

分类号 TP302.1

学号 09069034

U D C

密级 公开

工学博士学位论文

Cache一致性片上网络路由算法和流控机制 优化关键技术研究

博士生姓名 马 胜

学科专业 计算机科学与技术

研究方向 计算机体系结构

指导教师 王志英 教授

国防科学技术大学研究生院

二〇一二年十月

Research on the Key Techniques of Routing Algorithm and Flow Control Optimizations for Cache-Coherent Networks-on-Chip

**Candidate: MA Sheng
Supervisor: Professor WANG Zhiying**

A dissertation

**Submitted in partial fulfillment of the requirements
for the degree of Doctor of Engineering
in Computer Science and Technology**

Graduate School of National University of Defense Technology

Changsha, Hunan, P. R. China

October 17, 2012

独 创 性 声 明

本人声明所呈交的学位论文是我本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表和撰写过的研究成果，也不包含为获得国防科学技术大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文题目：Cache 一致性片上网络路由算法和流控机制优化关键技术研究

学位论文作者签名： 马胜 日期：2012 年 10 月 12 日

学 位 论 文 版 权 使用 授 权 书

本人完全了解国防科学技术大学有关保留、使用学位论文的规定。本人授权国防科学技术大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档，允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

(保密学位论文在解密后适用本授权书。)

学位论文题目：Cache 一致性片上网络路由算法和流控机制优化关键技术研究

学位论文作者签名： 马胜 日期：2012 年 10 月 12 日

作者指导教师签名： 卫东来 日期：2012 年 10 月 12 日

目 录

| | |
|--|-----------|
| 摘要 | i |
| ABSTRACT | iii |
| 第一章 绪论 | 1 |
| 1.1 研究背景 | 1 |
| 1.1.1 众核时代的到来 | 2 |
| 1.1.2 片上网络的提出 | 3 |
| 1.1.3 众核结构 Cache 一致性协议 | 5 |
| 1.2 国内外研究现状 | 7 |
| 1.2.1 国外研究现状 | 7 |
| 1.2.2 国内研究现状 | 12 |
| 1.3 课题研究的目标和意义 | 13 |
| 1.4 研究内容和创新点 | 15 |
| 1.4.1 主要研究内容 | 15 |
| 1.4.2 创新点 | 17 |
| 1.5 论文组织结构 | 19 |
| 第二章 Cache 一致性片上网络 | 21 |
| 2.1 片上网络背景知识 | 21 |
| 2.1.1 拓扑结构 | 21 |
| 2.1.2 路由算法 | 22 |
| 2.1.3 流控机制 | 25 |
| 2.1.4 路由器微结构和流水线 | 28 |
| 2.2 一致性协议对片上网络设计的影响 | 31 |
| 2.2.1 Cache 一致性协议存在的必然性 | 31 |
| 2.2.2 Cache 一致性协议通信特性 | 33 |
| 2.3 模拟环境和性能测试方法学 | 37 |
| 2.3.1 Booksim 模拟器 | 37 |
| 2.3.2 网络性能测试方法 | 39 |
| 2.3.3 FeS2 模拟器及其与 Booksim 模拟器的整合 | 39 |
| 第三章 面向负载整合工作模式的路由算法 | 41 |
| 3.1 引言 | 41 |

| | |
|---|-----------|
| 3.2 相关研究 | 42 |
| 3.3 研究动机 | 44 |
| 3.3.1 局部自适应算法的局限性 | 44 |
| 3.3.2 应用程序内部的干扰 | 45 |
| 3.3.3 应用程序之间的干扰 | 45 |
| 3.4 面向负载整合的自适应路由算法 | 48 |
| 3.4.1 拥塞信息传播网络 | 48 |
| 3.4.2 DBAR 路由器微结构 | 50 |
| 3.5 实验评估 | 52 |
| 3.5.1 单区域性能 | 53 |
| 3.5.2 多区域性能 | 56 |
| 3.5.3 Concentrated Mesh 性能 | 60 |
| 3.6 硬件开销讨论 | 62 |
| 3.6.1 连线资源 | 62 |
| 3.6.2 路由器开销 | 62 |
| 3.6.3 功耗和能量延迟积 | 62 |
| 3.7 进一步讨论 | 63 |
| 3.7.1 拥塞信息传播网络带宽 | 63 |
| 3.7.2 DBAR 的可扩展性 | 64 |
| 3.7.3 拥塞信息传播延迟 | 64 |
| 3.8 本章小结 | 64 |
| 第四章 面向 Cache 一致性通信的完全自适应路由算法 | 65 |
| 4.1 引言 | 65 |
| 4.2 相关研究 | 67 |
| 4.2.1 死锁避免理论 | 67 |
| 4.2.2 完全自适应路由算法设计 | 68 |
| 4.3 研究动机 | 68 |
| 4.3.1 虚通道分配策略 | 69 |
| 4.3.2 路由灵活性 | 70 |
| 4.4 基于全报文发送策略的完全自适应路由算法 | 71 |
| 4.4.1 全报文发送 | 72 |
| 4.4.2 完全自适应路由算法 | 74 |
| 4.4.3 路由器微结构 | 75 |

| | |
|---|-----------|
| 4.5 实验评估 | 76 |
| 4.5.1 合成流量模式结果 | 78 |
| 4.5.2 PARSEC 测试集结果 | 80 |
| 4.6 敏感性分析 | 80 |
| 4.6.1 单切片报文比例 | 81 |
| 4.6.2 虚通道深度 | 81 |
| 4.6.3 虚通道数目 | 82 |
| 4.6.4 网络规模 | 83 |
| 4.7 进一步讨论 | 84 |
| 4.7.1 报文长度和虚通道深度 | 84 |
| 4.7.2 DAMQ 和混合流控机制 | 85 |
| 4.8 本章小结 | 85 |
| 第五章 面向 Torus 片上网络的切片气泡流控机制 | 87 |
| 5.1 引言 | 87 |
| 5.2 传统设计的局限 | 89 |
| 5.2.1 Dateline | 89 |
| 5.2.2 本地气泡策略 (LBS) | 89 |
| 5.2.3 关键气泡策略 (CBS)) | 90 |
| 5.2.4 处理变长报文的低效性 | 90 |
| 5.3 切片气泡流控机制 | 91 |
| 5.3.1 理论描述 | 92 |
| 5.3.2 本地切片气泡策略 (FBFC-L) | 93 |
| 5.3.3 关键切片气泡策略 (FBFC-C) | 94 |
| 5.3.4 饿死现象 | 95 |
| 5.4 路由器流水线和微结构 | 96 |
| 5.4.1 FBFC 路由器 | 97 |
| 5.4.2 VCT 路由器 | 98 |
| 5.5 实验方法 | 98 |
| 5.6 一维 Torus 网络 (Ring 网络) 性能测评 | 100 |
| 5.6.1 性能 | 100 |
| 5.6.2 缓存利用率 | 102 |
| 5.6.3 短报文和长报文的传输延迟 | 103 |
| 5.7 二维 Torus 网络性能测评 | 103 |
| 5.7.1 4x4 Torus 网络性能 | 103 |

| | |
|---|------------|
| 5.7.2 单切片比例敏感性分析 | 105 |
| 5.7.3 缓存数量敏感性分析 | 107 |
| 5.7.4 8×8 Torus 网络可扩展性分析 | 108 |
| 5.7.5 饿死现象对性能的影响 | 108 |
| 5.7.6 PARSEC 测试集实验结果 | 110 |
| 5.7.7 FBFC 硬件开销 | 111 |
| 5.8 进一步讨论 | 112 |
| 5.9 相关研究 | 112 |
| 5.10 本章小结 | 113 |
| 第六章 高效支持 cache 一致性协议归约和多播通信的技术 | 115 |
| 6.1 引言 | 115 |
| 6.2 归约消息组合框架 | 117 |
| 6.2.1 消息组合表格式 | 118 |
| 6.2.2 消息组合实例 | 119 |
| 6.2.3 消息组合表项不足 | 121 |
| 6.3 均衡自适应多播路由算法 | 121 |
| 6.4 路由器流水线和微结构 | 123 |
| 6.5 实验评估 | 124 |
| 6.5.1 性能 | 126 |
| 6.5.2 BAM 和 RPM 多播虚拟网络性能 | 129 |
| 6.5.3 消息组合表大小 | 131 |
| 6.5.4 敏感性分析 | 132 |
| 6.6 功耗和能量延迟积分析 | 134 |
| 6.7 相关研究 | 136 |
| 6.7.1 消息组合 | 136 |
| 6.7.2 片上网络多播路由算法 | 136 |
| 6.8 本章小结 | 137 |
| 第七章 结束语 | 139 |
| 7.1 工作总结 | 139 |
| 7.2 研究展望 | 141 |
| 致谢 | 143 |
| 参考文献 | 145 |
| 作者在学期间取得的学术成果 | 161 |

表 目 录

| | | |
|-------|------------------------------|-----|
| 表 1.1 | 近三年有关片上网络的国家自然科学基金项目 | 13 |
| 表 2.1 | 各种报文传输层流控机制特性比较 | 27 |
| 表 2.2 | 合成流量模式定义 | 39 |
| 表 3.1 | 全系统模拟参数 | 53 |
| 表 3.2 | DBAR 平均性能提升 | 60 |
| 表 3.3 | DBAR 在不同带宽拥塞信息传播网络下的性能 | 64 |
| 表 4.1 | 物理或虚拟网络数目 | 66 |
| 表 4.2 | 虚通道分配器关键路径延迟和面积开销 | 76 |
| 表 4.3 | 实验配置参数 | 77 |
| 表 4.4 | 全系统模拟参数 | 77 |
| 表 4.5 | FULLY+WPF 的平均饱和吞吐率提升 | 80 |
| 表 5.1 | 延迟实验结果 (单位 FO4) | 99 |
| 表 5.2 | 全系统模拟参数 | 100 |
| 表 5.3 | 网络饱和时的缓存利用率分布 | 103 |
| 表 6.1 | 实验配置参数 | 125 |
| 表 6.2 | 全系统模拟参数 | 126 |
| 表 6.3 | MCT 表硬件开销 | 132 |

图 目 录

| | |
|---|----|
| 图 1.1 单芯片处理器核数量增长趋势 | 3 |
| 图 1.2 共享存储多处理器结构 | 6 |
| 图 1.3 广播协议 Read cache miss 处理过程 | 6 |
| 图 1.4 目录协议 Read cache miss 处理过程 | 7 |
| 图 1.5 近五年在体系结构顶级会议上发表的片上网络文章数 | 11 |
| 图 1.6 论文组织框图 | 20 |
| 图 2.1 片上网络常用的三种拓扑结构 | 21 |
| 图 2.2 褶曲 torus 网络 | 22 |
| 图 2.3 非最短路由和最短路由 | 23 |
| 图 2.4 维序、O1Turn 和自适应路由算法 | 23 |
| 图 2.5 四个报文相互阻塞的网络死锁 | 24 |
| 图 2.6 转向模型路由算法 | 24 |
| 图 2.7 存储 - 转发、虚切通和虫孔流控机制示意图 | 26 |
| 图 2.8 Dateline 设计使用两条虚通道消除循环依赖 | 27 |
| 图 2.9 典型片上网络虚通道虫孔路由器结构 | 28 |
| 图 2.10 基准流水线及其优化 | 30 |
| 图 2.11 消息依赖关系及其导致的死锁 | 34 |
| 图 2.12 AlphaServer GS320 机器一致性协议的四种基本事务 | 34 |
| 图 2.13 Cache 一致性协议通信模式 | 36 |
| 图 2.14 64 核平台负载整合工作模式 | 36 |
| 图 2.15 Booksim 模拟器结构框图 | 38 |
| 图 2.16 片上网络注入率 - 延迟曲线图 | 40 |
| 图 3.1 自适应路由算法的结构 | 43 |
| 图 3.2 LOCAL 和 NoP 报文路由实例 | 44 |
| 图 3.3 RCA-1D 报文路由实例 | 46 |
| 图 3.4 合成流量模式的平均跳数 | 46 |
| 图 3.5 负载整合场景报文路由实例 | 47 |
| 图 3.6 R0 区域性能曲线图 | 47 |
| 图 3.7 DBAR 报文路由实例 | 48 |
| 图 3.8 拥塞信息传播网络结构 | 49 |
| 图 3.9 路由器 (3,2) 的拥塞信息存储寄存器格式 | 50 |

| | |
|---|----|
| 图 3.10 DBAR 路由器流水线 | 51 |
| 图 3.11 DBAR 路由器微结构 | 51 |
| 图 3.12 SMC 模块的伪码 | 52 |
| 图 3.13 DP 模块的硬件实现 | 53 |
| 图 3.14 路由算法在 4×4 mesh 网络 (区域) 上的性能 | 54 |
| 图 3.15 路由算法在单区域 8×8 mesh 网络上的性能 | 55 |
| 图 3.16 PARSEC 测试集的全系统加速比 | 56 |
| 图 3.17 中型规则区域配置 | 57 |
| 图 3.18 非规则区域配置 | 57 |
| 图 3.19 路由算法在 6×6 mesh 网络 (区域) 上的性能 | 58 |
| 图 3.20 路由算法在非规则区域上的性能 | 59 |
| 图 3.21 CMesh 网络的配置 | 60 |
| 图 3.22 路由算法在 CMesh 网络 (区域) 上的性能 | 61 |
| 图 3.23 8×8 mesh 网络 transpose 模式的功耗和能量延迟积 | 63 |
| 图 4.1 PARSEC 测试集单切片报文比例 | 66 |
| 图 4.2 路由算法在 bit reverse 流量模式下的性能 | 67 |
| 图 4.3 完全自适应路由算法的死锁实例 | 69 |
| 图 4.4 保守分配策略降低了虚通道利用率 | 70 |
| 图 4.5 虚通道分配器的第一级仲裁器结构 | 70 |
| 图 4.6 优先选择输出端口设计中的死锁 | 71 |
| 图 4.7 全报文发送实例 | 72 |
| 图 4.8 基于 $Config_0$ 构造一个 $Alg + WPF$ 的网络配置 | 73 |
| 图 4.9 所提出的虚通道分配器结构图 | 76 |
| 图 4.10 路由算法在基准网络配置下的性能 | 78 |
| 图 4.11 PARSEC 测试集的全系统加速比 | 81 |
| 图 4.12 路由算法在不同单切片报文比例下的性能 | 81 |
| 图 4.13 路由算法在不同虚通道深度下的性能 | 82 |
| 图 4.14 路由算法在 4 条虚通道配置下的性能 | 83 |
| 图 4.15 路由算法在 8×8 mesh 网络中的性能 | 84 |
| 图 5.1 PARSEC 测试集单切片报文分布 | 88 |
| 图 5.2 Dateline 需要使用两条虚通道 (VC) | 89 |
| 图 5.3 LBS 要求每条虚通道至少能容纳两个报文 | 90 |
| 图 5.4 CBS 要求每条虚通道能容纳一个报文 | 90 |
| 图 5.5 LBS 在处理变长报文时面临的死锁 | 91 |

| | |
|---|-----|
| 图 5.6 虫孔交换网络中的报文路由实例 | 92 |
| 图 5.7 FBFC-L 报文路由实例 | 94 |
| 图 5.8 FBFC-C 报文路由实例 | 95 |
| 图 5.9 FBFC-L 饿死实例 | 96 |
| 图 5.10 FBFC-C 饿死实例 | 96 |
| 图 5.11 FBFC 路由器微结构 | 97 |
| 图 5.12 八节点 ring 网络上的性能 | 101 |
| 图 5.13 Uniform random 流量模式下的缓存利用率 | 102 |
| 图 5.14 短报文 (图中 S) 和长报文 (图中 L) 的传输延迟 | 104 |
| 图 5.15 4×4 torus 网络上的性能 | 105 |
| 图 5.16 网络达到饱和之后的性能 | 106 |
| 图 5.17 不同单切片报文比例下的饱和吞吐率 | 106 |
| 图 5.18 Uniform random 模式在其它缓存配置下的性能 | 107 |
| 图 5.19 8×8 torus 网络中的性能 | 109 |
| 图 5.20 不同饿死阈值下的性能 | 109 |
| 图 5.21 不同饿死阈值下的饱和吞吐率 | 110 |
| 图 5.22 PARSEC 测试集的全系统加速比 | 111 |
| 图 6.1 PARSEC 测试集多播报文的平均目标节点数目 | 116 |
| 图 6.2 多播 - 归约事务延迟 | 117 |
| 图 6.3 消息组合框架概述 | 118 |
| 图 6.4 消息组合表 (MCT 表) 格式 | 119 |
| 图 6.5 多播报文传输实例 | 119 |
| 图 6.6 ACK 报文传输实例 | 120 |
| 图 6.7 网络区域划分示意图 | 122 |
| 图 6.8 BAM 多播路由算法逻辑 | 123 |
| 图 6.9 路由器流水线 | 123 |
| 图 6.10 路由器微结构 | 124 |
| 图 6.11 网络总体性能 | 127 |
| 图 6.12 多播 - 归约事务传输延迟 | 128 |
| 图 6.13 PARSEC 测试集结果 | 129 |
| 图 6.14 单播通信在 RPM 和 BAM 多播虚拟网络中的性能 | 130 |
| 图 6.15 100% 多播报文比例下的性能 | 131 |
| 图 6.16 消息组合表 (MCT) 大小评估 | 132 |
| 图 6.17 RPM+Com 和 BAM+Com 相对于 RPM+NonCom 的性能提升 | 133 |

图 6.18 功耗和能量延迟积结果 135

摘要

半导体技术的发展不断增加芯片中集成的核数，传统总线或点对点通信架构面临着带宽、延迟、功耗和可扩展性等方面不足。针对这些局限，片上网络应运而生，它能在芯片内部提供一种简单、高效和可扩展的通信机制。另一方面，由于并行编程的高难度和兼容历史代码的需求，cache 一致性协议在众核结构上将长期存在。在 cache 一致性众核结构上，片上网络传输的通信主要由采用的一致性协议所决定，为高效支持这些通信，需要在分析一致性协议结构和通信特征的基础上对片上网络进行优化设计。本文的主要研究成果及创新点如下：

(1). 提出了一种面向负载整合工作模式的路由算法

Cache 一致性协议的层次结构和应用程序有限的并行度导致多应用程序同时运行在一个众核平台上，这种负载整合工作模式要求路由算法能提供良好的适应性和动态隔离性。本文提出了基于目标的自适应路由 (Destination-Based Adaptive Routing, DBAR) 算法。通过使用一个低开销的拥塞信息传播网络，DBAR 获得了本地和远端的网络状态信息，从而能有效避免网络拥塞。更重要的，通过将目标信息集成到输出端口选择中，DBAR 能动态隔离多应用程序。在多种网络配置下，DBAR 的性能都优于之前的设计。

(2). 提出了一种面向 cache 一致性通信的完全自适应路由算法

由于面积和功耗的限制，cache 一致性片上网络一般配置有限的虚通道数目，它给完全自适应路由算法设计提出了新的挑战。之前的死锁避免理论要求使用保守虚通道分配策略，严重限制了路由算法的性能。本文提出了全报文发送 (Whole Packet Forwarding, WPF) 虚通道分配策略，并证明了 WPF 不会带来死锁，WPF 能显著提升路由算法的性能，它对之前的死锁避免理论进行了重要扩展。在虚通道受限环境中，路由算法应该提供较高的路由灵活性，本文进一步给出了一种在较低硬件开销条件下提供较高路由灵活性的完全自适应路由算法。

(3). 提出了面向 torus 片上网络的切片气泡流控机制

Cache 一致性片上网络需要同时传输长报文和短报文，已有的 torus 网络死锁避免理论不能高效处理这种混合长度报文的传输，它们要么需要使用两条虚通道，要么需要将短报文视为长报文。本文提出了切片气泡流控 (Flit Bubble Flow Control, FBFC) 死锁避免理论，FBFC 通过在虫孔交换 ring 网络上维持一个空闲缓存单元避免死锁。FBFC 只使用一条虚通道，获得了较高的路由器频率；FBFC 不需要将每个短报文视为长报文，提高了缓存利用率。基于 FBFC 理论，本文给

出了两种实现，它们的性能都优于已有设计。

(4). 提出了一种高效支持 cache 一致性协议归约和多播通信的技术

Cache 一致性协议需要使用归约和多播通信，为了防止这些通信成为系统性能瓶颈，必须要对它们提供硬件支持。本文研究了对目录一致性协议的多播 cache 行作废消息和归约 acknowledgement (ACK) 消息的硬件支持。本文提出了一种消息组合框架支持归约 ACK 消息的组合操作，该框架不仅在低到中等负载下降低了报文平均延迟，同时也提升了网络吞吐率，它只需少量的额外硬件开销。此外，本文还提出了均衡自适应多播路由 (Balanced, Adaptive Multicast, BAM) 算法，该算法均衡了不同维度间的缓存资源，进一步提升了网络吞吐率。

综上所述，本文紧紧围绕“面向 cache 一致性通信优化片上网络设计”这一目标，基于对一致性协议结构和通信特征的分析，优化设计了路由算法和流控机制。这些优化设计不仅取得了一定的系统性能提升，同时也扩展了已有的死锁避免理论。因此，本文既具备一定的工程价值，又具备一定的理论意义。

关键词: 片上网络; Cache 一致性协议; 负载整合; 完全自适应路由; 虚通道分配; 切片气泡流控; 归约通信; 多播通信

ABSTRACT

The advancement of semiconductor technology increases the core count, and makes the traditional bus or point-to-point communication mechanisms face several challenges, including low bandwidth, high latency, high power consumption, low scalability and etc.. To address these challenges, Network-on-Chip (NoC) was proposed. NoC was regarded as a simple, efficient and scalable communication paradigm for future many-core platforms. On the other side, due to the difficulty of parallel programming and compatibility requirements of history codes, cache coherence protocols will exist in many-core platforms. In cache coherent many-core platforms, the traffic delivered by NoC is mostly decided by the applied cache coherence protocol. To provide efficient communication support for coherent traffic, it is necessary to analyze the traffic characteristics, and then optimize the NoC design. The main contributions of this thesis are as follows.

1. Efficient routing algorithm to support workload consolidation scenarios.

Due to the hierarchical cache coherence protocol and limited application parallelism, it is quite possible that multiple applications will run concurrently in a many-core platform. These workload consolidation scenarios require the routing algorithm to provide both sufficient adaptivity and dynamic isolation. This thesis proposes Destination-Based Adaptive Routing (DBAR). By leveraging a low-cost congestion propagation network, DBAR utilizes both local and non-local network status to efficiently avoid congestion. More importantly, by integrating the destination information into the output port selection procedure, DBAR dynamically isolate multiple concurrent applications. DBAR offers better performance than the best baseline algorithm for many measured configurations.

2. Efficient design of fully adaptive routing algorithm for cache coherent traffic

Due to area and power consumption limitations, cache coherent NoC generally configure a small number of virtual channels (VCs). Limited VCs pose several challenges to the design of fully adaptive routing algorithms. Previous deadlock avoidance theories require a conservative VC re-allocation scheme, which strongly limits the performance of routing algorithms. This thesis proposes a novel VC re-allocation scheme, whole packet forwarding (WPF). We prove that WPF does not induce deadlock, thus WPF is an important extension of previous deadlock-avoidance theories. WPF can greatly improve the performance of fully adaptive routing algorithms. To efficiently utilize WPF in VC-

limited networks, we design a novel fully adaptive routing algorithm which maintains packet adaptivity without significant hardware cost.

3. Flit bubble flow control for torus cache coherent NoCs

Short and long packets commonly co-exist in cache-coherent networks-on-chip (NoCs). Existing deadlock avoidance designs for torus networks do not efficiently handle this mix of packet sizes. These previous designs either leverage two VCs or regard each packet as a maximum-length packet. We propose a novel deadlock avoidance theory, flit bubble flow control (FBFC). The insight of FBFC is that maintaining one free flit-size buffer slot inside a ring can avoid deadlock for wormhole torus networks. Only one VC is required. FBFC does not treat short packets as long ones; this yields high buffer utilization. Based on this theory, we present two implementations, and both show large performance improvement than previous designs.

4. Efficient support of collective communication in cache coherence protocols

Cache coherence protocol utilizes reduction and multicast. Hardware support is necessary to prevent these collective communication from becoming a system bottleneck. This research explores support for reduction and multicast communication operations in a directory cache coherence protocol. This paper makes two primary contributions: an efficient framework to support the reduction of ACK packets and a novel Balanced, Adaptive Multicast (BAM) routing algorithm. By combining ACK packets during transmission, this framework not only reduces packet latency, but also improves the network saturation throughput with little overhead. The balanced buffer resource configuration of BAM helps to get some additional saturation throughput improvements.

In summary, this thesis aims to ‘optimizing the design of NoCs for cache coherence protocols’. Based on the analysis of the characteristics of coherent traffics, we optimize the design of routing algorithms and flow control mechanisms for NoCs. The proposed optimizations not only improve the performance, but also extend the deadlock avoidance theories. Thus, this thesis has both engineering value and theoretical significance.

