访存压力也比较大. 因此采用 SIMD 和 MIMD 结合的方式, 对于数据并行性好的核心程序, 在 4 个流处理核执行相同的代码而处理不同的数据. 对于不规则访问, 在某个流处理核上执行专门的数据重组核心程序避免访存, 或者负责启动一个新的存储系统专门访存.

图 8 显示流应用在多核流处理器上的性能加速、功能单元利用率和流处理核利用率, 左边纵坐标是加速比, 右边纵坐标是利用率. 对并行处理器来说, 处理器规模的扩展意味着其处理的数据能力的增加, 因此本文在评价性能加速时, 也考虑了数据流规模扩展的因素.

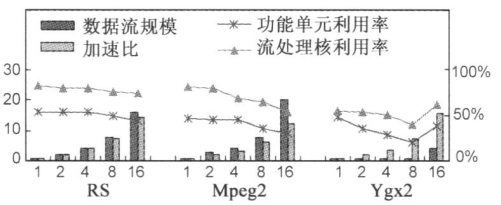

图 8 应用在多核流处理器上的性能

RS 在流处理核增加时码块数量同比增长, 加速基本呈线性增长. 由于主处理器、流控制器负担加重, 在流处理核在 8 以上时, 利用率出现一定程度的下降. Mpeg2 数据集规模根据标准帧大小(760×576 等) 以 360×288 位基准随流处理核增加有不同比率的增加, 与图 7 所示的簇间扩展相比, 对 RS 和 Mpeg2 来说, 多核扩展由于不是单纯的增加 DLP 的开发使得加速效果较好, RS 由于多路解码优势更加明显. 但由于核间通讯及其他开销的增长, 使得 Mpeg2 应用在流处理核的扩展时, 功能单元利用率和流处理核利用率都出现下降. Ygx2 的数据集规模在不变的情况下, 扩展流处理核同样获得加速, 加速效果与簇间扩展相似, 说明科学计算有丰富的数据并行, 并且由于数据集较大在 8 个流处理核时短流效应都不明显. 由于与 RS、Mpeg2 类似的原因, 利用率出现下降, 但是当扩展数据集后, 又有所回升.

实验数据表明: 簇内扩展时多数核心程序在 N 的数量超过 6 以后 IPC 没有明显增长, 在 N 逐渐增大后由于簇内通讯延迟增加和 IPC 不能充分开发使得运算单元利用率反而下降, 进而对性能造成不利影响; 簇间扩展时, 数据并行度为 8 的或是有固定数据集合的应用在 $C>8$ 以后, 由于不能充分开发 DLP 引发通讯、冗余计算以及短流效应等不利影响造成加速效果不理想; 多核扩展后的并行执行模式有多种, 结合不同应用的特征采用不同的映射方式, 由于多核扩展不是单纯的开发 DLP 或 ILP, 使得加速效果较好.

5 结论

本文基于 MASA 流体系结构, 在实验的基础上对扩展的成本和性能进行了讨论和分析.

从扩展的 VLSI 开销分析, 簇内扩展带来的资源/ALU 开销最大, 随运算簇内 ALU 扩展呈线性增长趋势, 簇间扩展与多核扩展带来的资源/ALU 开销相对缓和. 而多核扩展的资源利用率略低于簇间扩展, 并且由于微控制器分布在多个核内, 不会出现簇间扩展导致的微控制器与运算簇、运算簇间、微控制器与流控制器间通讯的长延时问题. 因此从 VLSI 成本角度考虑, 多核扩展具有优势.

从扩展的性能分析, 簇内扩展和簇间扩展后核心程序和流应用都不一定能获得性能上的同比加速, 因为核心程序的最小循环长度受限于内层循环流元素处理的依赖链长度, 而流应用不一定能提供更多的 DLP 或者更多的数据流长, 使得 DLP 不能得到充分地开发或者是出现短流效应. 多核扩展后的并行执行模式有多种, 结合不同应用的特征采用不同的映射方式, 不是单纯的开发 DLP 或 ILP, 因而使得加速效果较好.

多核扩展以较低的硬件开销和对应用更多适应性而与其他两个扩展维度形成有利互补. 总之, 流体系结构可扩展性极佳, 可支持扩展到单片内集成上千个 ALU.

参考文献:

[1] Scotty Rixner. Stream Processor Architecture[M]. Boston, MA: Kluwer Academic Publishers, 2001.

[2] U J Kapasi, W J Dally, et al. The imagine stream processor [A]. In Proceedings of the IEEE International Conference on Computer Design[C]. Piscataway, NJ: IEEE Press, 2002. 282-288.

[3] Karthikeyan Sankaralingam et al. Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture[A]. 30th Annual International Symposium on Computer Architecture[C]. New York, NY: ACM Press, 2003. 422- 433.

[4] J A Kahle, et al. Introduction to the Cell multiprocessor[J]. IBM J RES & DEV, 2005, 49(4/5): 589- 604.

[5] Xuejun Yang, Xiaobo Yan, et al. A 64-bit stream processor architecture for scientific applications[A]. 34th Annual International Symposium on Computer Architecture[C]. New York, NY: ACM Press, 2007. 210- 219.

[6] Leopold J Cantafio. SPACD-BASED RADAR HANDBOOK [M]. Norwood, MA: Artech House Inc, 1989.

[7] Robert Bond. High Performance DoD DSP Applications[DB/OL]. http://catfish.csail.mit.edu/wss03/, 2003.

[8] Mei Wen, Nan Wu, Haiyan Li, Chunyuan Zhang. A multiple-dimension scalable adaptive stream architecture[A]. Ninth Asiar Pacific computer system Architecture Conference[C]. Berlin: Spinger Press, 2004. 199- 211.

[9] Christos Kozyrakis, David Patterson. Vector vs. superscalar and VLIW architectures for embedded multimedia benchmarks[A]. In Proceedings of the 35th Annual IEEE/ACM International Symposium on Microarchitecture[C]. Los Alamitos, CA: IEEE Computer Society Press, 2002. 283- 293.

[10] Scott Rixner, William J Dally, et al. Register organization for media processing[A]. In Proceedings of the Sixth International Symposium on High Performance Computer Architecture[C]. San Fransisco, CA: Morgan Kaufmann Press, 2000. 375- 387.

[11] Nan Wu, Mei Wen, Haiyan Li, Li Li, Chunyuan Zhang. A stream architecture supporting multiple stream execution models[A]. Tenth Asiar Pacific computer system Architecture Conference[C]. Berlin: Spinger Press, 2005. 143- 156.

[12] Brucek Khailany. The VLSI Implementation and Evaluation of Area and Energy Efficient Streaming Media Processors"[D]. Palo Alto, CA: Stanford University. 2003.

[13] 文梅. 流体系结构关键技术研究[D]. 长沙: 国防科学技术大学计算机学院. 2006.

[14] 袁国兴, 邵京云. 评几种高档微处理器在运算科学计算问题时的性能[DB/OL]. www.ccw.com.cn. 2003- 04.

作者简介:

吴　伟　男, 1984年生于福建, 国防科学技术大学计算机学院研究生. 主要研究方向为高性能微处理器体系结构、并行程序设计.
E-mail: ww7tc@sina.com

文　梅　女, 1975年生于湖南, 国防科学技术大学计算机学院并行与分布重点实验室副教授, 博士. 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.