Key Words: Networks-on-Chip; Cache Coherence Protocol; Workload Consolidation; Fully Adaptive Routing; VC Re-allocation; Flit Bubble Flow Control; Reduction Communication; Multicast Communication

第一章 绪论

在过去 30 年的历程中，处理器借助半导体工艺的巨大牵引取得了飞速发展。截至目前，半导体工艺已具备单芯片集成几十亿只晶体管的能力，给处理器发展创造了源源不断的动力，但也给未来处理器体系结构带来了新的机遇与挑战。如何有效利用数目众多的晶体管是设计者必须回答的问题。经过近几年的发展，业界普遍意识到处理器多核化是提高计算性能的有效手段，已成为目前体系结构的最重要发展趋势。但人们对性能的追求是无止境的，工艺的发展必然会让芯片计算核数量持续增加。最近，人们已提出“众核”(many-core)概念，2007 年 6 月在西雅图召开了以“众核”为主题的研讨会，标志着“众核”已成为体系阶段发展的新趋势。在这种情况下，单芯片集成几十甚至上百个内核，诸多内核之间的互连机制将会成为性能瓶颈。为此，探索如何在芯片多内核之间提供一种简单、高效且稳定的通信机制显得意义深远，它将密切关系到未来“众核”体系结构的发展方向。近年来，芯片设计工程师们借鉴网络领域的概念提出一种新的通信技术——片上网络(Networks-on-Chip, NoC)^[1-3]。片上网络较好地适应了工艺发展趋势，它除具有更好的可预测性和更低的功耗开销，而且能提供更好的可扩展性以应对成百个处理器核的互连。

在多核或众核并行体系结构中，cache 一致性协议能很大程度地降低编程难度^[4]。一致性协议提供的全局共享地址空间减轻了管理并行程序通信的负担，从而使编程者能够更加专注于提高并行程序的计算性能。兼容大量诸如操作系统、数据库之类的基于一致性协议编写的历史代码需要在众核平台上继续提供硬件 cache 一致性支持^[5]，在 cache 一致性众核平台上，核间通信是通过对全局共享地址空间的访问隐式进行^[4, 6]，此时，片上网络传输的报文主要由 cache 一致性协议所决定^[5, 7, 8]。为了提高片上网络的效能和高效地支持一致性通信，必须要对一致性协议和片上网络进行协同设计。分析一致性协议的结构和通信特征，并基于这些特征对片上网络进行优化设计对于构建高效一致性通信片上网络、提高 cache 一致性协议和片上网络的效费比以及降低众核平台功耗具有重大意义。

1.1 研究背景

本节介绍论文的研究背景，首先论述了晶体管数量的不断增加给体系结构设计带来的机遇和挑战，以及众核时代的到来。其次分析了传统总线通信结构在众核时代面临的问题。为了解决这些问题，人们提出了片上网络通信架构。最后概述了众核结构 cache 一致性协议的设计。

1.1.1 众核时代的到来

摩尔定律一直引领着世界信息技术和芯片工艺的进步^[9]。它归纳了半导体工艺的发展规律，即大约每 18 个月集成电路单位面积的晶体管数量将翻番^[10]。当前半导体工艺已达到了单片集成 30 到 40 亿只晶体管的水平^[11]。按照半导体工业协会 SIA 制定的技术发展路线，摩尔定律在未来 5 到 10 年内依然有效^[12]。随着半导体工艺的进一步发展，ITRS 预测到 2013 年，单芯片晶体管数量将超过 80 亿^[12]。越来越多的晶体管资源在给处理器发展创造了源源不断动力的同时，也给处理器体系结构设计带来了巨大的挑战。如何有效利用芯片的海量晶体管资源设计高性能处理器成为研究热点。

传统的微处理器体系结构技术一直是沿着提升片上单处理器性能的设计道路发展，如利用超流水技术提高主频、利用超标量技术挖掘指令级并行、利用优化 cache 结构挖掘程序的局部性、设计更为复杂的分支预测器降低预测失败概率等。从计算机诞生到现在，串行计算的性价比已经提升了近 100 亿倍^[13]。但这些方法的使用也导致微处理器结构日趋复杂、难以控制，甚至出现了收益递减现象。与此同时，主频的提升与晶体管数目的增加造成处理器功耗以超线性方式飞速增加。事实上，通用单核处理器功耗高达上百瓦已经较为常见，例如 Pentium-4 3.0GHz 处理器的功耗已接近 100W^[14]。另外，随着线宽变窄、晶体管速度变快，线延迟已经成为处理器设计者必须考虑的问题^[12]。传统总线互连方式已经很难适应大规模部件互连的应用要求。上述各方面的挑战使得传统的单核性能提升方法已经很难满足现实需求。

单处理器设计面临的互连延迟、存储带宽、功耗极限等性能提升瓶颈引发了体系结构技术的深刻变革^[15]。人们开始提出采用多核处理器结构^{[16][17][18]}。随着更多的核集成到单芯片中，芯片性能得以继续提升。较之单核处理器，多核处理器具有与生俱来的优势：多核处理器可以在较低的时钟频率下达到单核处理器需要很高时钟频率才能达到的计算性能。较低的时钟频率也可以很好地满足对功耗、散热等方面限制。多核处理器的本质是采用相对较简单的多个计算内核并行工作，以提供较高的计算能力。因此，设计者在设计多核处理器时，只需设计相对较简单的计算内核，并通过一定的互连方式将其连接起来。与传统的设计单个内核的复杂处理器相比，这大大降低了设计的难度和成本，提高了设计的效率。另外，虽然单线程的性能已经不能满足摩尔定律，但是并行程序的性能完全满足摩尔定律的发展要求。正是由于多核的这些优势使它一出世就成为业界和学术界关注的焦点。

1996 年，斯坦福大学 Kunle Olukotun 主持设计了第一款主流多核芯片 Hy-

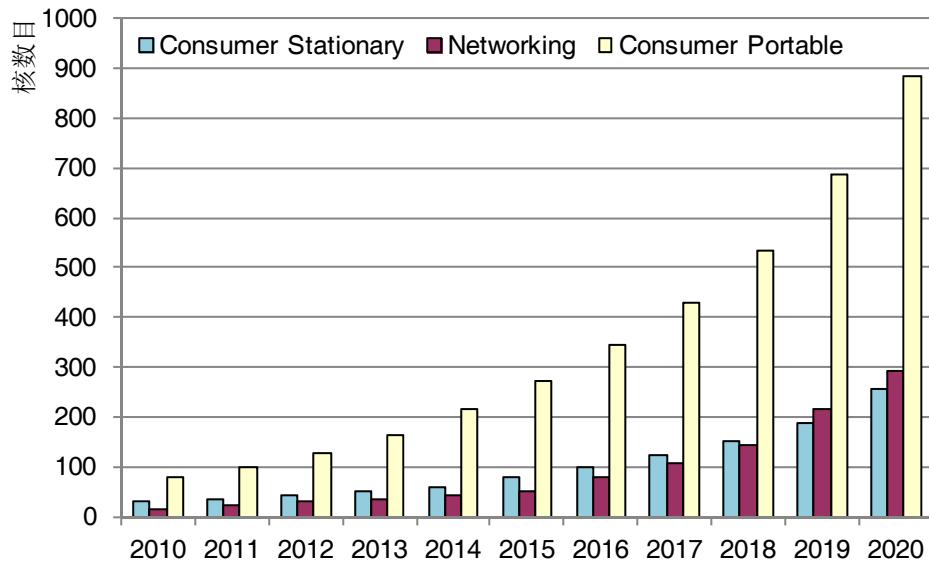


图 1.1 单芯片处理器核数量增长趋势

dra^[19]。随后，片上多核处理器逐渐进入计算机研究者的视野。但是直到占据通用处理器市场 95% 以上份额的 Intel 和 AMD 公司在 2005 年初争相推出多核芯片时，多核才被计算机用户大众所熟知。目前，多核处理器已经广泛应用于从巨型机、PC 机到手机等各个计算领域，计算技术已经全面迈入多核时代。一般来说，当单个芯片上集成处理器核数目较少时，即数个到数十个，称为多核结构；当单个芯片上集成处理器核较多时，即数十到数百个，称为众核结构。随着多核处理器的进一步发展，可以想象未来的处理器芯片上可以容纳成百甚至上千个核。这种众核结构将会成为处理器体系架构发展的必然趋势。Intel 在 2006 年秋展示了其 80 核处理器原型，标志着工业界开始从多核向众核方向发展^[20]。IBM 也推出 75 核处理器 Cyclops-64，为搭建千万亿次巨型机提供支持^[21]。Tilera 公司在 2007 年发布的 64 核处理器更是已投入实际应用^[22]。一种新的摩尔定律解释也将核的数量作为指数增长的参数^[23]。图 1.1 给出了处理器核数量随时间呈指数增长的趋势^[24]。

1.1.2 片上网络的提出

多核或众核处理器需要特别的“粘合逻辑”处理核间通信。与片外的计算机系统类似，多核片内系统一开始主要使用基于总线的通信结构。所有的核都连接到一个中央总线上，总线仲裁采用集中控制的方式。单个核通过总线向一个或多个目标核发送消息，所有连接到总线上的核监听总线并接收以它们为目标的消息。随着片上系统朝着多核化、众核化的方向发展，基于总线的通信结构面临严峻挑战^[25]：

- (1). 带宽受限：总线是一种共享介质的互连结构，无法支持一对以上的用户

同时通信，即串行访问机制限制了总线的带宽；

(2). 延迟：随着工艺特征尺寸的缩小，线延迟取代门延迟成为信号延迟的主要部分。总线结构是全局控制的，全局连线的延迟在高频条件下可能达到几个甚至几十个时钟周期，且时钟偏斜很难控制；

(3). 功耗：片上通信是系统总功耗的一个主要来源，总线结构传输数据需要对所有挂接总线的负载电容充放电，浪费了大量的功耗；

(4). 信号完整性：工作电压的下降，连线宽度和间距的缩小，使得连线上的信号对电流噪声更加敏感，总线上更多的功能部件则进一步加重了噪声；

(5). 全局同步：随着工艺特征尺寸的缩小和时钟频率的提高，全局信号的传输延迟将超过一个时钟周期，无法设计全局时钟树，且时钟树将导致大量的功耗和面积开销；

(6). 可扩展性：总线地址资源不能随着处理器核的增加而无限扩展；

(7). 可重用性：基于总线的片上系统设计中，虽然处理器核是可重用的，但通信结构却无法重用，每个新设计都需要重新设计通信结构；

因此，总线结构逐渐被基于网络的通信架构所替代，即在处理器核之间采用路由和分组交换的方式实现互连通信。

Dally 和 Towles^[1]、Hemani 和 Jantsch^[2]、Benini 和 De Michelis^[3] 提出将片上网络（Network-on-Chip, NoC）作为片上互连通信新的设计范式。片上网络通过散布全芯片的路由器构成网络，将处理器核连接起来。基于片上网络的系统实现了计算与通信的分离，处理器核构成的计算子系统完成广义的“计算”任务。片上网络构成了通信子系统，负责连接处理器核，实现计算资源之间的高速通信。

片上网络的拓扑结构具有很好的可扩展性和并行通信能力，使得通信带宽增加了好几个量级。片上网络将长连线变成路由器之间的短连线，对于功耗控制极为有利；另一方面，片上网络借鉴了通信协议中的分层思想，这就为从物理层到应用层全面控制功耗提供了可能。片上网络支持可重用设计，任意类型的处理器核通过网络接口即可与片上网络的路由器连接。片上网络的实现细节在应用层被完全隐藏，通信和计算的正交设计将重用范围从计算单元可重用扩展到计算和通信单元皆可重用的层次。节省了设计成本，提高了可靠性。片上网络可以采用全局异步局部同步（Global Asynchronous Local Synchronous, GALS）的时钟策略，时钟域、电压和频率的调整以及功耗管理都可在局部范围内进行。

虽然片上网络与片外的多处理器互连网络类似，但受面积、功耗、片内存储资源和集成技术的限制，片上网络的拓扑结构、路由的复杂性、流量控制等与片外网络相比有很大的不同，需要全面衡量设计时的限制因素。片上网络与片外网络的主要区别如下：

(1). 链路带宽：片外网络的链路带宽受限于芯片的引脚数目，但是片上网络的链路带宽不受引脚数目的限制。因此，片上网络的可用连线资源相当丰富，其支持比片外网络高一个数量级的链路带宽。

(2). 功耗和面积限制：片上网络需要与处理器核竞争使用有限的片上功耗和面积资源。因此，片上网络的功耗和面积限制比片外网络严格得多。由于缓存资源需要消耗大量的面积和功耗，因此，片上网络一般只设置有限的缓存资源。

1.1.3 众核结构 Cache 一致性协议

并行编程难度很大，但是随着单芯片上集成核数的增加，并行编程变得越来越重要^[7]。在多核或众核体系结构中，提供全局共享地址空间可以减轻程序员编写高性能并行程序的负担^[4, 6]，这是因为在全局地址空间上进行逻辑推理比在多地址空间上进行逻辑推理更为简单。在这种编程模型中，节点间通信通过对共享空间的访问隐式完成，地址的高位决定了需要访问的节点。相反的，消息传递（Message passing）编程模式显式地在节点和地址空间之间传输数据，编程者需要显式管理并行程序的通信。

在共享存储编程模式中，节点间通信通过对数据和指令的 load 和 store 操作隐式完成。逻辑上，由于所有处理器访问同一共享地址，每个处理器都能获得最新数据。但是在实际设计中，存储层次结构利用 cache 来提高访存效率和系统性能。Cache 可以减少了处理器访存的延迟，但是却使得共享存储模式各处理器的统一存储视图复杂化。当数据在多个处理器的 caches 中存在共享拷贝时，需要使用 cache 一致性协议维护存储空间的一致性视图。因此，cache 一致性协议决定了共享存储多处理器之间传输的通信。

图1.2描述了一个典型的共享存储众核处理器结构。处理器共有 36 个节点，每个节点包含一个处理核、私有一级数据和指令 cache、共享或私有二级 cache。节点还包含用于进行通信的网络接口和路由器。影响共享存储众核处理器通信的一个主要因素是所使用的 cache 一致性协议。

许多一致性协议都是“单写多读”（Single-writer, multiple readers）的变种^[6]。任意数量的节点可以将从内存中读取的数据存储在自己的 cache 中。但是，如果一个节点试图修改内存地址对应的值，它必须保证没有其它节点的 caches 中存在该存储地址的有效拷贝。因此，一次共享内存多处理器内存请求包含了数据请求、数据响应和一致性授权。在一个节点读取或者修改 cache 块之前必须获得该 cache 块的一致性授权。取决于具体 cache 一致性协议，网络其它节点可能需要对一致性授权请求作出响应。

多核或众核系统通常使用两种 cache 一致性协议：广播协议和目录协议^[6]。

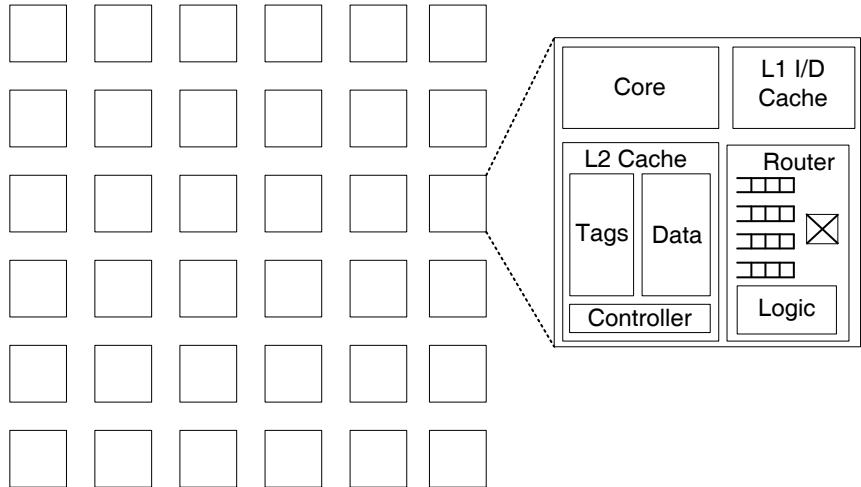


图 1.2 共享存储多处理器结构

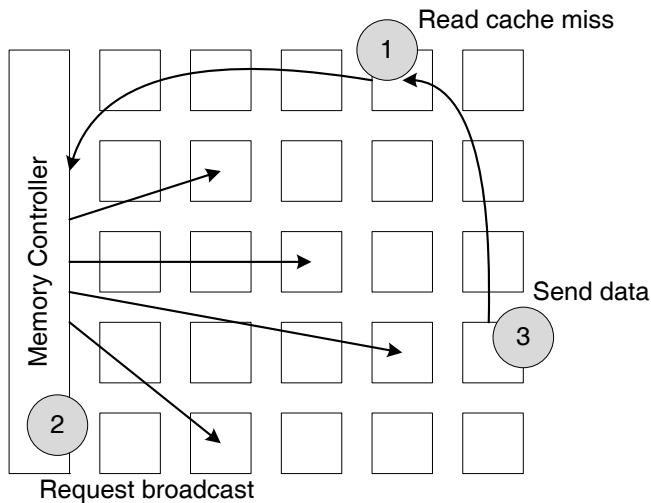


图 1.3 广播协议 Read cache miss 处理过程

这两种协议产生的网络通信的特征有所区别。广播协议需要将一致性请求发送到所有节点，因此，其通常需要网络提供很高的带宽。广播协议通常需要使用类似于总线之类的有序网络（ordered network）以保证正确处理多个同时发生的一致性事务。图1.3描述了一个 Read cache miss 的处理过程^[7]。Read cache miss 首先发往排序点（即图中的 Memory controller），然后广播至所有节点，最后接收数据。

与广播协议不同，目录协议不需要网络对一致性请求消息进行排序，因此，它可以在任意拓扑结构的网络上使用。目录协议不使用广播通信，其主要采用点对点通信。相对于广播通信，点对点通信能很大程度地降低一致性协议产生的消息数量，因此，目录一致性协议具有更好的可扩展性。由于目录中存储了 cache 块的共享者，因此不需要将一致性请求消息广播至所有节点，而只需将请求消息发送给 cache 行的当前拥有者和共享者即可。图1.4描述了目录协议一次读缺失

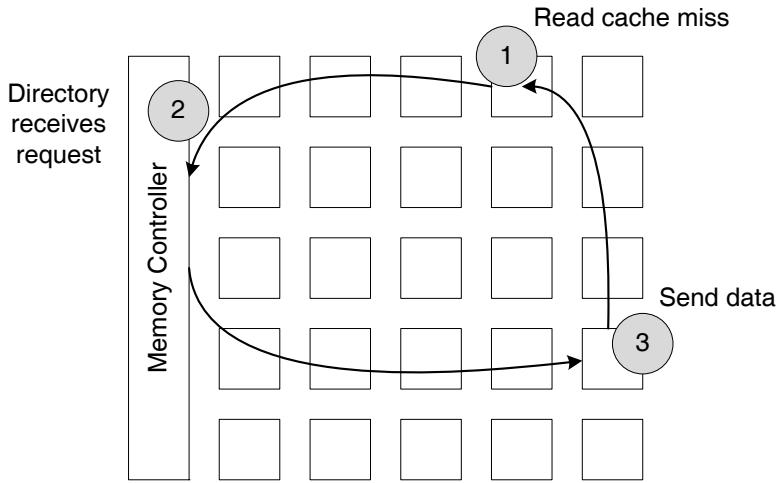


图 1.4 目录协议 Read cache miss 处理过程

的处理过程^[7]。此时 Memory controller 将 Read cache miss 请求报文直接转发给该 cache 行的当前拥有者，并且由这个拥有者提供最新的 cache 行。

目录维护系统中 cache 行存储地址在节点上交叉分布，每个地址分配一个主节点（Home node）。主节点负责对该地址一致性请求的进行排序，并对请求作出响应。由于目录一致性协议的可扩展性优于广播一致性协议，本文研究都是基于目录一致性协议开展的。但是本文基于一致性协议的通信特征对片上网络进行优化设计的方法学很容易应用到广播协议上。

1.2 国内外研究现状

本节主要回顾国内外在片上网络设计方面的相关工作。

1.2.1 国外研究现状

片上网络是目前体系结构领域的研究热点。越来越多的研究机构和芯片厂商意识到片上网络的潜力，纷纷投入其中并推动它的发展，使得片上网络成为了一个十分活跃的研究领域。本节简单回顾片上网络设计各方面的研究成果。

2000 年 3 月，巴黎第六大学的 Guerrier 等^[25, 26]提出了可扩展、可编程的集成网络（Scalable, Programmable, Integrated Network, SPIN），将分组交换技术引入了系统芯片用于片内通信。同年 11 月，瑞典皇家工学院的 Hemani 和 Jantsch 提出了“Network on a Chip”概念^[2]，定义片上网络为芯片内互连计算资源、存储资源和 I/O 资源的通信架构。2001 年 6 月，斯坦福大学的 Dally 和 Towles 在第 38 届设计自动化会议（Design Automation Conference, DAC-2001）上提出使用路由报文的方式替代各种片内连线的构想^[1]，使得片上网络成为 CMP 系统的一个研究重点。至此，体系结构学术界和工业界开始接受片上网络这一概念，并投入大量人

力物力推动片上网络的发展。

在片上网络拓扑结构方面，Balfour 和 Dally 评估了多种不同拓扑结构在片上实现时的性能，他们指出 concentration 是一种较好的减低网络延迟的设计^[27]。Kim 等将片外 Flattened Butterfly 拓扑结构运用到片上网络设计上，该拓扑结构充分利用了片上网络丰富的连线资源^[28]；Grot 等提出了片上 Express Cube 拓扑结构^[29]。与 Flattened Butterfly 拓扑结构相比，Express Cube 拓扑所需的连线资源较少。Bourduas 和 Zilic 在 mesh 网络上增加了一层 ring 拓扑以提高性能^[30]；Fallin 等基于无缓存流控评估了 ring 和层次 ring 拓扑在片上网络的性能^[31]。Mirza-Aghatabar 等评估了 mesh 网络和 torus 网络在多种流量模式和多种路由算法下的性能^[32]；Mishra 等观察到 mesh 网络的中间区域易于饱和，限制了全系统的性能^[33]。因此，他们提出了一种异构的片上网络拓扑结构，该结构为中间区域设置了更高的链路带宽；Shin 和 Kim 观察到 torus 网络相对于 mesh 网络能够获得更低的延迟，并支持更高的性能，从而针对 torus 网络提出了一种死锁恢复设计^[34]。Manevich 等提出将总线结构和片上网络结构混合使用^[35]；Das 等进一步提出了一种层次片上网络拓扑设计，局部通信通过总线完成，全局通信则通过片上网络完成^[36]；Zhao 等针对 cache 一致性协议提出将片上网络链路动态重构成总线，从而满足一致性协议局部通信低延迟的需求^[37]。

路由器微结构设计是片上网络研究的一个重点。Peh 和 Dally 对路由器流水线延迟进行建模，并提出使用猜测交叉开关分配来减少一个流水线阶段^[38]；Mullins 等在 Peh 和 Dally 工作的基础上进一步提出使用猜测交叉开关传输再减少一个流水线阶段^[39]；Matsutani 等将 Mullins 等的工作进行了推广，他们评估了多种猜测算法的性能^[40]。Kumar 等设计了一个单周期路由器结构，他们将交叉开关分配和虚通道分配合并成一个流水线阶段^[41]。Abad 等提出的 Rotary Router 将路由器简化成一个 ring 结构^[42]。Kim 基于优先级仲裁提出了一种面向维序路由算法的低开销、低延迟路由器结构^[43]。Becker 和 Dally 基于 45 nm 低功耗工艺评测了多种分配器的功耗、面积和延迟开销^[44]。Fallin 等针对无缓存路由器给出了一种低延迟的优先级分配器设计^[45]；类似的，Dimitrakopoulos 等基于置换网络给出了一种低延迟的优先级交叉开关分配器^[46]。Michelogiannakis 等针对 cache 一致性片上网络传输的大部分是短报文这个特性，提出了 Packet Chaining 分配器结构^[47]，该分配器通过尽量重用上一周期的分配器结果降低冲突。Hayenga 和 Lipasti 最近创造性地在路由器微结构设计上使用了异或逻辑的特性^[48]，他们的设计在减少一个流水线阶段的同时避免了猜测交叉开关分配带来的额外功耗和面积开销。

在缓存结构设计方面，Nicopoulos 等提出的 ViChaR 设计^[49]将动态分配多队列结构（Dynamically Allocated Multi-Queue, DAMQ）引入到片上网络设计上，其

获得了很多的功耗和面积提升；Xu 等观察到 ViChaR 设计所需的虚通道数目过多，会限制路由器的频率^[50]，因此，他们提出了一种虚通道数目固定的动态分配缓存结构；Ramanujam 等提出了一种分布式缓存结构设计，路由器上配置了输入缓存和采用 DAMQ 管理的中间缓存，其设计目标是达到输出缓存结构的性能，同时避免输出缓存结构的开销^[51]；Ahmadinia 等提出了一种在多物理端口之间共享缓存的设计^[52]；Becker 等观察到之前的 DAMQ 设计在提供 QoS 服务方面存在缺陷^[53]，为此，他们提出了一种保证每条虚通道至少拥有一定数量缓存的设计。Michelogiannakis 等提出的 Elastic Buffer 设计将路由器中的触发器用于存储报文，从而移除了路由器上的输入缓存^[54]；Kodi 等提出将片上长连线的中继器（repeater）用于存储报文，从而降低了对输入缓存的需求^[55]。Kim 等观察到片上缓存大部分时间处于空闲状态，因此提出使用动态功耗管理方式将这些缓存的逻辑门关闭以降低功耗^[56]。

在单播路由算法设计方面，Hu 和 Marculescu 提出的 DyAD 设计综合了自适应路由和确定性路由的优点^[57]；Seo 等提出的 O1Turn 路由算法综合使用 XY 维序路由和 YX 维序路由保证最坏情况下的性能^[58]；Singh 等提出的 GOAL 路由算法通过非最短路由实现 torus 网络的全局负载均衡^[59]。Kim 等提出了一种采用超前路由计算和预选择策略的低延迟自适应路由算法设计^[60]。Li 等提出的 DyXY 在邻居节点间使用专门连线来传输状态信息以做出更为有效的输出端口选择^[61]；Ascia 等提出的 Neighbors-on-path 设计则考查邻居节点的相邻节点，即两跳之外的节点的状态^[62]；Gratz 等提出的 Regional Congestion Awareness 设计通过一个网络拥塞信息传播网络获得全局网络状态^[63]；Ramanujam 和 Lin 提出将报文目标信息融入到输出端口的选择过程中^[64]。Rodrigo 等提出的 bLBDR 通过在相邻区域之间静态配置连接位来隔离多区域的报文路由过程^[65]。Zhang 等以 XY 维序路由算法为基础提出了一种能够容忍一条链路错误的路由算法设计^[66]。上述路由算法都是针对通用 CMP 平台设计的，Kinsky 等针对特定应用程序提出了一种设计显式无死锁路由算法的数学模型^[67]。

在多播路由算法设计方面，Lu 等使用基于路径的多播路由，他们的设计需要在多播报文传输前先发送路径建立请求和接收路径建立回复消息^[68]；Enright Jerger 等提出的 VCTM 多播路由算法引入了虚拟多播树的概念^[69]；bLBDR 路由算法使用小区域内部的广播实现多播操作^[65]；Wang 等提出的 RPM 路由算法主要研究提高网络带宽的有效使用率^[70]；基于 bLBDR 路由算法中的区域隔离机制，Wang 等将 RPM 算法扩展到非规则区域^[71]；Abad 等提出的 MRR 是一种基于 Rotary Router^[42] 的多播路由算法^[72]；Krishna 等提出的 Whirl 设计主要面向广播和密集多播操作^[73]；Kang 等采用动态报文切分机制支持长多播报文的无死锁路

由^[74]。

在流控机制设计方面，Peh 和 Dally 提出了 Flit-Reservation 流控机制能够避免虫孔交换网络由于信元往返延迟带来的缓存使用限制^[75]。Kumar 等提出的 Express Virtual Channel 通过预先分配多个路由器上的输入缓存降低报文传输延迟^[76]；在此基础上，Kumar 等进一步提出了 Token Flow Control^[77]，该设计通过向周围邻居节点广播路由器当前空闲输入缓存数量，报文发送能 bypass 掉很多中间路由器的流水线阶段。Lu 等提出的 Layered Switching 机制^[78] 将虫孔交换网络的长报文划分成多个分组，并且在这些分组内部保持交叉开关分配器的结果不变，从而降低交叉开关分配的冲突。Layered Switching 机制可以被视为混合使用虫孔交换和虚切通交换机制；Enright Jerger 等提出的 Hybrid Circuit Switching 设计混合使用了电路交换和包交换机制^[79]；Samman 提出的 Wormhole Cut-through 设计允许不同报文在虚通道内部以切片粒度混合存储^[80]。Concer 等评估了多种链路层流控机制的性能，包括 on/off 机制、信元机制和 ACK/NACK 机制^[81]。Chen 等针对 torus 网络提出了一种名为“Critical Bubble Scheme”的死锁避免流控机制^[82]；类似的，Joshi 等提出的 Prevention Flow Control 设计也能在 torus 网络上避免死锁^[83]。

在片上网络与 cache 一致性协议协同设计方面，Cheng 等观察到不同 cache 一致性消息对延迟和带宽的敏感程度不同^[84]。因此，他们提出采用异构连线分别传输不同类型的消息，将延迟敏感的消息通过低延迟连线传输，将带宽敏感的消息通过高带宽连线传输；Muralimanohar 和 Balasubramonian 将 Cheng 等的工作扩展到基于片上网络的大规模 NUCA cache 结构上^[85]。Eisley 等提出将目录 cache 一致性协议的目录信息存储在片上网络路由器上，从而减少 cache 读写事务的延迟^[86]。Agarwal 等在片上网络层实现消息的排序，从而支持在 unordered 网络上实现广播 cache 一致性协议^[8]；在这项工作的基础上，Agarwal 等进一步提出在路由器上设置过滤器（filter）消除一些不必要的监听消息^[87]；基于类似的思想，Enright Jerger 使用过滤器消除了粗粒度目录 cache 一致性协议中冗余的 cache 行作废消息^[88]。

片上网络研究还包含其它一些重要的领域，如应用程序映射^[89–92]、QoS 服务^[93–96]、网络演算模型^[97, 98]等。一些底层技术的创新进一步改变了片上网络体系结构的蓝图^[99]。互连电路最近新发展的低幅度全局信号（Low-Swing Global Signaling）技术能极大地改变片上网络的设计并催生新的体系结构^[100, 101]。一些颠覆性的互连技术也开始用于片上通信，包括光互连^[102, 103]和 RF 互连^[104]，这也显著改变片上网络体系结构。此外，三维芯片集成技术^[105–107]和电容 I/Os^[108]使得片上网络的带宽能提高几个量级，对片上网络设计也产生了巨大的影响。

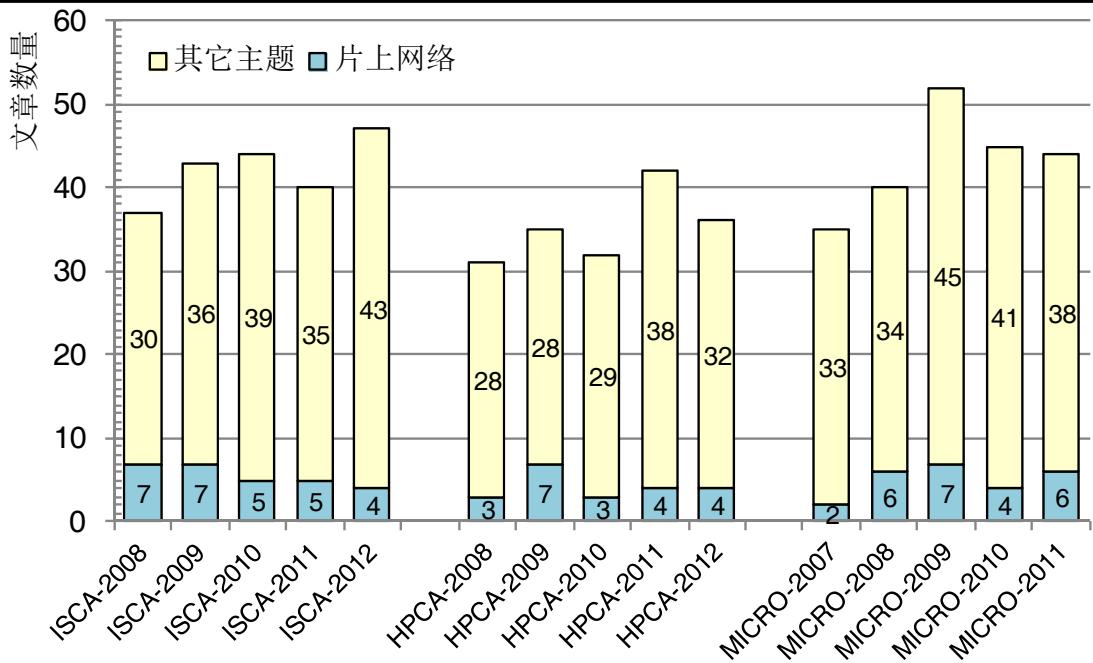


图 1.5 近五年在体系结构顶级会议上发表的片上网络文章数

学术界已投片原型系统和工业界产品包括德州大学奥斯汀分校的 TRIPS^[109]、马萨诸塞大学的 aSoC^[110]、瑞典皇家工学院的 Nostrum^[111]、意大利博洛尼亚大学的 Xpipes^[112]、以色列技术学院的 QNoC^[113]、丹麦科技大学的 MANGO^[114]、麻省理工学院的 16 核验证芯片^[115]、飞利浦公司的 Ethereal^[116]、IBM 的 Cell^[17]、ST Microelectronics 的 STNoC^[117]、Intel 公司的 80 核单芯片处理器 TeraFLOPS^[20]、Intel 公司的 48 核 Single-Chip Cluster^[118]、Tilera 公司的 64 核 TILE64^[22]、Tilera 公司的 80 核 TeraScale^[119] 等。

自 2007 年起，ACM 和 IEEE 已举办了六届片上网络国际研讨会（ACM/IEEE International Symposium on Networks-on-Chip, NOCS）^[120]，片上网络已成为一个相对独立且重要的研究领域。这种重要性也体现在最近五年发表在体系结构领域三大顶级国际会议上的片上网络文章数上。图 1.5 给出了最近五年发表在 ISCA、HPCA 和 MICRO 国际会议上的文章总数和其中的片上网络文章数，该图中的部分文章是来自国内研究机构。在近五年的三大体系结构顶级会议上发表的片上网络文章平均占全部文章的 12.3%，在某些年份的会议上甚至有 20% 的文章是片上网络方面的研究。例如，HPCA-2009 会议共录用了 35 篇文章，其中 7 篇是片上网络方面的文章。这些数据都说明了当前片上网络正成为体系结构领域的一个关键问题和研究热点。

1.2.2 国内研究现状

国内片上网络的研究方兴未艾，还没有较完备的片上网络设计工具的公开报道。以关键词“片上网络”在万方数据库中进行检索，截至 2012 年 10 月共有 471 篇文献，其中 316 篇期刊论文，52 篇会议论文，21 篇博士论文和 82 篇硕士论文。主要研究单位有清华大学^[121, 122]、北京大学^[123]、复旦大学^[124]、中国科学技术大学^[125, 126]、中科院计算所^[127, 128]、哈尔滨工业大学^[129]、浙江大学^[130]、电子科技大学^[131, 132]、西北工业大学^[133]、西安交通大学^[134]、北京邮电大学^[135]、北京科技大学^[136]、南京大学^[137]、南开大学^[138]、湖南大学^[139]、合肥工业大学^[140] 和国防科技大学^[141, 142] 等。值得注意的是这里列出的文章数目不包括国内研究机构以英文在国际会议或国际期刊发表的文章。

清华大学林世俊等提出了一种分布式同步策略实现全局异步局部同步 (GALS) 的片上系统^[121]。清华大学 Zheng Li 等在分析 cache 一致性事务报文关键性的基础上提出了一种优先级仲裁机制，它通过加速关键报文的传输来提升性能^[143]。北京大学王宏伟等提出了一种层次化的片上网络设计方法，能够有效提高系统性能，降低硬件实现的开销，同时满足一定的服务质量要求^[123]。复旦大学虞志益等在 ISSCC-2012 会议上公布了一个 16 核采用片上网络互联的验证芯片^[124]。中国科学技术大学朱晓静等针对片上网络节点数量多、距离近、物理实现复杂度受限等特点，提出了一种新的 Xmesh 拓扑结构及路由算法^[125, 126]。中科院计算所张磊等提出一种端到端反馈的随机路由算法，该算法具有低延迟和低功耗特点，并能提供高可靠性的片上通信^[127]。中科院计算所付斌章等提出的 Abacus 转向模型能够动态适应网络负载的变化提供完全自适应路由^[144]。哈尔滨工业大学付方发等采用了分层逐级抽象的建模方法，构建了适用于多种流量分布的片上网络性能评估平台^[129]。西北工业大学荆元利基于“通用 DSP 处理器 - 龙腾 D1”软核，系统的研究了片上网络结构，提出了采用 NetCMP 结构的“龙腾 DN”处理器模型^[133]。国防科学技术大学刘翔远等研究多核 SoC 通信网络，根据多核 DSP 的通信特点，提出了功耗与面积优化的片上网络设计方法^[141]。国防科学技术大学赖明澈等提出了一种具有拥塞避免能力的 DAMQ 缓存结构设计^[145]。国防科学技术大学钱悦等针对片上网络的演算模型进行了大量的研究^[142]。国防科学技术大学石伟等提出了一种能在多物理端口之间动态共享缓存的设计^[146]。

当前，国内对片上网络相关领域的研究正在逐步展开。表 1.1 列出了近三年获得自然科学基金资助的项目及其承担单位^[147]，主要的研究方向有拓扑结构、路由算法、低功耗、可重构设计、容错技术、片内存储优化、可测试性、光互连等。

表 1.1 近三年有关片上网络的国家自然科学基金项目

| 立项时间 | 项目名称 | 研究机构 |
|------|---|--|
| 2010 | 基于全局通信管理的 NoC 低功耗容错机制研究 超多核处理器片上网络性能模型研究 片上网络异构路由器关键问题研究 基于 NoC 的同构多核 SoC 并发在线测试研究 低功耗光片上网络的设计与研究 片上网络互联的多核系统中的区域化任务映射算法 | 南京航空航天大学 清华大学 中山大学 西安交通大学 西安电子科技大学 浙江大学 |
| 2011 | 片上网络演算模型及性能分析研究 NoC 高性能互连结构设计方法研究 多核芯片异步片上网络的微电路和建模研究 三维片上网络存储体系结构研究 片上多核处理器验证理论与关键技术 聚合物电光开关 S+C+L 超宽波段光谱平坦化及失效模式研究 专用三维片上网络体系结构综合技术研究 众核体系结构中的渗透式延迟容忍方法研究 面向成品率的三维片上网络 TSV 容错技术研究 | 国防科技大学 西安电子科技大学 广东工业大学 南京大学 国防科技大学 吉林大学 南京航空航天大学 国防科技大学 合肥工业大学 |
| 2012 | 应用片上网络的混合光子 - 等离子体路由研究 基于柏拉图立体多级裂变模型的三维 NoC 拓扑结构的研究 光控光子片上网络结构与光子模块研究 片上网络的通信性能 - 功耗联合建模与优化设计 基于自相似理论的 NoC 网络流量特征关键技术研究 可重构多核处理器设计方法及其关键技术研究 片上网络虚拟化关键技术研究 | 中科院西安光机所 天津工业大学 浙江大学 电子科技大学 合肥工业大学 厦门大学 中科院计算所 |

1.3 课题研究的目标和意义

基于 cache 一致性协议的结构和通信特征对片上网络路由算法和流控机制进行优化至关重要，主要有以下四方面的原因：

第一，路由算法和流控机制是片上网络设计中很重要的两个方面。良好设计的路由算法应该能有效避免网络中的拥塞，而流控机制则应该尽可能地提高网络缓存和链路的利用率，以支持更高的性能。

第二，在 cache 一致性众核结构上，片上网络传输的通信主要由一致性协议所决定。因此，一种高效的片上网络设计应该充分考虑一致性协议的特性，包括一致性协议的层次结构、通信报文长度分布和一致性通信模式等。层次一致性协

议结构基于应用程序的并行度有限这个特性将众核结构划分成多个局部一致性区域以提高一致性协议的性能和可扩展性。路由算法设计应该充分考虑这种层次结构，在高效支持局部通信的同时消除多个局部一致性区域之间的相互干扰。**Cache** 一致性协议传输的报文大部分长度较短，片上网络路由算法和流控机制设计应该充分利用这点属性。**Cache** 一致性协议需要使用包括多播和归约通信在内的聚合通信，为了防止这些聚合通信成为系统性能的瓶颈，应该在片上网络设计中对它们提供硬件支持。

第三，面积、功耗和频率的限制使得 **cache** 一致性片上网络只能配置有限的虚通道数和缓存资源。有限的虚通道数和缓存资源对片上网络设计提出了新的挑战。如何优化设计路由算法和流控机制，使它们能充分利用这些有限而又宝贵的虚通道和缓存资源来更好地支持 **cache** 一致性协议通信是一个亟待解决的问题。

第四，很多设计路由算法的死锁避免理论和流控机制都是针对片外网络提出的，但是片上网络与片外网络设计面临着不同的面积和功耗限制，片上网络能够利用的资源也与片外网络有很大不同。已有的路由算法死锁避免理论和流控机制并没有考虑 **cache** 一致性片上网络的通信特征和设计约束，它们在处理短报文和混合长度报文时存在一些局限性，这些局限性严重制约了 **cache** 一致性片上网络的性能。因此，需要对死锁避免理论和流控机制进行扩展和优化以更为高效地支持 **cache** 一致性通信。

本文研究的目标是在分析 **cache** 一致性协议的结构和通信特征的基础上，对片上网络路由算法和流控机制进行优化，从而提高 **cache** 一致性片上网络的效费比。具体而言，本文期待解决如下问题：

第一，在层次 **cache** 一致性协议中，或多应用程序同时运行的情况下，路由算法的设计一方面需要高效支持应用程序内部的通信，另一方面需要实现多应用程序之间的动态隔离。本文希望提出一种路由算法满足这两方面的要求。

第二，基于已有的死锁避免理论设计的完全自适应路由算法需要使用保守虚通道分配策略，该策略严重限制了片上网络处理大部分报文是短报文的 **cache** 一致性协议通信的性能。本文希望通过优化流控机制对这些死锁避免理论进行扩展，进而设计更为高效的完全自适应路由算法支持这些 **cache** 一致性通信。

第三，已有的 torus 网络死锁避免理论不能高效处理 **cache** 一致性通信中混合长度报文的传输，基于这些理论设计的路由算法在缓存资源受限的片上网络中性能较差。本文希望对这些 torus 网络死锁避免理论进行扩展从而在有限片上缓存资源条件下，更高效地传输 **cache** 一致性协议中混合长度的报文。

第四，目录 **cache** 一致性协议通过多播通信发送 **cache** 行作废消息，通过归约通信收集这些作废消息的 acknowledgement (ACK) 消息。为了防止这些聚合通

信成为系统性能的瓶颈，需要在片上网络中集成对归约和多播通信的硬件支持。具体而言，本文希望对归约 ACK 消息组合消除冗余网络操作，进而降低网络负载。同时，本文希望提出能高效利用网络带宽和缓存资源的多播路由算法。

由于片上网络传输的通信由 cache 一致性协议决定，将 cache 一致性协议和片上网络进行协同设计是体系结构研究的一个热点。这种协同设计的理念不仅能提高了片上网络传输 cache 一致性通信的效费比，同时也能够给 cache 一致性协议设计带来了一些启发。面向 cache 一致性通信优化路由算法和流控机制提高了片上网络的性能，同时也降低了对缓存和虚通道资源的需求。Cache 一致性协议的设计应该充分考虑片上网络缓存资源和虚通道数目受限的条件。本文针对负载整合工作模式提出的基于目标的输出端口选择策略可以被应用到许多其它拓扑结构上。本文所设计的归约消息组合框架不仅可以用于支持目录 cache 一致性协议，也可以用于支持令牌 cache 一致性协议和栅栏同步通信。本文的研究对完全自适应路由算法的死锁避免理论和 torus 网络的死锁避免理论进行了扩展，这些死锁避免理论扩展不仅可以应用于 cache 一致性片上网络，也可以应用于许多其它场合，包括消息传递的片上网络和巨型机系统的片外通信网络。因此，本文的研究不仅在工程上获得了系统性能的提升和硬件需求的降低，它也具备一定的理论价值。

本文研究的支撑项目是 973 国家重点科研项目“计算系统虚拟化基础理论与方法研究”(2007CB310901) 和国家自然科学基金项目“片上众核集群体系结构关键技术研究”(61070037)。

1.4 研究内容和创新点

1.4.1 主要研究内容

本文研究了众核结构 cache 一致性片上网络路由算法和流控机制优化关键技术，主要研究内容包括四个方面：

(1). 面向负载整合工作模式的路由算法

摩尔定律不断增加片上集成的核数，但是挖掘应用程序并行性依然很难。因此，多个应用程序将会同时运行在一个众核平台上。这种负载整合工作模式对片上网络路由算法设计提出了新的要求：一方面，路由算法应该能提供良好的适应性以避免网络拥塞。另一方面，路由算法应该能动态隔离多个同时运行的应用程序。已有设计不能很好地满足这些要求。局部自适应路由算法只能感知邻居节点的拥塞，这种短视性限制了它对网络拥塞的避免能力。为了更为有效地避免网络拥塞，全局自适应路由算法使用一个拥塞信息传播网络获得邻居节点之外的网络状态。但是在负载整合模式下，全局自适应路由算法的输出端口选择存在应用程

序内部和应用程序之间的干扰，这些干扰耦合了多个同时运行的应用程序，使得它们之间相互影响，从而降低了应用程序的性能。

本文试图解决已有设计在负载整合工作模式下的局限性。本文提出了基于目标的自适应路由算法，该算法采用一个低开销的拥塞信息传播网络获得本地和非本地的拥塞信息，从而能有效地避免邻居节点之外的网络拥塞。通过将目标信息集成到输出端口选择中，所设计的路由算法消除了应用程序内部和应用程序之间的干扰，从而能动态隔离多个同时运行的应用程序。与已有设计相比，基于目标的自适应路由算法在许多配置下都获得了更好的性能，同时它能在中等或较高负载情况下降低网络的能量延迟积。

(2). 面向 cache 一致性通信的完全自适应路由算法

由于芯片面积、功耗和频率的限制，片上网络只能配置少量的虚通道，虚通道数目受限环境给完全自适应路由算法设计提出了许多新的挑战。一种高效的路由算法应该具有较高的虚通道利用率，从而获得较好的性能。同时，在虚通道受限环境下，路由算法应该尽量维持路由灵活性，从而支持较高的报文发送适应性以有效避免网络拥塞。基于已有的死锁避免理论设计的完全自适应路由算法需要使用保守虚通道分配策略：只有空虚通道才能被重新分配。

这个保守策略在虚通道受限环境下严重限制了完全自适应路由算法的性能。因此，本文提出了全报文发送虚通道分配策略，该策略允许非空虚通道在其空闲缓存数足够容纳整个报文时被重新分配。全报文发送策略是针对 cache 一致性片上网络提出的：在 cache 一致性通信中，大部分报文的长度较短。本文证明了如果完全自适应路由算法在采用保守虚通道分配策略时没有死锁，则采用所提出的全报文发送策略一样也不会存在死锁。因此，全报文发送策略对之前的死锁避免理论进行了重要的扩展。为了在虚通道数目受限的环境中有效利用全报文发送策略，本文进一步给出了一种在较低硬件开销情况下能提供较高路由灵活性的完全自适应路由算法设计。

(3). 面向 torus 片上网络的切片气泡流控机制

Torus 网络的可扩展性优于 mesh 网络，同时其点对称特性有利于实现全局负载均衡，但是由于回绕链路的存在，torus 网络的死锁避免比较困难。另一方面，cache 一致性片上网络需要同时传输长报文和短报文，当前存在的 torus 网络死锁避免理论不能高效地处理这种混合长度报文的传输。一种传统的设计需要使用两条虚通道避免死锁，两条虚通道提高了对缓存资源的需求，同时也增大了分配器的规模，降低了路由器的频率。虚切通网络存在一些只使用一条虚通道的优化设计，但是这些优化设计需要将每个短报文视为长报文，从而降低了缓存的利用率，给性能带来了负面影响。

针对这些设计的局限性，本文提出了 torus 网络切片气泡流控死锁避免理论，该理论通过在虫孔交换 ring 网络上维持一个空闲缓存单元避免死锁。切片气泡流控只需要使用一条虚通道，因此降低了对缓存资源的需求，同时支持更高的路由器频率。切片气泡流控不需要将短报文视为长报文，因此，与虚切通网络的优化设计相比，切片气泡流控能够支持更高的缓存利用率。基于这个死锁避免理论，本文给出了两种实现：本地切片气泡策略和关键切片气泡策略，它们在缓存资源需求和额外连线资源开销方面进行了权衡，这两种切片气泡策略都获得了比已有设计更高的性能。

(4). 高效支持 cache 一致性协议归约和多播通信的技术

聚合通信大都处于应用程序的关键路径上，它们对许多体系结构和编程模式的性能和正确性具有关键影响，这种重要性使得许多巨型机设置了专用的聚合通信网络。类似的，为了防止这些重要的聚合通信成为系统性能的瓶颈，众核体系结构必须要对它们提供硬件支持，但是片上网络可能无法承受设置专用的聚合通信网络。因此，本文试图在已存在的片上网络中集成对聚合通信的高效硬件支持。在片上网络中对多播通信或一对多通信提供硬件支持已经取得了相当的网络吞吐率提升，同时也获得了一定的功耗下降。本文研究了对归约通信或多对一通信的硬件支持。作为一个案例，主要研究目录 cache 一致性协议中一条 cache 行被提升到修改状态时需要收集的 acknowledgement (ACK) 消息。这些 ACK 消息的格式类似，同时它们携带的信息相对简单。因此，可以在不损失信息的前提下，将它们在网络中进行组合降低负载。

本研究主要包括两个方面：首先提出了一种消息组合框架支持 ACK 消息的归约操作。通过将 ACK 消息组合，该消息组合框架不仅能在低到中等负载下降低报文传输延迟，同时也能提升网络饱和吞吐率。这个框架结构只带来较低的额外硬件开销，它可以与多种多播路由算法一起使用。其次设计了均衡自适应多播路由算法，该算法在不同维度间均衡地利用了缓存资源，同时也高效地利用网络带宽，从而进一步提升了网络饱和吞吐率。

1.4.2 创新点

本文系统深入地研究了 cache 一致性片上网络设计，做出了很多开创性的工
作。主要创新点如下：

(1). 提出了一种面向负载整合工作模式的路由算法

层次 cache 一致性协议结构和应用程序有限的并行度导致多应用程序会同时运行在一个众核计算平台上，这种负载整合工作模式需要路由算法在提供足够适应性的同时维持多应用程序之间的动态隔离性。本文重点研究了负载整合模式下

自适应路由算法的设计。本研究的主要创新点包括：1. 分析了已有的局部自适应路由算法、路径上邻居自适应路由算法和区域拥塞感知自适应路由算法在负载整合模式下的局限性，并进一步提出了基于目标的自适应路由算法，该算法能够提供良好的适应性避免网络拥塞，同时也能动态隔离多个同时运行的应用程序；2. 研究了应用程序内部干扰和应用程序之间干扰对路由算法性能的影响。实验结果表明，路由算法所考虑的网络状态信息对其性能至关重要，尤其在负载整合工作环境下；3. 设计了一个低开销的拥塞信息传播网络，这个网络只引入 3.125% 的额外开销。通过使用这个拥塞信息传播网络，基于目标的自适应路由算法获得了本地和远端的网络状态信息，从而能有效避免网络拥塞。

该研究成果的部分内容发表在体系结构顶级会议第 38 届国际计算机体系结构大会（International Symposium on Computer Architecture, ISCA-2011）上^[148]，完整内容发表在体系结构旗舰期刊 IEEE Transactions on Computers 上^[149]。

(2). 提出了一种面向 cache 一致性通信的完全自适应路由算法

基于之前的死锁避免理论设计的完全自适应路由算法只能使用保守虚通道分配策略：只有空虚通道才能被重新分配给报文使用。该策略在虚通道数目受限的 cache 一致性片上网络上严重限制了路由算法的性能。本文主要研究了在保持网络无死锁情况下的虚通道分配策略和虚通道受限环境下的路由算法灵活性。本研究的主要创新点包括：1. 提出了全报文发送虚通道分配策略，该策略允许非空虚通道被重新分配，同时也不会带来网络死锁。全报文发送策略在虚通道数目受限的 cache 一致性网络中能显著提升完全自适应路由算法的性能；2. 证明全报文发送策略可以被许多已有的完全自适应路由算法采用，它是之前的死锁避免理论的一种重要扩展；3. 观察到在虚通道受限的环境中提供足够的路由灵活对性能非常重要。因此，本研究基于全报文发送策略进一步设计了一种完全自适应路由算法，它能在较低硬件开销情况下提供较高的路由灵活性。

该部分研究成果发表在体系结构顶级会议第 18 届国际高性能计算机体系结构大会（International Symposium on High Performance Computer Architecture, HPCA-2012）上^[150]。

(3). 提出了面向 torus 片上网络的切片气泡流控机制

Torus 网络的死锁避免相对比较复杂。之前的 torus 网络死锁避免机制要么需要采用两条虚通道，要么需要将短报文视为长报文。这两种类型的设计在处理 cache 一致性通信时都具有一定的局限性。使用两条虚通道增加了对缓存资源的需求，同时降低了路由器的频率；将短报文视为长报文在缓存资源有限的片上环境中严重制约了性能。本文主要研究了在使用一条虚通道实现 torus 网络死锁避免的条件下提升缓存资源的利用率。本研究的主要创新点包括：1. 分析了已有 torus

网络死锁避免机制的局限性，并指出它们无法在 cache 一致性片上网络上获得较高的性能；2. 证明了在虫孔交换 torus 网络中，死锁避免可以通过在每个 ring 上维持一个空闲缓存单元实现。基于这个观察，本研究提出了切片气泡流控理论。切片气泡流控解决了之前研究的局限性，它只需使用一条虚通道，同时其不需要将网络中的短报文视为长报文。3. 基于切片气泡流控理论，本研究给出了两种不同的实现，这两种实现在最低缓存资源需求和额外连线资源开销方面进行了权衡，它们相比于之前的设计都获得较大的性能提升。

(4). 提出了一种高效支持 cache 一致性协议归约和多播通信的技术

Cache 一致性协议需要使用包括归约和多播在内的聚合通信，对这些聚合通信提供硬件支持能获得性能的提升和功耗的下降。本文主要研究了对目录 cache 一致性协议中的多播 cache 行作废消息和归约 cache 行作废 acknowledgement (ACK) 消息的硬件支持。本研究的主要创新点包括：1. 针对目录 cache 一致性协议提出了一种高效的归约消息组合框架，该组合框架可以与多种多播路由算法一起使用。2. 通过将传输中的 ACK 消息组合，该框架不仅能在低到中等网络负载情况下降低报文平均传输延迟，同时也能支持更高的网络饱和吞吐率。此外，该消息组合框架还能降低网络的能量延迟积。3. 设计了均衡自适应多播路由算法，该算法均衡了不同维度间的缓存资源，并且高效使用了网络链路带宽，它能进一步提升网络吞吐率。

该部分研究成果发表在体系结构顶级会议第 18 届国际高性能计算机体系结构大会 (International Symposium on High Performance Computer Architecture, HPCA-2012) 上^[151]。

1.5 论文组织结构

本文紧紧围绕基于 cache 一致性协议的结构和通信特征对片上网络路由算法和流控机制进行优化，本文共分为七章。论文的组织框图及本文提出的设计与 cache 一致性协议的关系如图1.6所示。

第一章为绪论，介绍了论文的研究背景和国内外研究现状，指出了论文研究的目标和意义，简述了论文的主要研究内容、创新点和组织结构。

第二章介绍了论文研究的技术背景，包括片上网络相关背景知识、cache 一致性协议对片上网络设计的影响，以及本论文使用的实验模拟环境和性能评估方法。

第三章研究了面向负载整合模式下路由算法的设计，并提出了一种能够提供良好适应性和动态隔离性的自适应路由算法。

第四章研究了 cache 一致性片上网络中完全自适应路由算法的设计，提出了

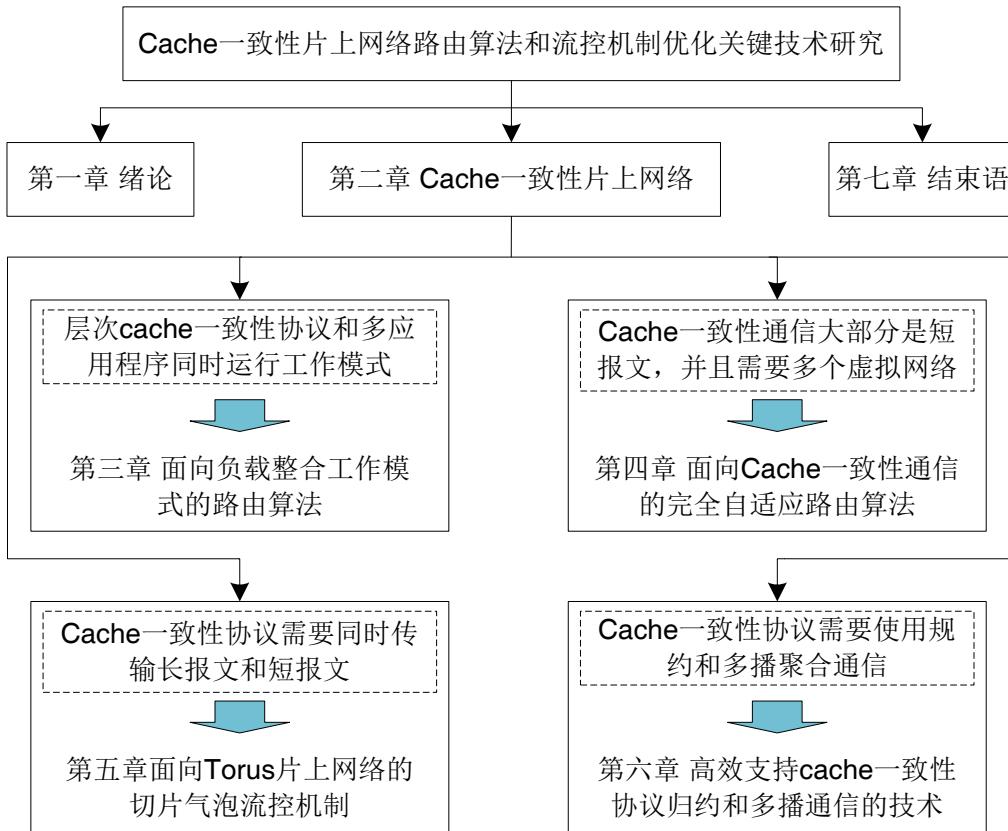


图 1.6 论文组织框图

全报文发送虚通道分配策略，并在该策略的基础上设计了一种高效完全自适应路由算法。

第五章研究了 cache 一致性 torus 片上网络的死锁避免机制，并提出了切片气泡流控死锁避免机制。

第六章研究了如何高效支持 cache 一致性协议中的聚合通信，并提出了一种支持归约通信的消息组合框架和均衡自适应多播路由算法。

第七章总结了全文，并展望未来的工作。

第二章 Cache 一致性片上网络

本章主要论述 cache 一致性片上网络设计的相关背景知识。首先介绍了片上网络设计的各个方面，包括拓扑结构、路由算法、流控机制、路由器微结构等。之后论述了 cache 一致性协议对片上网络设计带来的影响。最后介绍了本文使用的模拟器平台和实验评估方法。

2.1 片上网络背景知识

片上网络的拓扑结构决定了节点和链路的物理分布以及它们之间的连接关系。路由算法在一个拓扑结构中计算报文从源节点到目标节点的传输路径，流控机制负责网络缓存和链路的分配，路由器微结构具体实现路由算法和流控机制。本节讨论这些方面的背景知识。

2.1.1 拓扑结构

拓扑结构对整个网络性能和成本的影响显著。拓扑决定了报文传输需要经过的网络跳数及跳与跳之间的链路长度。拓扑同时决定了节点间可用的总路径数和网络所能支持的有效带宽。

为了方便芯片布局布线，片上网络大都采用较简单的拓扑结构。图2.1给出了三种常见的拓扑结构：ring、二维 mesh 和二维 torus。Ring 结构在当前的商用处理器中被采用，包括 Cell^[17] 和 Ivy Bridge^[152]。Ring 的特性是其结构非常简单、实现路由算法和流控机制相对容易。但是由于 ring 的节点平均跳数较高，其可扩展性较差^[153]，因此，ring 只能适合于核数相对较少的情况。

二维 mesh 的特点是：1、易于进行金属层布局布线；2、实现无死锁路由相

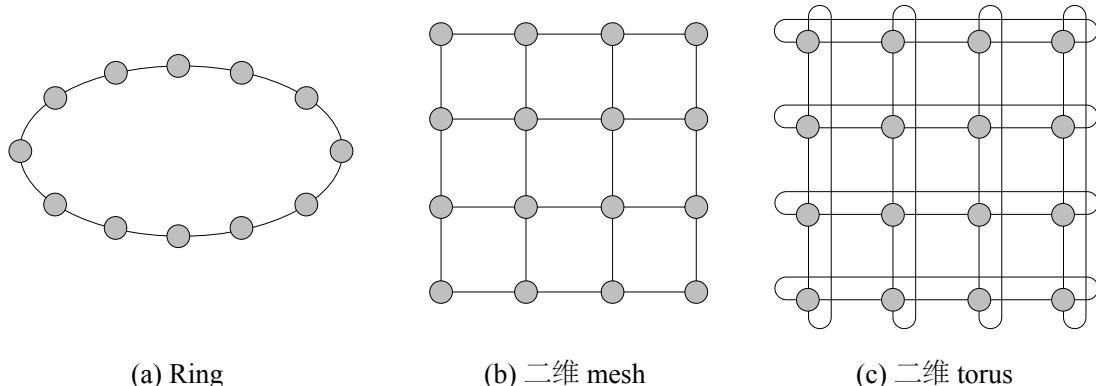


图 2.1 片上网络常用的三种拓扑结构

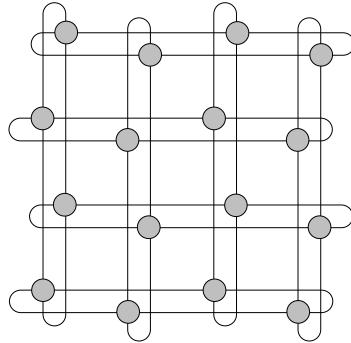


图 2.2 褶曲 torus 网络

对简单；3、可扩展性优于 ring 网络。这三方面的原因使得大量产品采用 mesh 网络^[22, 109]，同时它也是在研究中使用较多的一种拓扑结构。但是 mesh 在网络规模进一步增大时会面临着中间区域快速饱和的情况^[33]，从而影响整个系统的性能。

相对而言，二维 torus 的节点对称性有效地消除了中间区域快速饱和的问题，同时其网络平均跳数比 mesh 和 ring 要低^[34, 59]。因此，二维或更高维的 torus 在片外网络中大量采用，比如说 Cray T3E^[154] 和 Blue Gene/L^[155] 都采用三维 torus 网络。图2.1(c)所示的二维 torus 的长回绕链路可能会降低网络的频率，同时网络链路长度不一致会增加流控机制的复杂性。为此，torus 在芯片布局时会进行褶曲处理使链路长度相同，即采用褶曲 (folded) torus 结构，如图2.2所示。本文后续章节的 torus 网络默认使用褶曲结构。

除了上述三种拓扑外，人们还提出了许多其它片上网络拓扑结构，包括 concentrated mesh^[27, 156]、flattened butterfly^[28]、Express Cube^[29]、物理^[157] 或虚拟 express channel^[76] 以及各种针对专门应用程序定制的拓扑等。本文第3章、第4章和第6章采用二维 mesh 网络，第5章采用 ring 和二维 torus 网络。

2.1.2 路由算法

在确定了拓扑结构之后，路由算法负责为报文从源节点到目标节点选择一条传输路径。本节以二维 mesh 网络为例介绍路由算法的分类和死锁等问题。

路由算法有多种分类标准。根据报文传输路径长度可以分成非最短路由和最短路由，如图2.3所示。非最短路由算法选择的传输路径不一定处于报文源节点和目标节点确定的最小象限内。Valiant 算法是一种典型的非最短路由算法^[158]。该算法随机选择一个中间目标节点（图2.3(a)中的 inter. dest 节点），路由包括两个阶段：第一阶段将报文从源节点发送到这个中间目标节点；第二阶段将报文从中间目标节点发送到最终目标节点。

最短路由算法的路径总是位于源节点和目标节点确定的最小象限内。图2.3(b)给出了最短路由算法提供的两条传输路径。最短路由的报文传输跳数

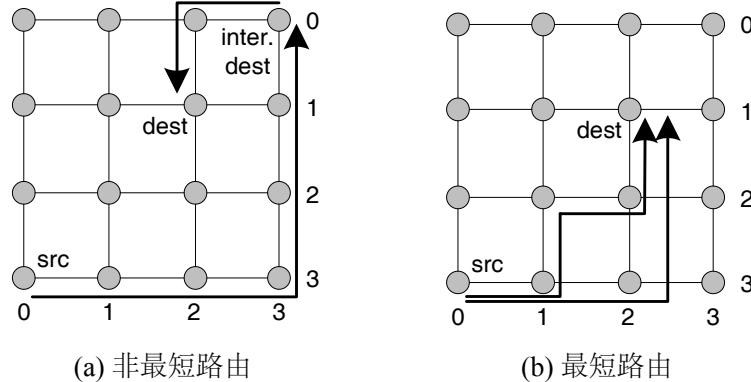


图 2.3 非最短路由和最短路由

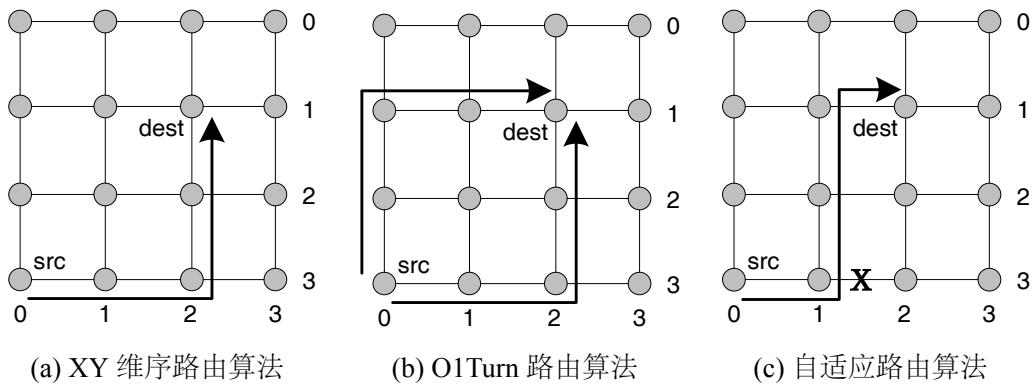


图 2.4 维序、O1Turn 和自适应路由算法

较低，其功耗一般低于非最短路由，因此，片上网络研究大都采用最短路由。但是非最短路由在支持负载均衡^[59] 和链路容错^[66] 等方面具有一定的优势。本文所有路由算法都是最短路由。

根据提供的传输路径条数可以将路由算法分成确定性路由和非确定性路由。确定性路由只为源节点和目标节点提供一条传输路径。维序路由是一种典型的确定性路由算法。如图2.4(a)所示，XY维序路由算法为节点(3,0)和节点(1,2)只提供了一条传输路径，即先沿着X维度传输到目标节点的X位置，然后沿着Y维度传输到目标节点的Y位置。

根据非确定性路由算法是否考虑网络状态可以将其划分成显式路由 (obvious routing) 和自适应路由 (adaptive routing)。显式路由在路径选择时没有考虑网络状态。如图2.4(b)所示，O1Turn 路由算法是一种典型的显式路由算法^[58]。O1Turn 路由算法为源节点和目标节点提供了两条传输路径，即 XY 传输路径和 YX 传输路径^[58]，报文注入时随机选择其中一条。自适应路由根据网络状态自适应地选择传输路径。如图2.4(c)所示，如果算法检测到节点 (3,1) 和节点 (3,2) 之间的链路出现拥塞，则自适应路由算法试图绕开这个拥塞，将报文发送到节点 (2,1) 上，而不

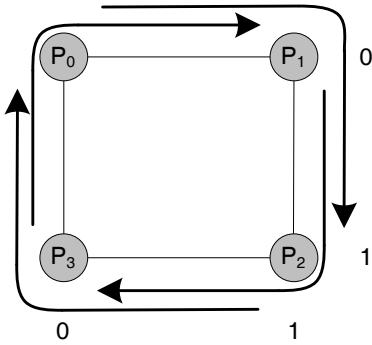


图 2.5 四个报文相互阻塞的网络死锁

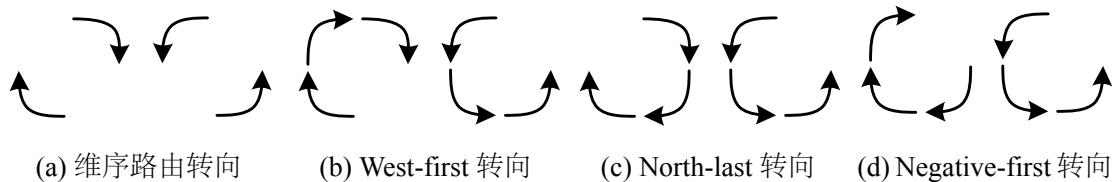


图 2.6 转向模型路由算法

是节点(3,2)上。自适应路由算法根据网络状态选择报文传输路径，能够在一定程度上避免网络拥塞，因此，其一般支持比确定性路由算法更高的网络吞吐率。本文第3章、第4章和第6章的研究都是针对自适应路由算法展开的，第5章的研究是针对 torus 网络的确定性路由算法展开的。

路由算法设计的一个关键问题是保证网络无死锁。图2.5给出了一个典型的由4个报文组成的死锁实例。在该图中，报文 P_0 当前占用了节点(1,0)和节点(0,0)之间的链路，并请求节点(0,0)和节点(0,1)之间的链路。而节点(0,0)和节点(0,1)之间的链路当前被报文 P_1 所占用。类似的情况也发生在报文 P_1 、 P_2 和 P_3 上。没有一个报文可以移动，网络处于死锁状态。

一种消除图2.5中死锁的方法是限制报文只能使用网络中的部分转向，即采用转向模型路由算法 (Turn model routing algorithm) [144, 159, 160]。图2.6给出了4种无死锁的转向模型路由算法。图2.4(a)中的维序路由算法的无死锁特性也可以采用转向模型进行解释，如图2.6(a)所示。XY 维序路由算法禁止报文从 Y 维度转向到 X 维度上，因此，不可能出现图2.5中所示的死锁。

West-first 模型、north-last 模型和 negative-first 模型都在逆时针和顺时针圈中分别禁止了一种转向[159]。它们可以用于设计部分自适应路由算法，即允许报文使用处于源节点和目标节点之间的部分最短路径。例如，west-first 路由要求报文在必要的时候首先向西发送。因此，如果目标节点处于源节点的西侧，则 west-first 路由算法只提供一条最短路径[159]。而如果目标节点处于源节点的东侧，则报文可以使用所有最短路径。除了上述转向模型，还有一些其它转向模型，

odd-even 模型^[160] 在奇数列和偶数列分别限制了不同的转向，abacus 模型^[144] 能动态适应网络负载变化在不同节点设置不同的转向限制。

避免死锁的另外一种方式是在节点之间设置多条物理或虚拟通道，并增加一些使用限制^{[161][162]}。下一节将讨论这方面的设计。

2.1.3 流控机制

本节介绍了两层流控机制，第一层是报文传输层流控，其负责网络缓冲区和链路的分配；第二层是链路层流控，其决定报文传输层流控的物理实现方式。

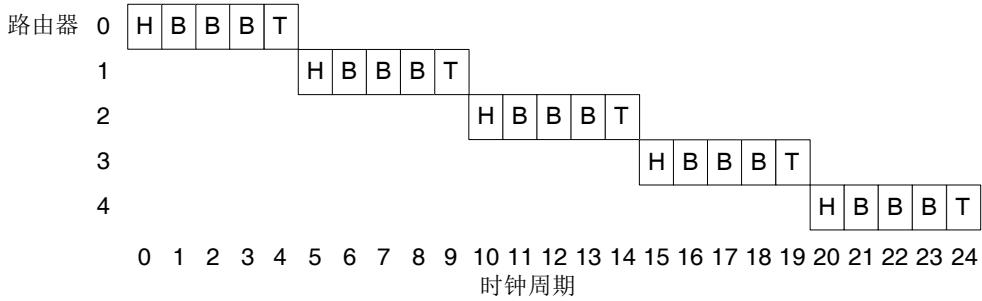
2.1.3.1 报文传输层流控

存储 - 转发^[7, 163]、虚切通^[164] 和虫孔交换^[165] 是三种主要的报文传输层流控机制。存储 - 转发机制只有在接收到完整报文后才能申请下一跳路由器上的缓存和链路。图2.7(a)给出了一个实例。一个包含 5 切片的报文从路由器 0 发送到路由器 4。在每一跳上，只有当接收到所有的 5 个切片后才能进行报文转发，因此，存储 - 转发在每一跳上都引入了串行延迟（serialization latency）^[7, 163]。

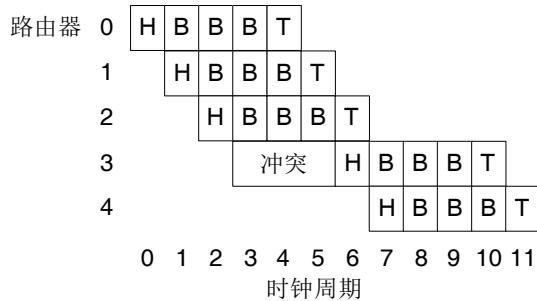
为了减少串行延迟，虚切通机制在接收到报文头切片时即向下游节点转发，如图2.7(b)所示。在路由器 1、路由器 2 和路由器 3 上不存在冲突，因此，每个路由器在接收到报文头切片后即可转发。但是虚切通的缓存分配依然是以报文粒度进行的，也就是说，能够转发的前提是下游路由器有足够的存储该报文的缓存资源。在图2.7(b)中，路由器 3 在时钟周期 3 只有两个可用缓存单元，因此，报文需要在路由器 2 上等待 3 个时钟后才能被转发。虚切通机制要求路由器端口上的缓存资源至少能够容纳一个完整的报文。

虫孔交换机制以切片为单位进行缓存分配。只要下游路由器有一个可用缓存单元，就可以开始发送报文。当不存在网络拥塞时，虫孔交换机制与虚切通机制相同，如图2.7(c)和图2.7(b)路由器 0、路由器 1 和路由器 2 上的情况。但是当网络中出现拥塞时，两者的处理不同。虫孔交换机制只要求下游路由器有一个空闲缓存单元。因此，虽然路由器 3 在时钟周期 3 时只有两个可用缓存单元，路由器 2 依然将该报文的头切片和第一个体切片发送出去，如图2.7(c)所示。虫孔交换的链路分配依然是以报文的粒度进行的。虽然路由器 2 和路由器 3 之间的链路在时钟周期 5 到时钟周期 7 内处于空闲状态，这些链路不能被分配给其它报文使用，直到该报文的尾切片在时钟周期 10 被发送之后。与虚切通机制相比，虫孔交换的最大优点是它没有对缓存数量提出要求。虫孔交换降低了对片上缓存资源的需求，因此，它在片上网络中被大量采用^[166, 167]。

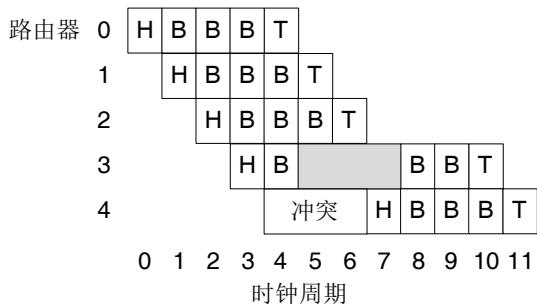
虚通道技术可以与上述三种流控机制结合使用^[7, 163, 168]。一条虚通道其实就是一个独立的先入先出缓存队列。虚通道技术能够在多种场合下被使用，包括提



(a) 存储 - 转发 (H: 头切片, B: 体切片, T: 尾切片)



(b) 虚切通



(c) 虫孔

图 2.7 存储 - 转发、虚切通和虫孔流控机制示意图

高虫孔交换机制的链路利用率、死锁避免和支持 QoS 设计等。这里主要介绍虚通道技术在提高虫孔交换机制的链路利用率和死锁避免方面的作用。

通过配置多个虚通道共享路由器间的物理链路，虚通道允许其他报文跨越被阻塞的报文。路由器以一个切片接着另一个切片的顺序为虚通道仲裁链路带宽。当某个报文的下游虚通道没有可用缓存单元时，别的报文可以通过其它虚通道继续穿越物理链路。因此，虚通道技术提高了链路的利用率。

Dateline 是一种典型的使用虚通道实现死锁避免的设计^[163]。如图2.8所示，dateline 在 ring 网络上配置两条虚通道消除类似于图2.5中的循环等待关系。Dateline 要求注入报文首先使用虚通道 VC_{0i} ，在报文越过网络中的 dateline 之后切换到虚通道 VC_{1i} 上。这种设计消除了虚通道之间的循环等待。虚通道技术还可以

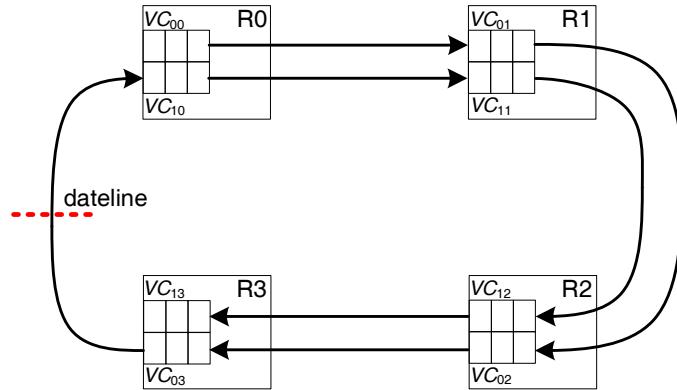


图 2.8 Dateline 设计使用两条虚通道消除循环依赖

表 2.1 各种报文传输层流控机制特性比较

| 流控 | 链路分配粒度 | 缓存分配粒度 | 特性 |
|---------|--------|--------|---|
| 存储 - 转发 | 报文 | 报文 | 只有接收到完整报文后才能申请下一跳 |
| 虚切通 | 报文 | 报文 | 可以在没有接收到完整报文时申请下一跳，但是要求下一跳有足够的存储完整报文的缓存 |
| 虫孔 | 报文 | 切片 | 可以在没有接收到完整报文时申请下一跳，不要求下一跳有足够的存储完整报文的缓存，但是阻塞时不允许其它报文使用物理链路 |
| 虚通道 | 切片 | 切片 | 可以在没有接收到完整报文时申请下一跳，不要求下一跳有足够的存储完整报文的缓存，允许其它报文越过阻塞报文使用物理链路 |

用于避免第2.2.2节中介绍的协议层死锁。

表2.1对上述4种流控机制的链路分配粒度、缓存分配粒度和特性进行了描述。除了这4种流控机制外，还存在其它的流控机制，包括电路交换机制、无缓存流控机制^{[169][45][170]}。同时人们还提出了一些混合流控机制，包括 hybrid switching^[171]、 buffered wormhole^[172] 和 layered switching^[78] 机制。由于虫孔交换机制 + 虚通道技术在片上网络中得到广泛应用，本文的所有设计都是基于这两种流控机制进行的。第4章和第5章提出了两种虫孔交换 + 虚切通的混合流控机制分别用于解决完全自适应路由算法和 torus 网络上死锁避免机制的低效性问题。

2.1.3.2 链路层流控

报文传输层流控机制需要使用链路层流控机制具体实现。链路层流控机制的作用是将下游路由器的可用缓存信息通知给上游路由器，从而避免向一个没有可用缓存的下游路由器发送报文。两种主要的链路层流控机制是基于 on/off 的机制和基于信元（credit）的机制^[7, 81, 163]。

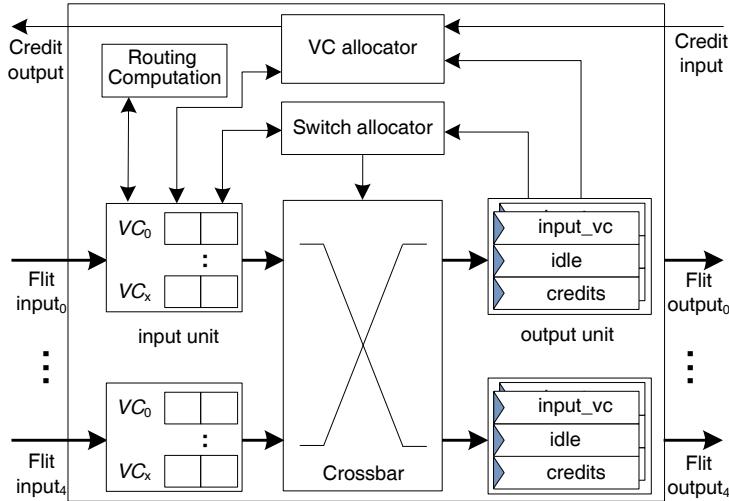


图 2.9 典型片上网络虚通道虫孔路由器结构

On/off 信号在相邻路由器间传递，当下游路由器剩余缓存数目低于某个阀值时，off 信号通知上游路由器停止发送切片。设定的阀值要能保证正在传输的切片到达时下游缓存有足够的存储空间。信元记录了下游路由器可用的缓冲区大小，当有切片离开缓存并释放其存储空间时，就向上游路由器发送一个信元。上游路由器一旦接收到信元就增加信元计数器。信元机制即使在虚通道深度为 1 时依然能够正常运行，但是为了防止缓存使用过程中出现气泡，每条虚通道的缓存数目最好能覆盖信元往返延迟。On/off 机制比信元机制所需要的额外连线资源要低，但是其性能也要低于信元机制^[7, 81, 163]。片上丰富的连线资源使得大部分片上网络都采用基于信元的流控机制，本文假设采用基于信元的链路层流控机制。

2.1.4 路由器微结构和流水线

路由器微结构决定路由器的延迟和功耗。图2.9给出了一个典型的片上网络虚通道虫孔路由器的微结构^[7, 163]。假设网络拓扑为二维 mesh 结构，路由器具有五个输入和输出端口，分别对应于相邻四个方向和本地处理单元。路由器主要由输入单元 (Input units)、路由计算模块 (Routing Computation)、虚通道分配器 (VC allocator)、交换开关分配器 (Switch allocator)、交叉开关 (Crossbar) 和输出单元 (Output units) 构成^[7, 163]。下面介绍各部件的主要作用。

输入单元：输入单元主要包括缓存区和相关链路层流控逻辑。缓存由 SRAM 或寄存器构成，这些存储单元组织成队列形式用于虚通道分配，具有物理环形或逻辑链表结构，动态维护头和尾指针。

路由计算模块：如果当前处于虚通道头部的切片是报文头切片，则输入单元向路由计算模块提出路由计算请求。路由计算模块根据头切片中的报文目标地址

进行路由计算，得到一组输出端口和输出虚通道。不同路由算法返回的输出端口数和输出虚通道数不同。路由计算模块返回的可能是某端口的一条虚通道，或某端口的多条（包括全部）虚通道，或多个端口的多条虚通道。

虚通道分配器：在完成路由计算之后，头切片请求输出虚通道。多个请求通过虚通道分配器进行分配。虚通道分配器收集所有输入虚通道发出的请求信号，并根据这些请求信号将可用的输出虚通道授权给输入虚通道使用。提出了请求但是没有获得授权的头切片将在下一周期继续申请输出虚通道。虚通道分配器由多个仲裁器（arbiter）组成。考虑到面积和功耗的限制，这些仲裁器一般是简单的轮转仲裁器（Round-robin arbiter）^[163] 或矩阵仲裁器（Matrix arbiter）^[163]。只有头切片需要使用路由计算模块和虚通道分配器。

交叉开关分配器：在切片获得某条输出虚通道的使用权后，路由器检查该输出虚通道是否有空闲缓存。如果有可用缓存，则切片向交叉开关分配器请求相应的输出端口。切片在获得输出端口授权后，会将它的虚通道编号更新为输出虚通道号。更新切片虚通道号是为了保证下游路由器将切片存储到相应的输入虚通道中。交叉开关分配器产生控制信号连接交叉开关的输入和输出端口，将切片发送到下游路由器，同时产生信元信号通知上游路由器切片之前占用的缓存单元已被释放。与虚通道分配器类似，交叉开关分配器也是由多个仲裁器组成。

交叉开关（crossbar）：交叉开关一般采用多路选择器实现。多路选择器的控制信号由交叉开关分配器输出。交叉开关在面积和功耗方面的硬件开销为 $O(pw^2)$ 。式中， p 是交叉开关物理端口数， w 是物理端口位宽。

输出单元：输出单元记录下游路由器的虚通道使用情况，其为每条输出虚通道配置了 3 个寄存器：*input_vc*、*idle* 和 *credits*。*Input_vc* 寄存器记录虚通道分配器授权使用该输出虚通道的输入虚通道号；*idle* 记录该下游虚通道是否可以重新分配给新的报文使用，*idle* 值的计算取决于所采用的虚通道分配策略，本文第4章讨论了多种虚通道分配策略；*credits* 寄存器记录下游虚通道的可用缓存数。虚通道分配器和交叉开关分配器访问这三个寄存器以请求合适的输出虚通道。

图2.9中的典型虚通道虫孔路由器结构对应一个 4 级流水线，如图2.10(a)所示^[7, 38, 163]。当头切片到达虚通道头部时，首先进行路由计算（RC），之后申请输出虚通道（VA）；当获得输出虚通道的授权后，申请交叉开关输出端口（SA）；最后执行交叉开关传输（ST）；切片还需要在链路上传输一个时钟周期（LT）。体切片和尾切片不需要执行 RC 和 VA 阶段，它们继承所属报文的头切片获得的输出虚通道使用。

为了降低路由器延迟，人们提出了超前路由计算技术^[60, 63, 173]，其流水线如图2.10(b)所示。超前路由计算在当前路由器上计算下一跳路由器的输出端口，其



图 2.10 基准流水线及其优化

可以与确定性路由算法^[173]或自适应路由算法^[60, 63, 148]一起使用。头切片下一跳路由计算 (Next-hop Routing Computation, NRC) 与当前路由器的虚通道分配阶段同时执行，减少了一个流水线阶段。

猜测技术是另一种流水线优化技术^{[38][39][40]}。图2.10(c)给出了猜测交叉开关分配 (Speculative Switch Allocation, SSA) 的流水线^[38]。头切片在执行下一跳路由计算和虚通道分配的同时申请交叉开关分配。猜测执行假设头切片会获得输出虚通道授权，因此可以并行执行交叉开关分配和虚通道分配。如果这个猜测是错误的，则需要作废刚才的交叉开关分配结果，并在下一时钟周期重新申请交叉开关分配。猜测交叉开关分配技术在较低网络负载时能获得性能提升，因为此时的虚通道分配冲突较低，猜测基本上是正确的。但是，随着网络负载的增加，猜测正确性逐渐下降。更为激进的猜测技术包括猜测交叉开关传输，即将 VA、SA 和 ST 段同时执行^{[39][40]}。这种激进的猜测技术只有在网络负载非常低时才能带来性能提升。

除了上述路由器流水线优化技术外，人们还提出了可变长度流水线结构^[174]、低延迟流水线结构^[43]等。本文第3章、第4章和第6章的路由器流水线结构类似于图2.10(c)。第5章为了与虚切通路由器进行公平比较，其流水线结构类似于图2.10(b)。每章在具体实现时也采用其它一些流水线优化技术，包括超前信号传输^[41, 175]、为整个报文传输保持交叉开关分配结果^[41]等。

2.2 一致性协议对片上网络设计的影响

在 cache 一致性众核结构上，片上网络主要传输一致性协议产生的消息。本节主要论述两方面的问题：1、众核平台上 cache 一致性协议存在的必然性；2、cache 一致性协议的通信特征。

2.2.1 Cache 一致性协议存在的必然性

随着片上集成核数的增加，有些研究者认为 cache 一致性协议的可扩展性受限于通信数量、存储开销、延迟和功耗等方面^[118, 176, 177]。因此，他们开始质疑在众核平台上将无法继续提供硬件支持 cache 一致性协议。例如，Intel 的 48 核 Single-Chip Cluster (SCC) 验证芯片^[118] 没有提供硬件 cache 一致性支持，该芯片核之间采用 MPI 通信模式交换数据。但是 Martin 等最近从五个方面论述了 cache 一致性协议的可扩展性能够满足众核平台的需求^[5]，他们的论述以目录的 cache 一致性协议为例。

1、一致性协议的通信数量

通过平摊分析 (amortized analysis)，Martin 等观察到每个 miss 操作（包括 read 或 write miss）带来的通信与核数无关，即一致性协议通信具有良好的可扩展性。根据具体的一致性协议实现方式，对共享状态 cache 行的 read miss 有可能需要发送给最多一个处理核，因此，read miss 操作产生的通信与处理器核数无关。当出现对共享状态 cache 行的 write miss 时，需要向所有的该 cache 行的共享者发送 invalidation 消息，并且这些共享者需要回复 ACK 报文。在最坏情况下，invalidation 消息需要发送给所有的处理器核。这种场景经常被部分研究者用来说明一致性协议通信的可扩展性较差。但是每个需要发送到 N 个处理器核的 write miss 之前一定有 N 个对该 cache 行的 read miss 操作。因此，如果将 write miss 操作发送的 N 个 invalidation 消息平摊到这个 $N+1$ 的 miss 操作（1 个 write miss 和 N 个 read miss），则 write miss 产生的通信也是与处理核数无关的。也就是说，cache 一致性协议产生的通信数量具有良好的可扩展性。

2、一致性协议的存储开销

一致性协议需要记录 cache 行的共享者。在一个 C 核平台上采用位编码方式进行精确记录，需要使用 C 位。很显然，这种设计的存储开销可扩展性较差。粗粒度记录方式和 cache cluster 层次结构是两种降低这些存储开销的设计。粗粒度记录方式中每位代表多个共享者处理器核。这种设计面临的问题是在某些共享模式 (sharing pattern) 下会导致冗余 invalidation 消息剧增。因此，尽管粗粒度记录

共享者设计的存储可扩展性较好，但是它带来的通信可扩展性较差。

Cluster 层次结构是一种能有效提高 cache 一致性协议可扩展性的设计。由于受到应用程序并行度的限制，多核或众核结构已经具备 cluster 层次性^[178-180]。这些结构在处理核私有 cache 和最后一级共享 cache 之间增加中间 cache 层次，该中间 cache 层次被部分私有 cache，而非所有私有 cache 共享。这些共享同一个中间层次 cache 的多个处理器核组成一个 cluster。采用 cluster 层次设计时，最后一级 cache 上的共享者列表只需记录每个 cluster 上是否存在 cache 行的共享，而中间层次 cache 则精确记录该 cluster 内部的每个核是否为 cache 行的共享者。因此，这种设计能有效降低一致性协议的存储开销，使其具备良好的可扩展性。该设计的另外一个优点是：当 invalidation 消息发送到多个 clusters 时，可以采用 Acknowledgement (ACK) 消息组合使每个 cluster 向最后一级 cache 结构只回复一个消息。本文第6章给出了一种 ACK 消息组合的设计。

3、维持 inclusive cache 结构的开销

cache 层次结构上维持了 inclusive 特性可以简化一致性协议的设计，inclusive cache 结构是指下一级 cache 中的内容一定是上一级 cache 的超集。为了维持 inclusive 特性，每次当最后一级共享 cache 由于冲突需要将某个 cache 行替换时，该 cache 行在各私有 cache 上的 copies 也需要被替换。因此，经常替换最后一级 cache 行会导致性能的降低。Martin 等通过实验模拟的方法证明当共享 cache 的容量大于私有 cache 容量的 4 倍时，共享 cache 的替换概率低于 0.1%。因此，维持 inclusive 结构并不会限制 cache 一致性协议的可扩展性。

4、一致性事务的延迟

如果 load 或 store 指令在私有 cache 上出现 miss，处理核会向共享 cache 发送一个请求信号。在不提供硬件 cache 一致性支持的结构中，如果共享 cache 不能满足这个请求，则其会访问下一层次存储器。而在提供硬件 cache 一致性支持的结构中，共享 cache 有可能将该请求信号转发到另外一个私有 cache 上。因此，硬件支持 cache 一致性结构在这种情况下的延迟有可能高于不提供硬件支持 cache 一致性的结构。但是值得注意的是，即使需要将请求信号转发到另外一个私有 cache，核数增加并不会导致该 miss 事务的延迟无法接受。随着核数的增加，片上网络的规模会增大，这些转发操作的网络延迟可能会上升。类似地，不提供硬件 cache 一致性支持平台上的 miss 事务延迟也会上升。因此，一致性事务的延迟并不是限制 cache 一致性协议可扩展性的一个因素。

5、一致性协议的功耗

影响动态功耗的主要因素是额外的 cache 一致性消息和 cache 查询操作。如之前所分析，核数的增加并不会导致 cache 一致性 miss 操作的消息数增加。同样

的，核数的增加也不会导致 cache 查询操作数急剧上升。影响静态功耗的主要因素是记录 cache 行状态的字段。在采用层次 cluster 结构时，这些记录字段随着核数增加的可扩展性较好，因此，不会导致静态功耗急剧上升。一致性协议需要在处理核上增加一致性状态机的相关逻辑电路。这些电路的漏功耗在每个核上是固定的，因此，也具备良好的可扩展性。

基于上述五点分析，cache 一致性协议的可扩展性并不会限制其在众核平台上的使用。考虑到 cache 一致性编程模式相比于 MPI 编程模式能较大程度减轻编程人员的负担，同时为兼容大量基于 cache 一致性编程模式的历史代码，cache 一致性协议在众核时代将会继续存在下去。因此，对众核时代的 cache 一致性协议提供有效的通信支持是很有必要的。

2.2.2 Cache 一致性协议通信特性

本节分析 cache 一致性协议通信的一些特性，深入理解这些特性有助于设计面向 cache 一致性协议通信优化的片上网络。

2.2.2.1 Cache 一致性事务及协议层死锁

Cache 一致性协议定义了一系列一致性事务^[6]。一致性事务的处理需要传输多个具有依赖关系的消息，这些额外的依赖关系导致原本无死锁的设计出现死锁^[163, 181, 182]。图2.11给出了一个实例。该例假设两种不同的消息：request (req.) 消息和 response (resp.) 消息。在接收到 request 消息后，存储控制器 (memory controller) 会回复一个 response 消息。网络采用无死锁的 XY 维序路由算法。系统在网络接口 (Network Interface, NI) 上设置两个队列分别接收输入的 request 消息和等待输出的 response 消息。由于每次接收到 request 消息时需要回复一个 response 消息，因此，存储控制器只有在 response 队列有空闲空间时才能处理输入的 request 消息。也就是说，存储控制器使得 request 队列依赖于 response 队列。这种依赖关系在网络层使得 Y 维度依赖于 X 维度，而 XY 维序路由算法只允许从 X 维度到 Y 维度的依赖 (图2.6(a))，新增加依赖关系可能导致网络中出现死锁。

两种常用的设计消除这种依赖关系是：1、增加在网络接口上 response 队列的深度，使其能够存储所有可能的 request 消息产生的 response 消息^[181, 182]。为了保证正确性，该设计需要计算所有可能的 request 消息的理论数量上限。在得到这个理论上限之后，还需要在网络接口上设置大量的缓存资源，而片上缓存资源非常宝贵，片上网络很少采用这种设计。2、为存在依赖关系的消息设置独立的物理或虚拟网络^[163, 181, 182]。物理网络是通过配置独立的物理链路和路由器实现，而虚拟网络 (Virtual Network, VN) 则是通过在同一条物理链路上配置多条虚通道实现。每个虚拟网络可能包含一条或多条虚通道，每种消息只能使用属于自己的虚

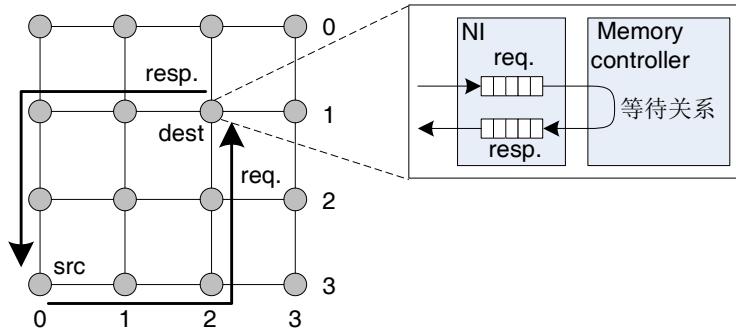


图 2.11 消息依赖关系及其导致的死锁

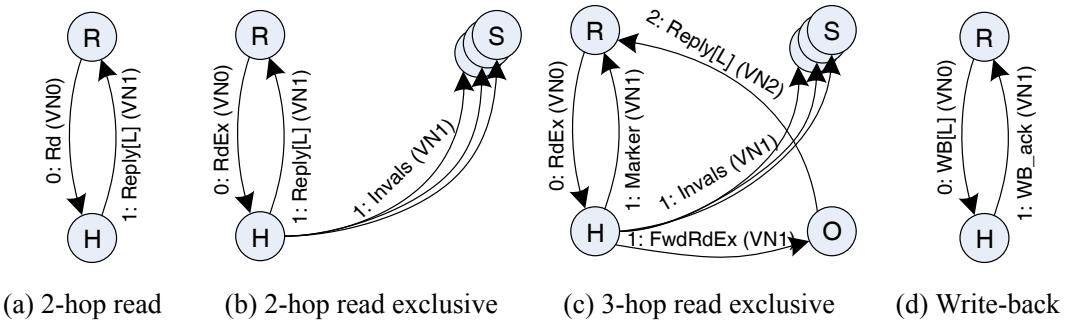


图 2.12 AlphaServer GS320 机器一致性协议的四种基本事务

拟网络的虚通道。这种设计带来的额外开销相对较少，大部分片上网络都是采用这种设计来避免协议层死锁。

Cache 一致性协议要使用多种不同的一致性事务，本节以一个真实产品的 cache 一致性协议描述这些一致性事务。图2.12给出了 AlphaServer GS320 机器^[183]一致性协议的四种基本事务的消息依赖图。图2.12(a)是 2-hop read 事务的消息依赖图。当节点 (图中 R 节点) 的 load 指令出现 cache miss 时，首先通过 VN0 发送一个 read 请求消息 (图中的 *Rd*)。主节点 (图中 H 节点) 接收到 *Rd* 消息后，通过 VN1 回复一个携带了所请求的 cache 行 *Reply* 消息 (图中 ‘[L]’ 表示长消息)。

图2.12(b)和图2.12(c)分别是 2-hop read exclusive 和 3-hop read exclusive 事务的消息依赖图。当 R 节点的 store 指令出现 cache miss 时，首先通过 VN0 发送一个 read exclusive 请求 (图中的 *RdEx*)。主节点接收到 *RdEx* 消息后，如果其有最新的 cache 行内容，则通过 VN1 传输一个携带所请求 cache 行的 *Reply* 消息，同时通过 VN1 向该 cache 行的共享节点 (图中 S 节点) 发送 invalidation 消息 (图中的 *invals*)。由于 AlphaServer GS320 机器采用有序网络 (ordered network) 传输消息，这些共享节点不需要回复 acknowledgement 消息。如果主节点没有该 cache 行的最新内容，则需要将 *RdEx* 消息转发到该 cache 行的拥有者节点 (图2.12(c)中的 O 节点)，*FwdRdEx* 消息通过 VN1 进行传输。当拥有者节点接收到 *FwdRdEx* 消息时，其通过 VN2 回复一个携带 cache 行的 *Reply* 消息。

图2.12(d)给出了 write-back 事务的消息依赖图，当节点的 cache 行需要被替换时执行这个一致性事务。节点首先通过 VN0 传输携带 dirty cache 行的 write-back 消息（图中的 *WB*）。如果主节点没有冲突，则回复 Write-back acknowledgement 消息（图中的 *WB_ack*）以通知请求节点 write-back 事务执行成功。

基于这四种一致性事务，可以观察到 cache 一致性协议的一些通信特性。

1、为了消除协议层死锁，cache 一致性事务需要使用多个虚拟网络。虚拟网络数目取决于最长消息依赖链的长度^[181, 182]。AlphaServer GS320 的最长消息依赖链是图2.12(c)中的 *RdEx* → *FwdRdEx* → *Reply*，其长度是 3，因此，AlphaServer GS320 机器采用了 3 个虚拟网络。

本文第4章中给出了更多其它工业界产品所配置的虚拟网络数目。配置多个虚拟网络会导致每个虚拟网络的虚通道数目受限。本文第4章和第5章的研究面向这个约束分别对完全自适应路由算法和 torus 网络的死锁机制进行优化设计。

2、同一个虚拟网络既需要传输长报文，又需要传输短报文。例如，在 2-hop read exclusive 事务中，VN1 上既传输短 *Intals* 消息，又传输长 *Reply* 消息。VN0 既传输 2-hop read 事务中的短 *Rd* 消息，又传输 write-back 事务中的长 *WB* 消息。

本文第5章观察到当前存在的 torus 网络死锁避免机制无法高效处理这种混合长度报文的传输，因此提出了一种设计来解决这一局限性。

3、Cache 一致性协议中包含多种通信模式。图2.12包含两种通信模式：单播和多播。本文第6章研究了对归约和多播通信的有效硬件支持。下一节详细论述 cache 一致性协议中的通信模式。

2.2.2.2 通信模式

Cache 一致性协议中大部分报文是单播报文，即从一个源节点发送到一个目标节点的报文，如图2.13(a)所示。但是一致性协议也需要使用一些多节点参与的聚合通信，包括多播通信和归约通信。多播通信（图2.13(b)）是指一个源节点将同一个报文发送到多个目标节点上。图2.12(b)和图2.12(c)中的 *Intals* 消息就是一种典型的多播消息。主节点需要将 cache 行作废的 *Intals* 消息发送到多个共享节点上。在采用无序网络（unordered network）的 cache 一致性协议中，每个共享节点在接收到 *Intals* 消息时会回复一个 Acknowledgement (ACK) 消息，这种多对一的通信模式被称为归约通信（图2.13(c)）。

尽管多播和归约通信所占的比例较低 (<5%)，但是它们对网络性能的影响很大。Enright Jerger 等观察到如果不对多播通信提供专门的硬件支持，则 1% 的多播报文会导致网络吞吐率下降 40% 左右^[69]。因此，为了防止这些聚合通信成为系统性能的瓶颈，众核结构需要为它们提供硬件支持。本文第6章研究如何对归约通信和多播通信提供高效的硬件支持。

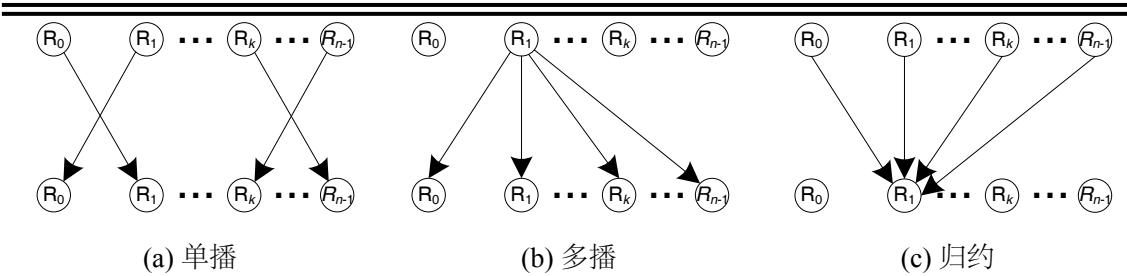


图 2.13 Cache 一致性协议通信模式

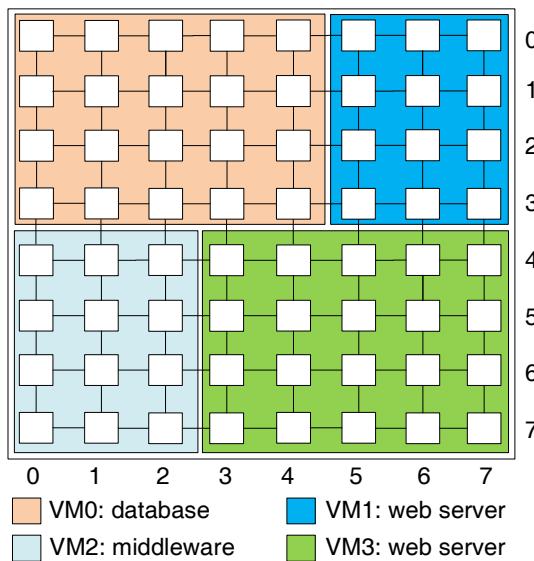


图 2.14 64 核平台负载整合工作模式

2.2.2.3 负载整合模式

为了充分利用众核平台的强大计算能力，一种通用的计算模式是将众核平台划分成多个区域，每个区域配置一个虚拟机（Virtual Machine, VM）运行一个应用程序^[184]，虚拟机的映射区域可能在运行时发生动态变化。如图2.14所示，一个64核平台上同时配置了4个虚拟机运行4个应用程序。Marty和Hill注意到在负载整合模式下，传统的单层目录cache一致性协议中使用cache行的节点和该cache行的目录节点距离可能较远，从而带来大量的全局通信。为了高效支持这种负载整合工作模式，Marty和Hill提出了虚拟一致性层次（Virtual Coherence Hierarchy）的设计^[184]。虚拟一致性层次采用两层目录协议，局部协议满足每个虚拟机内部的cache一致性请求，而全局协议则满足多个虚拟机之间由于进程迁移等引起的cache一致性请求。

设计两层目录结构可以尽量隔离不同的应用程序之间的相互干扰。大部分 cache 一致性请求都能在虚拟机内部的局部协议上的得到满足，只有很少一部分请求需要经过全局协议处理。因此，虚拟一致性层次设计较好地满足了动态隔离

应用程序的需求。在采用这种层次 cache 一致性协议之后，通信大部分是在虚拟机内部完成的，也就是说大部分是局部通信。此时路由算法的设计也需要考虑在多个应用程序之间提供动态隔离性。本文第3章观察到已有路由算法设计不能较好满足负载整合模式的需求，进而提出了一种面向负载整合模式的高效自适应路由算法。

2.3 模拟环境和性能测试方法学

本文大部分实验结果是基于 Booksim 和 FeS2 模拟器获得的。本节介绍这两个模拟器的结构和特性，以及网络性能测试的方法学。

2.3.1 Booksim 模拟器

Booksim 模拟器是 Stanford 大学 Bill Dally 教授领导的 CVA 小组开发的，其最开始是 Dally 和 Bowles 2005 年撰写的书籍《Principle and Practices of Interconnect Networks》的配套模拟器^[163]。CVA 小组成员后续逐渐增加了许多其它功能。Booksim 模拟器由于其模块性、代码易读性和书籍配套性被大量的片上网络研究人员采用。本文实验部分采用了这个模拟器。

图2.15给出了 Booksim 的结构，其主要包括两部分：*Interconnection Network* 模拟网络的拓扑、路由和流控等，*Terminal Instrumentation* 负责网络性能的测试。性能指标是网络吞吐率或报文传输延迟。*Packet source* 模块按照配置的注入率向 *Interconnection Network* 注入报文，报文的目标节点由模拟的流量模式决定。一个无限的 *source queue* 将 *packet source* 和 *Interconnection Network* 隔离开，该模块的作用是隔离报文产生过程和报文传输过程。在 *packet source* 和 *source queue* 之间设置一个 *input count&timing* 模块记录注入的报文数和每个报文的注入时间。将 *input count&timing* 配置在 *source queue* 之前是为了将报文在 *source queue* 中消耗的时间也记录在报文延迟中。*output count&timing* 模块记录每个节点接收的报文数目以及每个报文的结束时间。网络吞吐率通过统计到达每个节点的报文数目表示，而报文传输延迟则通过将每个报文的结束时间减去其注入时间得到。

Booksim 支持两种模拟方式：open-loop 和 closed-loop 模拟。Open-loop 模拟方式中网络状态不会影响报文的注入过程。图2.15中的 *source queue* 能够消除网络状态对报文注入的影响。如果没有这个 *source queue*，可能会出现如下情况：当某个节点按照配置注入率注入报文时，*Interconnection Network* 由于没有可用缓存资源而无法接收该报文，此时会改变报文的注入过程。本文后续章节对合成流量模式的实验评估都是采用 open-loop 模拟方式的。Closed-loop 模拟允许网络状态影响报文的注入过程，其主要用于模拟全系统的性能。本文使用了一种全系统模

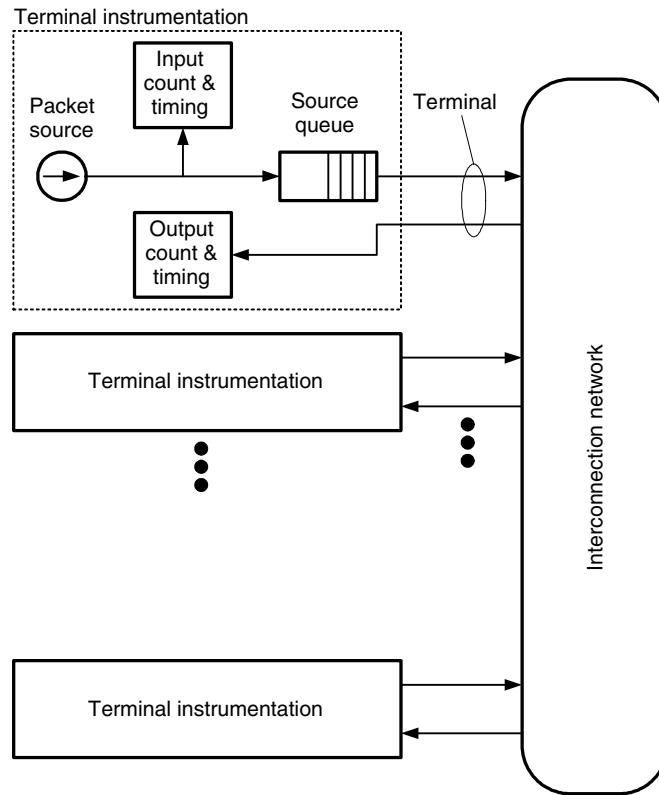


图 2.15 Booksim 模拟器结构框图

拟器评估全系统性能，因此没有使用 closed-loop 模拟方式。

网络的性能是指模拟器稳定状态下的性能。为了得到稳定状态时的性能，Booksim 的模拟过程包括三个阶段：warm-up、measurement 和 drain。首先，网络需要运行 N_1 个 warm-up 时钟周期以使网络逐步进入稳定状态。在 warm-up 阶段注入的报文数不会被统计，同时每个报文的注入时间也不会被记录。当 warm-up 时间结束后，模拟器运行 N_2 个 measurement 时钟周期，此时模拟器已进入稳定状态。在 measurement 阶段注入的报文数被统计，同时每个报文的注入时间也被记录。这些报文被称为 measurement 报文。最后在 drain 阶段，网络运行足够长的时间直到所有的 measurement 报文都达到了目标节点。在 drain 阶段只统计 measurement 报文的到达时间。网络吞吐率是通过统计每个目标节点在 measurement 阶段接收的报文数得到。而报文平均延迟则通过记录所有 measurement 报文的传输延迟得到。值得注意的是，drain 阶段是必不可少的，因为该阶段保证了所有 measurement 报文的传输延迟都能够被统计到，否则会造成实验结果的失真。尽管 warm-up 和 drain 阶段注入的报文不会被统计性能，它们与 measurement 报文在网络中一起传输，因此会影响实验结果。

表 2.2 合成流量模式定义

| 分类 | 模式名 | 定义 |
|----------|----------------|---|
| 非置换 | Uniform random | $\lambda_{sd} = 1/N$ |
| Bit 置换 | Bit complement | $d_i = \neg s_i$ |
| | Bit reverse | $d_i = s_{b-i-1}$ |
| | Bit rotation | $d_i = s_{(i+1) \bmod b}$ |
| | Shuffle | $d_i = s_{(i-1) \bmod b}$ |
| | Transpose | $d_i = s_{(i+b/2) \bmod b}$ |
| Digit 置换 | Tornado | $d_x = (s_x + (\lceil k/2 \rceil)) \bmod k$ |
| | Neighbor | $d_x = (s_x + 1) \bmod k$ |

2.3.2 网络性能测试方法

网络性能测试一般采用两种方法：1、合成流量模式测试。2、真实应用程序测试^[163]。合成流量模式测试的优点是实验速度快，方便通过多种不同流量模式测试网络设计的各个方面。但是合成流量模式的实验结果与真实系统情况有一定的差距。真实应用程序测试的优点是实验结果直接反应了设计对系统性能的影响，但是其缺点是实验消耗时间长，而且全系统模拟参数众多，任何一个参数的改变都有可能会影响最终实验结果。其它的网络性能测试方法包括使用真实应用程序的 trace 文件等。本文实验评测既采用了合成流量模式，又采用了真实应用程序。本文在进行实验配置参数敏感性分析时，一般采用合成流量模式测试。

合成流量模式一般采用置换矩阵定义^[163]。在置换矩阵定义的流量模式中，每个源节点只向一个目标节点发送报文。表2.2给出了本文使用的合成流量模式的定义^[163]。除了 Uniform random 模式外，其它流量模式都是置换流量模式。Uniform random 模式以相同的概率向网络中任意节点发送报文。

合成流量模式测试的性能指标主要有两个：零负载延迟和网络饱和吞吐率^[7, 163]。如图2.16所示，零负载延迟是指一个报文在没有其它报文冲突时的平均网络传输延迟。网络饱和吞吐率是指网络达到饱和时的注入速率。网络达到饱和的一个现象是报文平均延迟急剧上升。因此，一般根据平均延迟定义网络是否达到饱和。本文定义当网络平均延迟是零负载延迟 3 倍时，网络达到饱和。本文全系统模拟测试结果使用程序运行时间表示。

2.3.3 FeS2 模拟器及其与 Booksim 模拟器的整合

FeS2 模拟器是面向 x86 体系结构的全系统时钟精确模拟器^[185]，其主要由三部分组成：Virtutech Simics 实现底层的 x86 体系结构功能模拟^[186]，PTLsim 实现

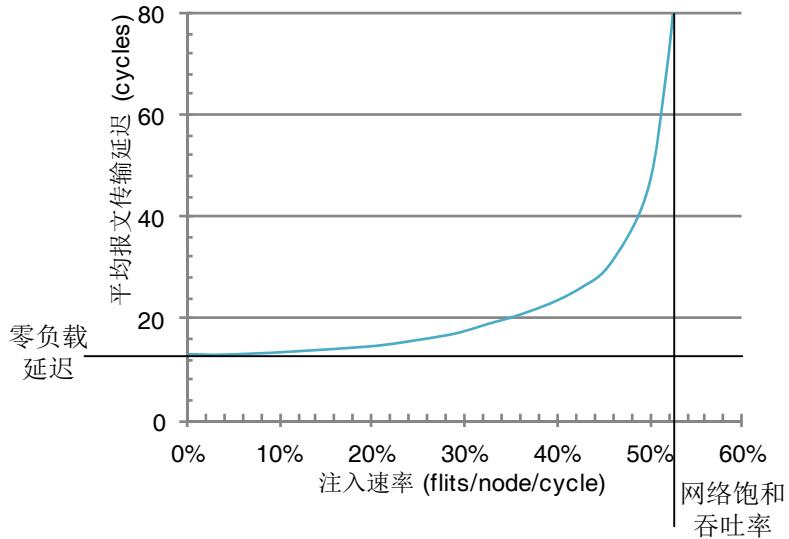


图 2.16 片上网络注入率 - 延迟曲线图

时序模拟^[187]，Ruby 实现存储系统模拟^[188]。但是 FeS2 所使用的 Ruby 存储模型提供的功能有限，并且没有很详细地模拟报文在片上网络中的传输。在 FeS2 模拟器中只包含一个简单的全相连交叉开关网络。

为了解决这一局限性，我们将 Booksim 模拟器和 FeS2 模拟器进行了集成。具体而言，就是将 FeS2 模拟器中已有的简单全相连交叉开关网络模块替换成 Booksim 模拟器。报文由 FeS2 模拟器产生，其通过网络接口注入到 Booksim 模拟器中。Booksim 模拟器按照所配置的路由器微结构、路由算法和流控机制完成报文的传输。当报文到达目标节点后，其会被排出给 FeS2 模拟器。通过这种方式我们可以调整 Booksim 模拟器中的网络参数，测试不同路由算法、不同流控机制下的全系统整体性能。

第三章 面向负载整合工作模式的路由算法

随着片上集成核数的增加，多个应用程序会同时运行在一个众核平台上，这种负载整合工作模式对片上网络路由算法设计提出了新的挑战。当前存在的局部自适应路由算法和全局自适应路由算法不能很好地满足负载整合模式对性能的需求。局部自适应路由算法的短视性限制了其有效避免网络拥塞的能力。为了解决这个缺点，全局自适应路由算法使用一个拥塞信息传播网络获得超越邻居节点的网络状态，它能更为有效地避免网络中的拥塞。但是，在负载整合模式下，全局自适应算法的输出端口选择存在应用程序内部和应用程序之间的干扰，这些干扰耦合了多个同时运行的应用程序，降低了它们的性能。

为了解决已有路由算法在负载整合工作模式下的局限性，本章提出了基于目标的自适应路由（Destination-Based Adaptive Routing, DBAR）。DBAR 采用一个低开销的拥塞信息传播网络获得本地和全局的状态信息，从而使其能有效地避免邻居节点之外的网络拥塞。更重要的，通过将目标信息集成到输出端口选择中，DBAR 消除了应用程序内部和应用程序之间的干扰，从而为多个同时运行的应用程序提供了动态隔离性。与之前的设计相比，DBAR 在许多配置下都获得了更好的性能，同时它能够在中等或较高网络负载情况下降低能量延迟积。

3.1 引言

受限于应用程序的并行度，一个众核计算平台^[189-191] 上很有可能同时运行多个应用程序，这种工作模式被称为负载整合（workload consolidation）^[184]。在负载整合工作模式下，支持多个应用程序之间的隔离和有效共享片上资源是影响系统整体性能的一个关键问题。人们已经对一些片上资源（如 cache^[192] 和存储控制器^[193]）的共享和隔离开展了大量的研究，并且提出了许多有效的机制，但是当前对片上网络^[1] 这种资源的共享和隔离的研究较少。一个应用程序在片上网络中的通信会影响另一个程序的性能，本章从路由算法设计的角度出发，试图在支持多个应用程序高效共享片上网络的前提下，维持应用程序之间的隔离性。

负载整合工作模式对路由算法提出了不同的需求。首先，路由算法应该能提供足够的适应性以有效避免网络中的拥塞；其次，它不应该利用冗余的网络状态信息，因为这些信息会导致路由算法对网络状态做出不精确的评估；更重要的，路由算法应该能动态隔离多个同时运行的应用程序。当前存在的路由算法不能满足这些要求。类似于维序路由的确定性路由算法忽略了网络状态，它们不能避免网络中的拥塞。为了避免网络中的拥塞，自适应路由算法在源节点和目标节点之

间提供了多条传输路径，路由算法使用一种端口选择策略在这些候选路径中选择一条进行报文传输。

路由算法应该选择具有最低拥塞程度的路径。局部自适应路由算法只考虑本地网络状态，其做出的决策有可能影响网络的全局均衡性^[63]。路径上邻居(Neighbors-on-path, NoP)路由算法考查邻居节点的相邻节点的状态^[62]，这种路由算法完全忽视了邻居节点的状态，从而限制了它的性能。类似于区域拥塞感知(Regional Congestion Awareness, RCA)^[63]的全局自适应路由算法使用一个专门的拥塞信息传播网络来获得本地和远端的网络状态，然后根据这些状态信息进行输出端口选择。但是RCA算法使用的网络状态中包含冗余信息，它不能为应用程序提供动态隔离性，因此不适合于负载整合工作模式。RCA算法中的冗余信息可以划分为应用程序内部干扰和程序之间的干扰，这些冗余信息耦合了多个同时运行的应用程序的行为，降低了单个应用程序性能的可预测性。

在众核平台上将会大量出现负载整合工作场景，因此，一种高效的路由算法应该既能提供良好的适应性，又能动态维持应用程序之间的隔离性。为了实现这个目标，路由算法应该使用适当的网络状态信息，不足或者冗余的信息都会导致性能的下降。本章提出的DBAR路由算法通过一个低开销的拥塞信息传播网络获得了本地和远端的状态信息，从而提供了良好的适应性；同时，DBAR通过将报文目标位置集成到输出端口选择中消除了冗余网络状态信息。DBAR只考虑报文可能经过的节点的状态，而忽略位于当前位置和目标位置定义的最小象限外的网络状态，因此，它为多应用程序同时运行提供了动态隔离性。实验结果表明，DBAR在很多网络配置下优于其它路由算法。

3.2 相关研究

本节主要讨论应用程序映射和自适应路由算法设计方面的相关工作。

离线应用程序映射采用了分支定界算法^[89]和两步基因算法^[92]；由于不能在设计时获得程序的到达和执行时间，因此需要采用运行时应用程序映射技术^[89–92]；在负载整合模式下，将每个应用程序映射到一个凸区域能够获得最优的性能^[90, 91]。大部分应用程序映射算法只考虑了源节点和目标节点之间的曼哈顿距离^[90, 92]，而没有考虑报文的具体传输路径，因此，路由算法设计与这些应用程序映射研究工作是互补的。

如图3.1所示，自适应路由算法由两部分组成：路由函数和选择策略^[62]。路由函数根据报文的当前位置和目标位置计算出所有可能的候选输出通道；选择策略根据网络状态，从这些候选输出通道中选择一条进行报文传输。之前的研究工作也提出了一些与网络状态无关的选择策略，包括Z字前进、XY、非转向和最大

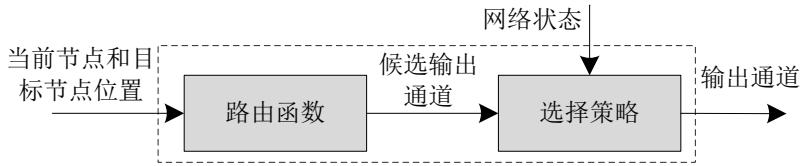


图 3.1 自适应路由算法的结构

灵活度选择策略等^[159, 194–197]。

路由函数必须保证无死锁^[159, 160, 162, 198, 199]；死锁避免理论一般需要将一条物理通道划分成多条虚通道（Virtual Channel, VC）^[162, 198, 199]。Dally 和 Seitz 证明了确定性路由算法无死锁的充分必要条件是通道依赖图中无圈^[198]；Duato 进一步给出了完全自适应路由算法无死锁的一系列理论^[162, 199, 200]，这些理论被广泛应用于路由算法的设计中。完全自适应路由算法允许报文使用源节点和目标节点之间的所有路径，本章所提出的 DBAR 算法是一种完全自适应路由算法，其死锁避免是基于 Duato 理论设计的。除了确定性路由算法和完全自适应路由算法，还存在另外一种类型的路由算法，即部分自适应路由算法。转向模型是典型的部分自适应路由算法^[159, 160]，它们只允许报文使用源节点和目标节点之间的部分路径。

片外网络的带宽受限于芯片的引脚数，而片上网络没有这些限制，因此，它们连线资源更为丰富，丰富的片上连线资源可以用于实现拥塞信息传播网络。这些因素使得片上网络路由算法设计逐渐成为一个研究热点。DyAD 路由综合了自适应路由和确定性路由的优点^[57]；DyXY 在邻居节点间使用专门连线来传输状态信息以做出更为有效的输出端口选择^[61]；Kim 等提出了一种采用超前路由计算和预选择策略的低延迟自适应路由算法设计^[60]。这三个研究的输出端口选择都是基于邻居节点的状态进行的^[57, 60, 61]，NoP 路由算法则考查邻居节点的相邻节点，即两跳之外的节点上的状态^[62]。

RCA 路由算法^[63]是第一种既使用了本地状态信息，又使用了远端状态信息的设计。然而，RCA 使用的状态信息中包含冗余信息，这些冗余信息在负载整合模式下更为明显。冗余信息降低了输出端口拥塞状态评估的准确性，为了解决这个问题，Ramanujam 和 Lin 提出将报文目标信息融入到输出端口的选择过程中^[64]。他们的设计为每个目标节点维持一个延迟估计，并且使用这些估计完成输出端口的选择。尽管本章研究的出发点与他们类似，但是 DBAR 与他们的设计有很大不同，同时，本章主要着眼于提升路由算法在负载整合模式下的性能。

就我们所知，迄今为止，只有一篇文章考虑了负载整合模式下的路由算法设计^[65]；bLBDR 在相邻区域之间静态配置一些连接位来隔离应用程序^[65]，它的死锁避免是基于转向模型设计的，同时其输出端口选择是基于邻居节点的状态进行的。与之不同，DBAR 路由算法采用一种基于目标的输出端口选择策略，该策

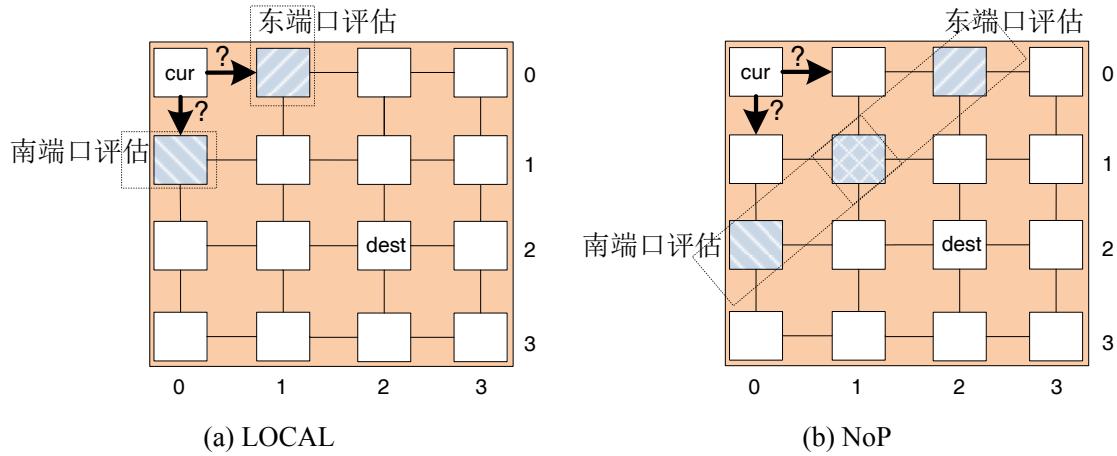


图 3.2 LOCAL 和 NoP 报文路由实例

略能够在多个同时运行的应用程序间提供动态隔离性。同时，DBAR 的路由函数是基于 Duato 理论设计的，其允许报文使用源节点和目标节点之间的所有路径。DBAR 最主要的特性是其输出端口选择策略既提供了良好的适应性，也能动态隔离多个同时运行的应用程序。

3.3 研究动机

本节从两方面说明设计一种高效的路由算法的必要性。首先，为了有效避免网络拥塞，路由算法必须使用足够的网络状态信息；局部自适应（LOCAL）路由算法和 NoP 路由算法^[62] 使用的网络状态信息不足，限制了它们的性能。第二，虽然 RCA 路由算法^[63] 引入了一个轻量级的边带监视网络获得全局网络状态信息，它的性能依然受限于应用程序内部和应用程序之间的干扰。本章所提出的 DBAR 路由算法在这两种设计之间进行了折中。

3.3.1 局部自适应算法的局限性

局部自适应（LOCAL）路由算法根据邻居节点的状态完成输出端口的选择，这些状态信息可以是空闲缓存数^[57, 60-63]、空闲虚通道数^[63, 197]、交叉开关请求数^[63]、或者它们的组合^[63]。图3.2给出了一个报文路由的实例，报文的当前位置（cur）是路由器（0,0），目标节点位置（dest）是路由器（2,2）。此时报文通过南输出端口和东输出端口都可以到达目标节点，因此，需要使用一种策略从它们中间选择一个进行传输。LOCAL 路由算法（图3.2(a)）通过考察当前位置的两个邻居节点的状态做出选择，它只考虑路由器（0,1）和路由器（1,0）的状态，而没有考虑邻居节点之外的节点的状态，因此，LOCAL 不能避免 1 跳以外的网络拥塞。

如图3.2(b)所示，NoP 路由算法^[62] 考察邻居节点的相邻节点的状态，NoP 存

在的问题是：它忽略了邻居节点的状态（路由器(0,1)和路由器(1,0)），其端口选择完全是基于距离当前位置两跳的节点的状态做出的。在这个实例中，当评估东输出端口时，NoP 考察了路由器(0,2)和路由器(1,1)的状态；当评估南输出端口时，NoP 考察了路由器(2,0)和路由器(1,1)的状态。由于奇偶路由函数限制了一些转向以实现死锁避免，NoP 策略在该路由函数下获得了较好的性能^[62]。然而，在完全自适应路由函数下，NoP 策略因为网络状态信息不足而导致性能下降。

3.3.2 应用程序内部的干扰

Gratz 等提出了三种 RCA 变体：RCA-1D、RCA-Fanin 和 RCA-Quadrant^[63]。RCA-1D 沿着每个维度传输拥塞信息，这些拥塞信息在传输过程中进行聚集操作；RCA-Fanin 聚集了正交维度的信息，从而获得更多的网络状态；RCA-Quadrant 基于象限划分网络状态信息，其获得了比 RCA-Fanin 更为精确的信息；没有任何一种 RCA 变体在所有流量模式下都获得了最佳性能。实验结果显示，RCA-Quadrant 和 RCA-Fanin 在负载整合模式下比 RCA-1D 引入了更多的干扰，因此，本章研究主要与 RCA-1D 进行比较。

图3.3给出了RCA-1D单区域内报文路由的实例。在该例中，一个应用程序运行在这16个节点上。当评估东输出端口时，RCA-1D考虑了三个节点的状态：路由器(0,1)、路由器(0,2)和路由器(0,3)。类似的，当评估南输出端口时，RCA-1D考虑了路由器(1,0)、路由器(2,0)和路由器(3,0)的状态。报文的目标节点是路由器(2,2)，路由器(0,3)和路由器(3,0)位于由当前位置和目标位置确定的最小象限之外，报文传输不会经过这两个节点，因此，来自路由器(0,3)和路由器(3,0)的状态属于冗余信息。本章将这些冗余信息称为应用程序内部干扰，它们导致路由算法选择次优的输出端口，从而带来性能的下降。RCA-1D在评估输出端口时，考虑了同一个方向上的所有路由器的状态，这会导致在通信具有较大局部性时不精确地评估网络的拥塞状态。

为评估合成流量模式^[163]的通信局部性，图3.4在4×4和8×8 mesh网络上测试了它们的平均跳数；大部分流量模式在4×4和8×8 mesh网络上平均跳数低于3跳（平均2.63跳）和5.6跳（平均5.58跳）。这些流量模式的源节点距离目标节点一般较近，它们具有较大的通信局部性。因此，为获得更高性能，路由算法应该消除应用程序内部的干扰。

3.3.3 应用程序之间的干扰

图3.5给出了一个8×8 mesh网络上的负载整合模式实例，随着核数的增加，类似的场景将大量出现在众核平台上。这个实例中有4个应用程序同时运行，每

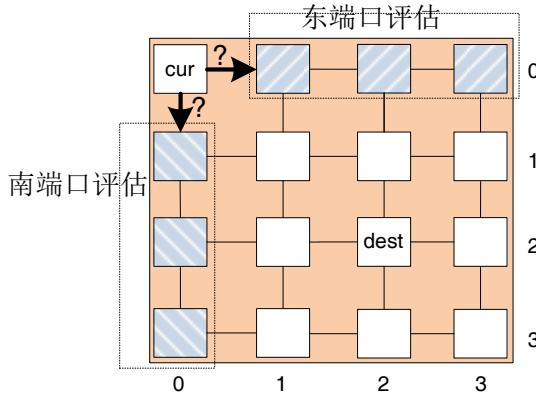


图 3.3 RCA-1D 报文路由实例

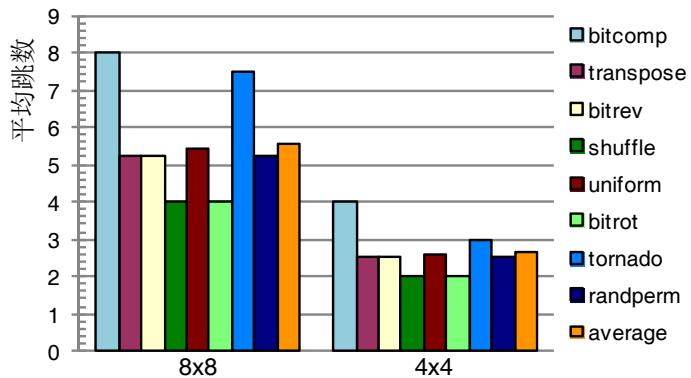


图 3.4 合成流量模式的平均跳数

个应用程序映射到一个 4×4 的 mesh 区域。区域 R0 由节点 (0,0) 和节点 (3,3) 定义，区域 R1 由节点 (0,4) 和节点 (3,7) 定义，区域 R2 由节点 (4,0) 和节点 (7,3) 定义，区域 R3 由节点 (4,4) 和节点 (7,7) 定义。图3.5中给出了一个当前处于路由器 (0,2) 上的报文，其目标节点是路由器 (2,0)。尽管区域 R0 的通信与其它区域是不相关的，但是 RCA-1D 在为区域 R0 内的报文选择输出端口时，考虑了区域 R2 节点的状态。显然，RCA-1D 的输出端口选择中存在大量的干扰，这些干扰既降低了应用程序的性能，也影响了它们之间的隔离性。

为评估应用程序之间干扰的影响，我们为区域 R0 配置了 transpose 流量模式，为区域 R1-R3 配置了 uniform random 流量模式，图3.6给出了区域 R0 的性能。*RCA-uni_region* 曲线是 RCA-1D 在 4×4 mesh 网络上的性能，此时没有任何应用程序之间的干扰，即完全隔离条件下 RCA-1D 的饱和吞吐率约为 65%。然而，当多个应用程序运行在 8×8 mesh 网络上时，RCA-1D 的饱和吞吐率出现了下降。*RCA-multi_regions(4%)* 曲线是当区域 R1-R3 的注入率为 4% 时，区域 R0 的性能；RCA-1D 的饱和吞吐率约为 50%。当区域 R1 的注入率是 64%，区域 R2 和 R3 的注入率保持 4% 时，RCA-1D 的性能进一步下降，此时其饱和吞吐率约为 47%（曲线 *RCA-multi_regions(64%)*）。显然，区域 R1-R3 的状态信息影响了区域

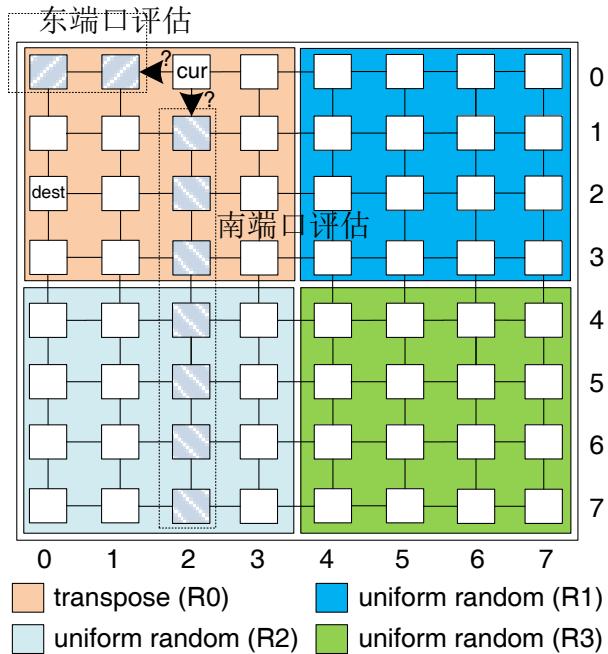


图 3.5 负载整合场景报文路由实例

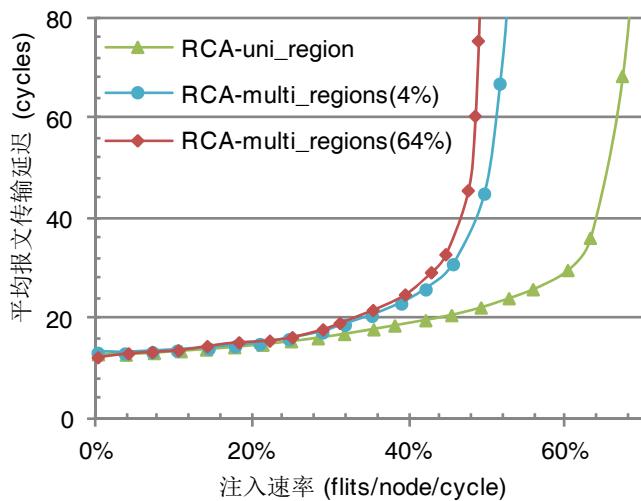


图 3.6 R0 区域性能曲线图

R0 的路由决策。也就是说，RCA-1D 耦合了本来应该相互独立的应用程序。

可以在区域边界上静态配置隔离位扩展 RCA-1D 设计，从而消除区域之间的干扰。这种机制比较复杂，同时边界隔离位必须离线计算，但是应用程序及其映射的区域可能在运行时动态变化，这种机制缺乏足够的灵活性处理这种情况。此外，该机制无法消除应用程序内部干扰，而这种干扰在小规模区域中影响很大。

本章提出的 DBAR 算法只考虑报文当前位置和目标位置所定义的最小象限内的节点的状态，从而消除了应用程序内部和应用程序之间的干扰。图3.7给出了 DBAR 算法的一个报文传输实例，当评估东输出端口时，DBAR 考虑路由器

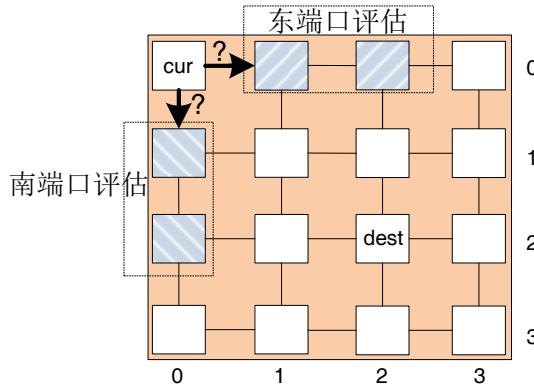


图 3.7 DBAR 报文路由实例

(0,1) 和路由器 (0,2) 的状态；当评估南输出端口时，DBAR 考虑路由器 (1,0) 和路由器 (2,0) 的状态。DBAR 同时考虑了本地节点和非本地节点的状态，因此它比 LOCAL 和 NoP 能提供更好的适应性。另一方面，DBAR 没有考虑路由器 (0,3) 和 (3,0) 的状态，从而消除了干扰。更重要的，如果负载整合模式的每个应用程序被映射到一个凸区域^[90, 91]，DBAR 为不同区域内的应用程序提供了动态隔离性。

3.4 面向负载整合的自适应路由算法

自适应路由算法的端口选择策略对性能至关重要^[62, 194–196]。一种面向负载整合模式的高效选择策略应该满足如下两点要求：良好的适应性和动态隔离性。选择策略应该能够感知本地节点和远端节点的拥塞，以便对网络状态做出精确的评估。同时，它不应该引入冗余信息。最重要的是，在负载整合模式下，选择策略应该能够为应用程序提供动态隔离性，进而消除应用程序之间的干扰。

本节首先介绍 DBAR 所采用的低开销拥塞信息传播网络，因为传输空闲虚通道数目所需开销较低，DBAR 使用它作为反映节点拥塞状态的指标，其它指标获得的性能类似。路由器向同一维度的其它路由器发送自己的空闲虚通道数目，这样每个路由器都能感知到同一维度其它路由器的状态。

3.4.1 拥塞信息传播网络

片上网络拥有大量的连线资源，这些连线资源允许设计一个专门的拥塞信息传播网络。通过该网络，路由器能够感知本地和非本地的拥塞，从而对网络状态做出精确的评估。NoP 和 RCA 路由算法都使用了这样一个专门的拥塞信息传播网络^[62, 63]，NoP 通过此网络在节点之间交换自己邻居节点的状态，RCA 通过此网络获得全局网络状态。DBAR 的设计同样希望获得全局网络状态，因此，本节主要与 RCA 的拥塞信息传播网络进行比较。

在 RCA 拥塞信息传播网络的每一跳上，本节点的状态信息与邻居节点发送

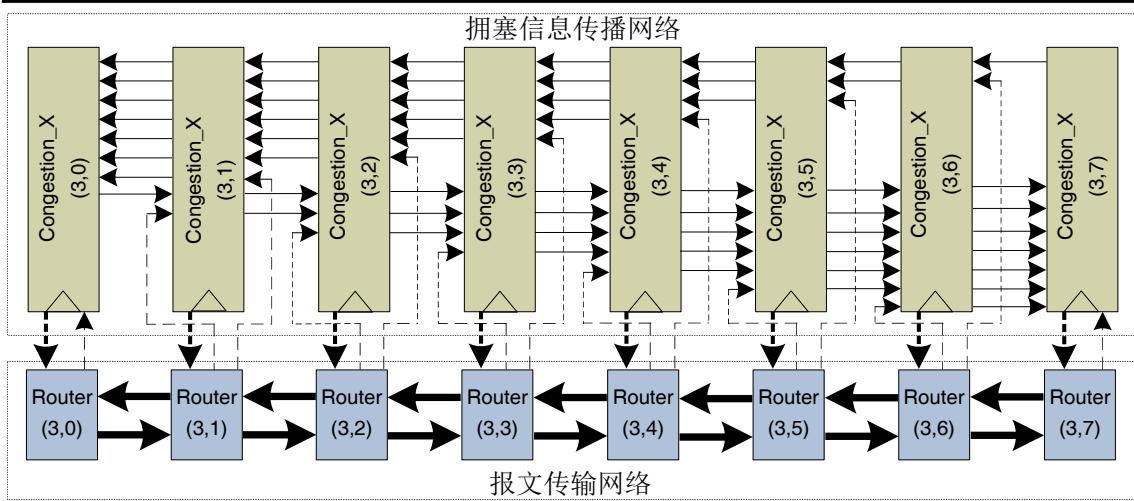


图 3.8 拥塞信息传播网络结构

来的信息进行聚合，然后向上游节点转发^[63]。这种设计存在两方面的局限性：第一，聚合操作将本节点的状态信息和其它节点的状态信息混合在一起，导致选择策略无法过滤冗余信息；第二，聚合操作在拥塞信息传播的每一跳中引入了一个额外的时钟延迟，导致 RCA 不能及时获得远端状态信息。针对这些局限性，本节提出的拥塞信息传播网络的每一跳只消耗一个时钟周期，从而使 DBAR 能获得更为及时的状态信息。同时，该拥塞信息网络方便基于目标位置过滤冗余信息。

图3.8给出了8×8 mesh 网络第3行上的DBAR 拥塞信息传播结构，其它的行或列上的结构与之类似。路由器为每个维度设置了1个专用寄存器（congestion_X 或 congestion_Y）存储该维度的拥塞信息，输入的拥塞信息和本节点的状态信息在下个时钟通过拥塞信息传播链路发送到邻居节点上。为覆盖所有可能的空闲虚通道数目，每条拥塞信息传播链路需使用 $\log(\text{numVCs})$ 位。但是两方面的原因使得采用粗粒度评估方式就足够了：一方面，精确计数邻居节点的空闲虚通道数对性能的影响很低。比如说，如果两个候选端口分别有5条和6条空闲虚通道，则这两个端口都处于低负载状态，因此可以使用任何端口；另一方面，拥塞信息传播网络根据距离加权节点拥塞值，距离越远，权重越低，此时精确计数远端节点的状态也没有实际意义。实验表明，采用1位拥塞信息传播链路就可以获得较高的性能。路由器以 on/off 方式向邻居节点发送自己的状态，当物理链路配置8条虚通道时，信号“0”（拥塞）表示空闲虚通道数低于4条，信号“1”（非拥塞）表示空闲虚通道数大于或等于4条。这种粗粒度方式降低了拥塞信息传播链路的翻转次数，因此有利于降低功耗。

在8×8 mesh 网络上，使用粗粒度拥塞信息传播机制，congestion_X 和 congestion_Y 的位宽是9位，其中2位存储本路由器的两个输出端口的状态，另外7位存储同一维度的其它路由器的状态信息。拥塞信息传播网络基于距离加

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| E(3,2) | E(3,1) | E(3,0) | W(3,2) | W(3,3) | W(3,4) | W(3,5) | W(3,6) | W(3,7) |

(a) *Congestion_X*

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| S(3,2) | S(2,2) | S(1,2) | S(0,2) | N(3,2) | N(4,2) | N(5,2) | N(6,2) | N(7,2) |

(b) *Congestion_Y*

图 3.9 路由器 (3,2) 的拥塞信息存储寄存器格式

权拥塞信息，距离每多一跳，权重减半，这个加权比例是基于已有工作^[63]和硬件实现难度选定的。寄存器相邻位之间隐式维持 0.5 的权重比例，因此，只需将信息存储到寄存器的相应位就可以实现上述加权机制。

图3.9给出了路由器 (3,2) 的 *congestion_X* 和 *congestion_Y* 寄存器的格式。*Congestion_X* 第 0 位存储本地路由器的东 (E) 输入端口的状态，第 1 位和第 2 位分别存储距离路由器 (3,2) 一跳和两跳之外的西 (W) 边路由器的状态，即路由器 (3,1) 和 (3,0) 的状态。第 0 到 2 位被发送给路由器 (3,2) 的东邻居节点：路由器 (3,3)。*Congestion_X* 第 3 位存储本地路由器的西输入端口的状态，接下来的 5 位按照距离依次存储东边的各路由器的西输入端口状态。第 3 到第 8 位被发送到路由器 (3,2) 的西邻居节点：路由器 (3,1) 上。*Congestion_Y* 寄存器结构类似，其用于存储路由器的南 (S) 端口和北 (N) 端口的状态。

RCA-Fanin 和 RCA-Quadrant 在评估输出端口时使用了来自不同维度的状态信息，根据 Gratz 等的实验结果，来自其它维度的状态信息有时能提高性能^[63]。DBAR 在评估输出端口时只使用该端口所属维度的状态信息，它也可以被扩展以使用来自不同维度的状态信息。然而，这会增加上述拥塞信息传播网络的复杂性，因此，本章没有考虑这种设计。

3.4.2 DBAR 路由器微结构

DBAR 路由器是基于一个传统虚通道路由器^[7, 163] 实现的，传统虚通道路由器的流水线包括四个阶段：路由计算 (RC)、虚通道分配 (VA)、交叉开关分配 (SA) 和交叉开关传输 (ST)^[7, 163]，此外，链路传输 (LT) 还需要使用一个时钟周期。为了获得较高的基准性能，传统虚通道路由器采用了一些优化技术：猜测交叉开关分配^[38]在低网络负载情况下将 VA 和 SA 并行执行；超前路由^[60, 63, 173]通过在前一跳路由器上计算下一跳的最多两个候选输出端口将 RC 从关键路径移除。DBAR 路由器采用超前信号（图3.10中的 *Bundle Signals*）^[41, 175] 编码报文的目标节点，这些超前信号比切片早一个时钟周期进入 LT 段，当切片处于 ST 时，

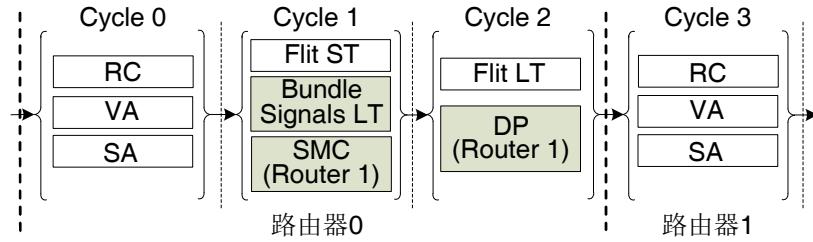


图 3.10 DBAR 路由器流水线

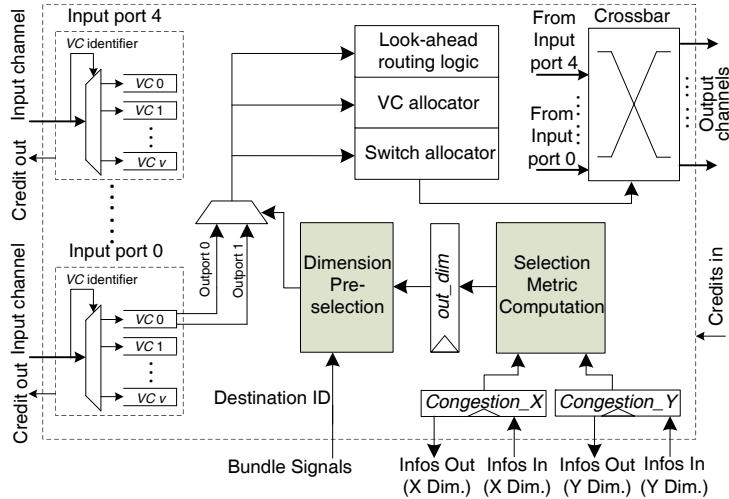


图 3.11 DBAR 路由器微结构

超前信号就进入 LT^[41, 175]。图3.10描述了 DBAR 路由器的流水线。

3.4.2.1 选择指标计算

如图3.11所示，DBAR 路由器在传统虚通道路由器上增加了选择指标计算（Selection Metric Computation, SMC）和维度预选择（Dimension Pre-selection, DP）两个模块。基于存储在 *congestion_X* 和 *congestion_Y* 中的拥塞信息，SMC 模块计算每个网络节点的最优输出端口，计算结果存储在寄存器 *out_dim* 中。DBAR 使用最短路由，任何网络节点最多只有两个处于不同维度的候选输出端口。因此，*out_dim* 寄存器为每个节点分配 1 位记录其最优端口的维度。如果值为“0”，最优端口位于 X 维度，否则，位于 Y 维度。

图3.12给出了计算第 *pos* 号节点最优输出端口的逻辑，计算逻辑只考察位于当前位置和第 *pos* 号节点所定义的最小象限内的节点的状态。在每个维度上，代表这些节点状态的 *congestion_X* 和 *congestion_Y* 的相应位被选择，它们表示每个维度的拥塞程度，根据它们的相对大小，SMC 模块设置 *out_dim* 第 *pos* 位的值，如果它们相等，SMC 模块随机选择一个维度。SMC 模块只考虑当前位置和第 *pos* 号节点所定义的最小象限内的节点的状态，因此消除了冗余信息。SMC 模块同时考虑邻居节点和远端节点的状态，因此获得了较高的适应性。

```

1:   if ( pos_x < cur_x )
2:     tmp_x[0:cur_x-pos_x-1] ← congestion_X[1:cur_x-pos_x];
3:   else if (pos_x > cur_x )
4:     tmp_x[0:pos_x-cur_x-1] ← congestion_X[cur_x+2:pos_x+1];
5:   else {
6:     out_dim[pos] ← 1;
7:     return;
8:   if ( pos_y < cur_y )
9:     tmp_y[0:cur_y-pos_y-1] ← congestion_Y[1:cur_y-pos_y];
10:  else if (pos_y > cur_y )
11:    tmp_y[0:pos_y-cur_y-1] ← congestion_Y[cur_y+2:pos_y+1];
12:  else {
13:    out_dim[pos] ← 0;
14:    return;
15:  if( tmp_x < tmp_y )
16:    out_dim[pos] ← 1;
17:  else if( tmp_x > tmp_y )
18:    out_dim[pos] ← 0;
19:  return;

```

图 3.12 SMC 模块的伪码

3.4.2.2 维度预选择

为了将端口选择从关键路径移除，维度预选择模块（Dimension Pre-selection, DP）在切片到达的前 1 个周期访问 *out_dim* 寄存器，图3.13给出了 DP 模块的结构图。DP 模块根据超前信号携带的目标节点位置读取 *out_dim* 的相应位，它使用 6 个 XOR 门和 1 个 NOR 门计算掩码信号。若目标节点位置是 *pos*，该位掩码为“1”，其它位掩码为“0”。掩码信号与 *out_dim* 进行逻辑与，最后通过一个 OR 门得到最优端口的维度。报文头切片中有上一跳计算的当前路由器的最多两个候选输出端口，当头切片到达当前路由器时，根据 DP 模块的结果选择相应的输出端口。基于逻辑功效延迟模型^[38]，DP 模块延迟大概为 8.1 FO4。如果将 DP 执行添加到 VA 阶段，关键路径从 20 FO4 上升到 28.1 FO4，因此，DBAR 路由器采用超前信号将 DP 模块延迟从关键路径移除。

3.5 实验评估

实验对时钟精确模拟器 Booksim^[163] 进行修改，以模拟第3.4.2节描述的路由器的流水线和微结构。实验评估了 DBAR 路由算法和其它多种路由算法。确定性路由算法是维序路由（DOR），自适应路由算法包括 LOCAL、NoP 和 RCA-1D（实验部分称为 RCA）。

实验使用了多种合成流量模式^[160, 163]，实验为每条虚通道配置了 5 个缓存单元，报文长度在 1 个切片和 6 个切片之间均匀分布。实验通过在 Booksim 中配置多个区域模拟负载整合模式，每个区域就像一个独立网络，该区域的注入过程被独立控制，同时其延迟和吞吐率也是单独统计的，每个区域注入的报文的目标节点在本区域内部，整个网络配置一个路由算法。

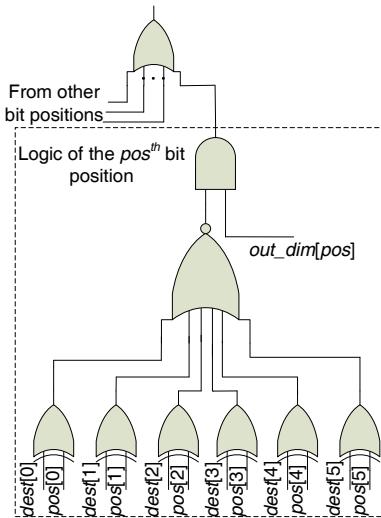


图 3.13 DP 模块的硬件实现

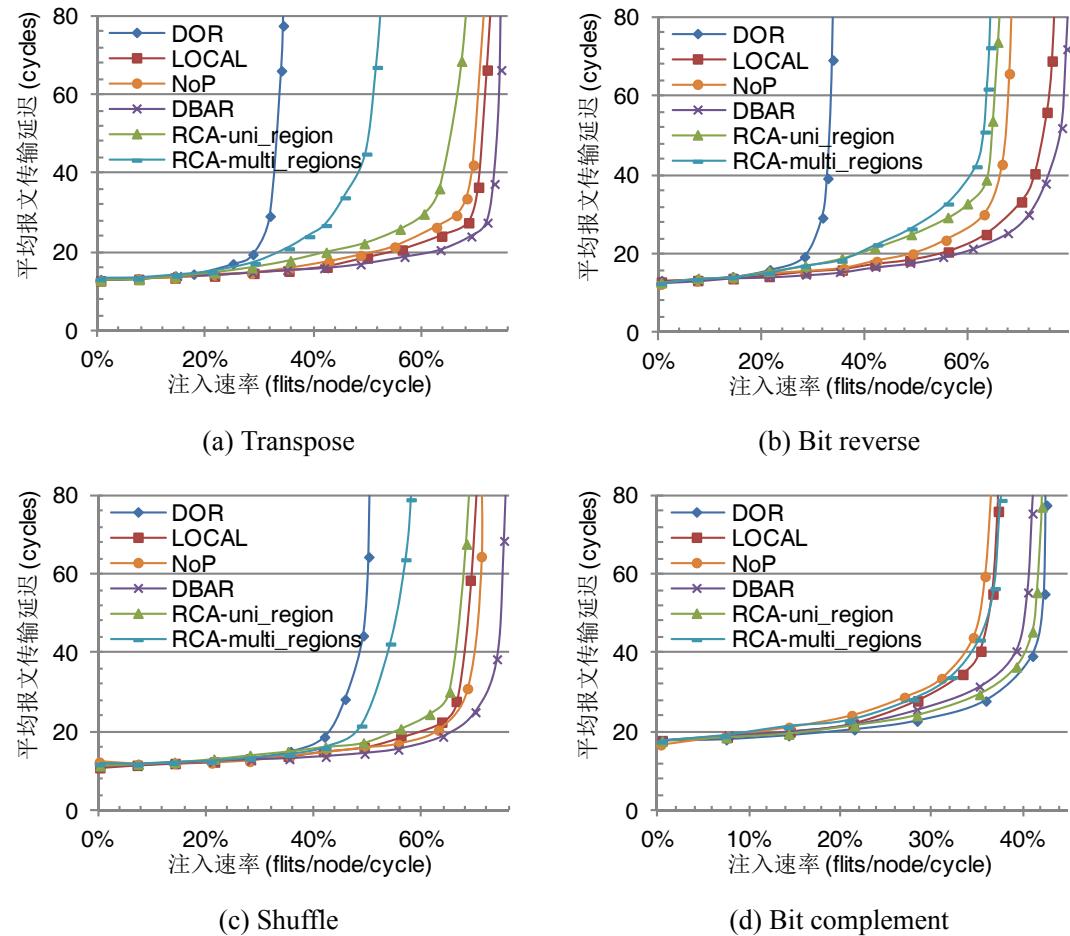
表 3.1 全系统模拟参数

| 参数名 | 参数值 |
|------------------|---------------------------------|
| 核数 | 16 |
| 一级 cache (数据或指令) | private, 4-way, 32KB each |
| 二级 cache | private, 8-way, 512KB each |
| Cache 一致性协议 | 分布式目录 MOESI 协议 |
| 网络拓扑 | 4×4 Mesh, 4 个虚拟网络 (VN), 8VCs/VN |

实验使用两个模拟器评估全系统性能：FeS2^[185] 模拟 x86 体系结构，Booksim 模拟 FeS2 产生的报文在片上网络中的传输。实验模拟了一个 16 核，拓扑结构是 4×4 mesh 的 CMP 平台，并评估了 PARSEC 测试集^[201] 在 16 线程运行时的性能。实验同时也评测了 PARSEC 测试集在负载整合模式下的性能。由于操作系统可扩展性^[202] 的限制，负载整合模式性能测试混合使用了真实程序和合成流量。Booksim 模拟了 1 个 8×8 mesh 网络，该网络包含 4 个 4×4 mesh 区域，区域 R0 传输 FeS2 产生的报文，其它区域传输 uniform random 模式的报文。之前的研究表明 CMP 平台上的简单处理核的频率可以为 5~10 GHZ^[203]，而片上网络路由器的频率则受限于分配器的速度，因此，实验假设处理器频率是网络频率的 4 倍。Cache 行包括 64 字节，网络切片大小是 16 字节。PARSEC 应用程序采用 *simsmall* 输入集，全系统性能指标是程序运行时间，表3.1给出了系统的配置参数。

3.5.1 单区域性能

本节给出了 4×4 和 8×8 mesh 网络上的单区域性能，此时，整个网络只传输一种流量模式。单区域配置实验能评估网络状态信息不足对 LOCAL 和 NoP 性能

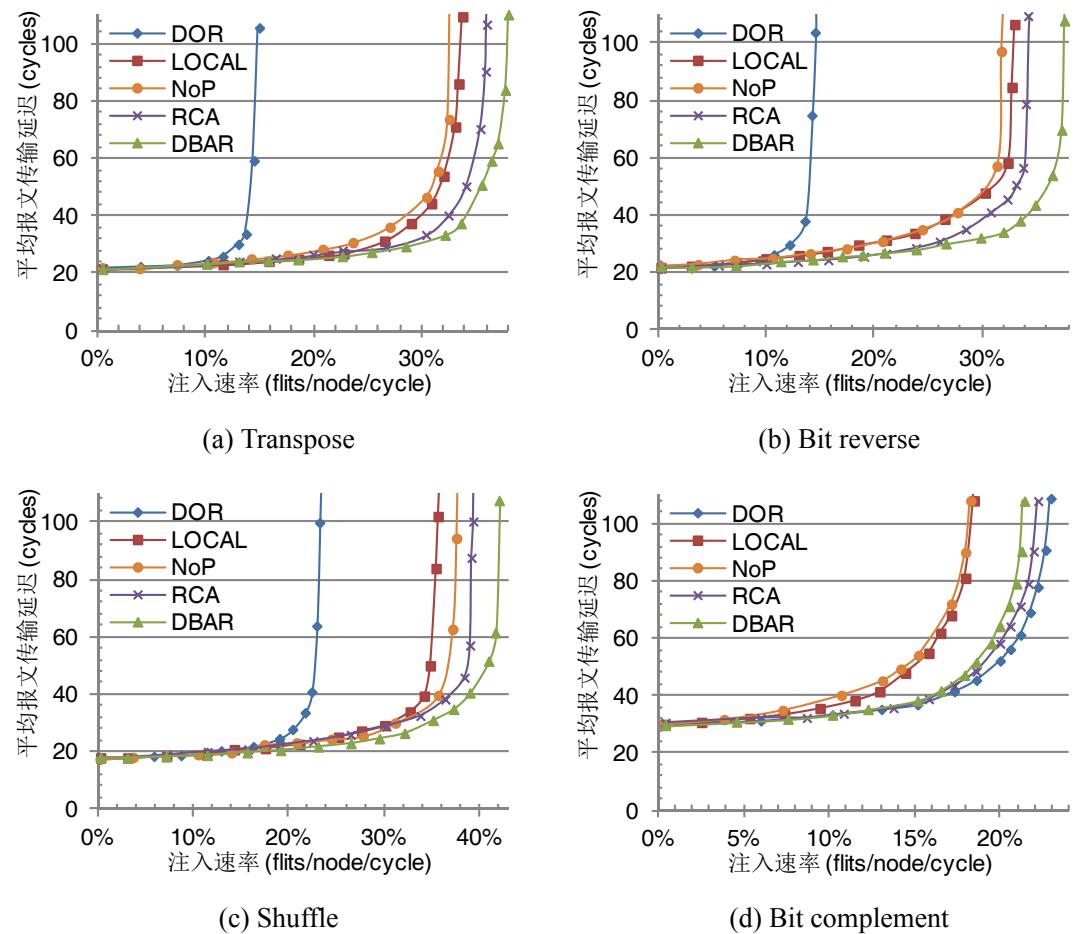
图 3.14 路由算法在 4×4 mesh 网络（区域）上的性能

的影响，同时也能评估区域内部干扰对 RCA 性能的影响。

3.5.1.1 合成流量模式结果

图3.14和3.15给出了合成流量模式在 4×4 和 8×8 mesh 上的性能，图3.14中的 *RCA-uni_region* 曲线是 RCA 在单区域配置下的性能，*RCA-multi_regions* 曲线是 RCA 在下一节所评估的多区域配置下的性能。在 4×4 mesh 的 transpose、bit reverse 和 shuffle 模式下，DBAR 的性能是所有路由算法中最好的。Bit complement 模式有所不同，RCA 在该模式下的性能比 DBAR 高 2.1%，这是因为该模式的较高报文平均跳数（4 跳）缓解了区域内部干扰对 RCA 性能的影响。LOCAL 和 NoP 使用的网络状态信息有限，因此，它们的性能在 bit complement 模式下低于其它路由算法。Transpose 模式的平均跳数较低（2.5 跳），导致此模式下 RCA 的性能低于其它自适应路由算法，DBAR、LOCAL 和 NoP 的性能比 RCA 高大约 13%。在 bit reverse 模式下，DBAR 相对于 RCA 获得了最大性能提升：21.9%。

Shuffle 和 bit complement 模式存在全局拥塞，LOCAL 无法避免全局拥塞，因

图 3.15 路由算法在单区域 8×8 mesh 网络上的性能

此，DBAR 在这两种模式下比 LOCAL 分别高 10.2% 和 8.5%。NoP 的性能受限于其忽略了邻居节点的状态，DBAR 的性能在 bit reverse 和 bit complement 模式下比 NoP 分别高 17.7% 和 11.1%。在这 4 种模式下，LOCAL 的性能基本优于 NoP，这说明了拥塞信息加权机制中给予邻近节点更高权重的合理性。

在 4×4 mesh 上，LOCAL 的性能优于 RCA；在 8×8 mesh 上，RCA 的性能优于 LOCAL。RCA 和 LOCAL 之间的性能趋势在不同规模网络中的变化是由拥塞信息传播网络的加权机制造成的。拥塞信息的权重随着距离的增加而下降，使得来自较远节点的干扰对性能的影响也会较低。越大规模网络的报文平均跳数越高，因此区域内部干扰对性能的影响随着网络规模的增大而降低。在 8×8 mesh 中，尽管加权机制减轻了干扰对性能的影响，DBAR 相对于 RCA 在 bit reverse 模式下依然获得了 11.1% 的性能提升。LOCAL 的短视性在越大规模网络对性能的影响越严重，因此，DBAR 相对于 LOCAL 的性能提升在 8×8 mesh 中更为明显，在 shuffle 和 bit complement 模式下，DBAR 的性能比 LOCAL 分别高 12.4% 和 16.5%。DBAR 相对于 NoP 的性能提升也表现出类似的趋势。DOR 只为报文提

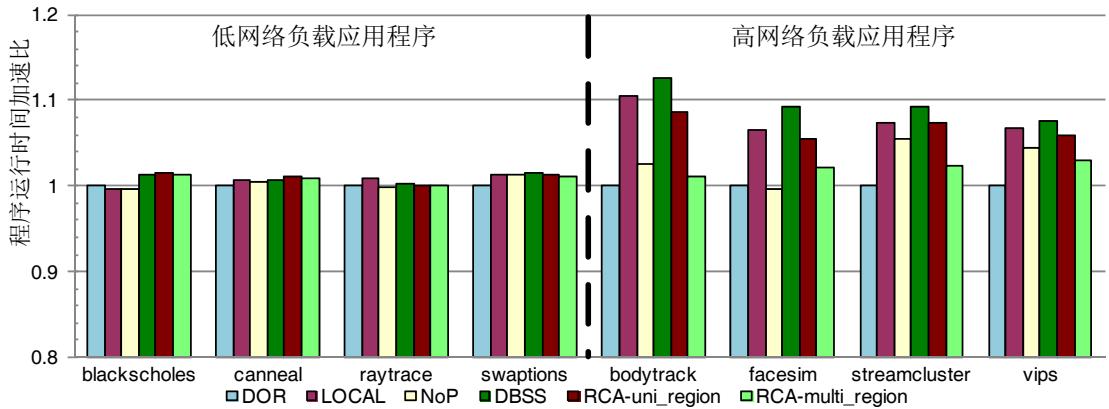


图 3.16 PARSEC 测试集的全系统加速比

供一条路径，不能避免网络拥塞，它的性能在大部分流量模式下最低。

3.5.1.2 真实应用程序结果

图 3.19 给出了各种路由算法在 PARSEC 程序下相对于 DOR 算法的全系统性能加速比。这 8 个应用程序被划分成两种类型：低网络负载和高网络负载。优化的路由设计能够支持更高的网络饱和吞吐率，但是全系统性能取决于每个程序产生的负载和流量模式^[204]。具有较高网络负载和较多猝发流量的应用程序能够从优化的路由设计上获得性能提升。私有二级 cache 能够载入 blackscholes、canneal、raytrace 和 swaptions 的工作集，这些程序的网络负载较低，因此，不同路由算法的性能基本相似。其它 4 个程序具有大量的猝发流量，增强了网络层优化对性能的影响，支持较高网络吞吐率的路由算法能够给它们带来性能提升。全系统模拟的网络规模较小，因此，DBAR 和 LOCAL 在这 4 个程序下的性能最好，DBAR 的性能比 DOR 平均高 9.6%，最大提升幅度是 12.5%。RCA 的性能优于 NoP。NoP 在 facesim 下的性能低于 DOR，这是因为 NoP 忽略了邻居节点的状态，选择了次优的输出端口。

3.5.2 多区域性能

本节实验使用了三种多区域配置，包括两种规则区域配置（图3.5和图3.17）和一种非规则区域配置（图3.18），实验只关注区域 R0 的性能。

3.5.2.1 小型规则区域结果

在第一种和第二种配置中（图3.5和图3.17），区域 R1、R2 和 R3（只有图3.5具有区域 R3）传输 uniform random 模式，它们的注入率是 4%，实验测试区域 R0 在不同流量模式下的性能。对于规则区域配置，LOCAL、NoP 和 DBAR 在选择输出端口时只考虑了属于同一区域的节点的状态，它们都没有区域间干扰，因

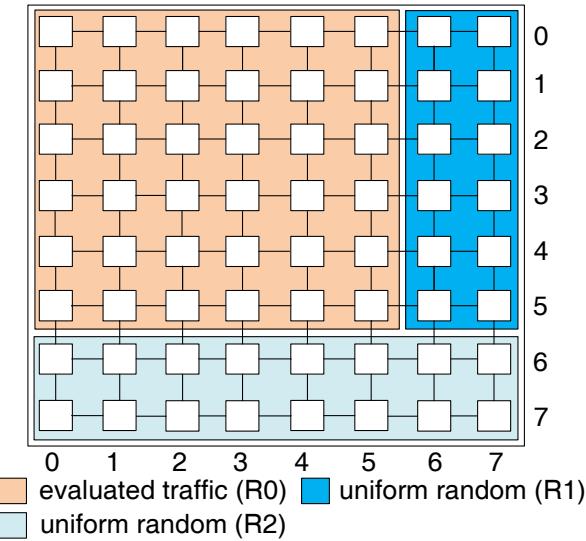


图 3.17 中型规则区域配置

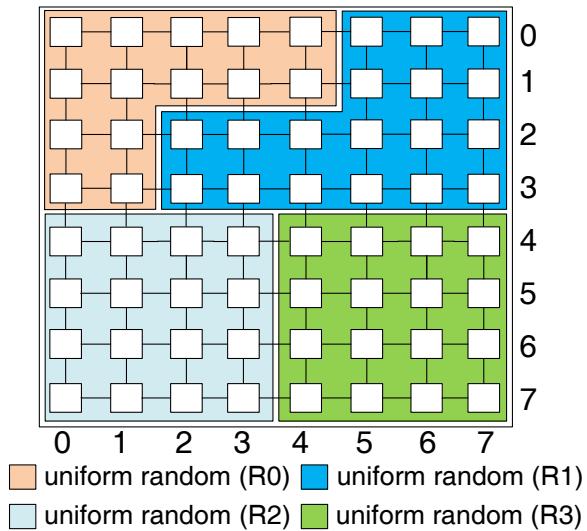
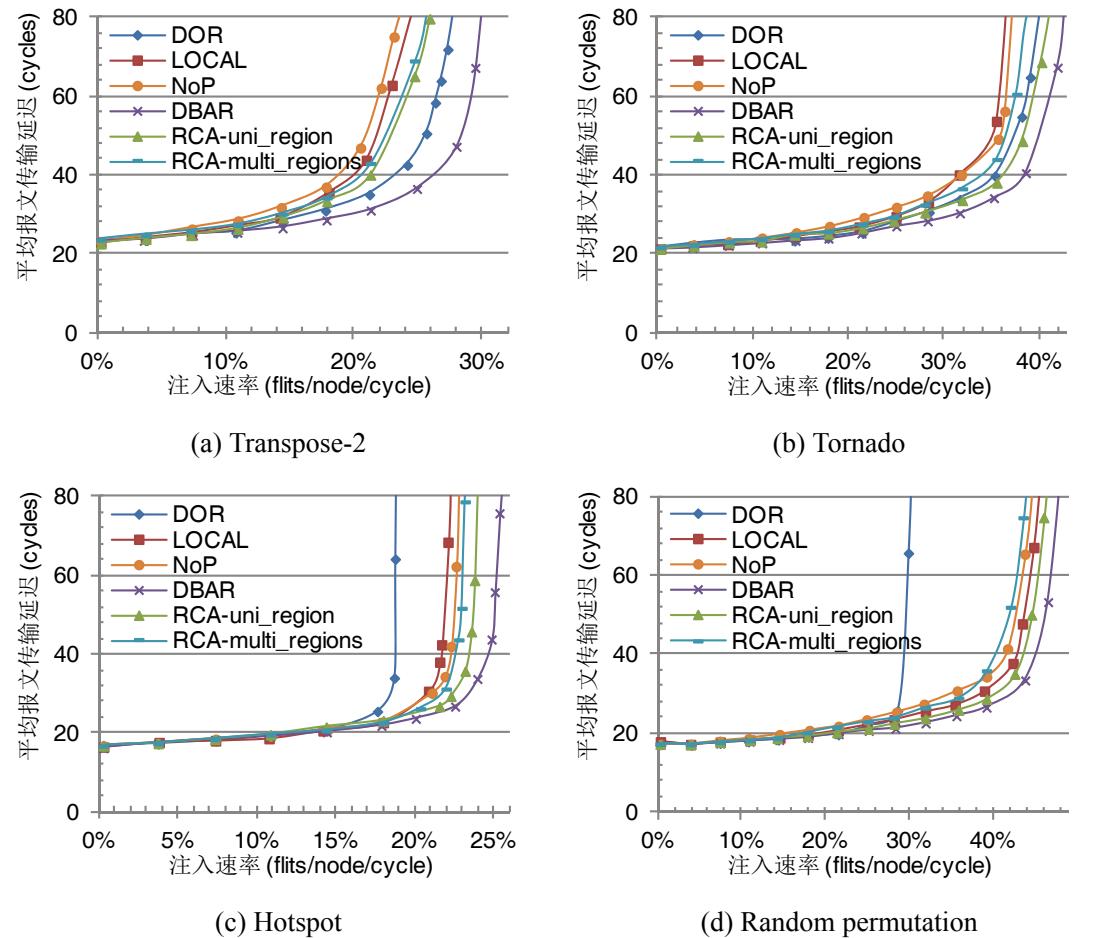


图 3.18 非规则区域配置

此这三种算法和 DOR 算法的性能与图3.14相同。RCA 的输出端口选择考虑了属于同一维度的所有节点的状态，它的性能受到区域间干扰的影响，图3.14中的 *RCA-multi_regions* 曲线是 RCA 在多区域配置下的性能。

在多区域配置情况下，RCA 的性能不仅受到区域内干扰的影响，而且还受到区域间干扰的影响。因此，RCA 在多区域配置下出现了性能下降，*transpose* 和 *shuffle* 模式分别出现了 22.7% 和 16.9% 的性能下降。*Bit reverse* 模式的区域内干扰已很大程度地降低了 RCA 的性能，隐藏了区域间干扰的负面影响，因此，RCA 在该模式下的性能下降较少。DBAR 在多区域配置时依然维持自己的性能，其相比于 RCA 取得了更大的性能提升。在这 4 种流量模式下，DBAR 相对于 RCA 的平均性能提升是 25.2%，最大的性能提升是 *transpose* 模式下的 46.1%。图3.16中

图 3.19 路由算法在 6×6 mesh 网络（区域）上的性能

的 *RCA-multi_regions* 显示 RCA 的性能在 4 个高网络负载应用程序下都出现了下降，最大的性能下降是 *bodytrack* 上的 7.3%。

区域边缘的一些输入端口总是处于空闲状态，它们严重影响了 RCA 的性能。比如说，没有任何报文从西边输入，路由器 (0,4) 的西输入虚通道总是处于空闲状态。结合拥塞信息加权机制，这些边缘端口上的 8 条空闲虚通道掩盖了区域 R1 和 R2 内部路由器的状态对区域 R0 的影响。因此，当 R1 中的注入率从 4% 提高到 64% 时，R0 的饱和吞吐率只是从 50% 下降到 47%（图3.6）。

3.5.2.2 中型规则区域结果

图3.19给出了第二种规则多区域配置（图3.17）的性能，这个配置反应了区域规模与算法性能之间的关系。实验使用 4 种合成流量模式：*transpose-2*^[57]、*tornado*、*hotspot* 和 *random permutation*。*Hotspot* 模式在网络中选定 3 个热点节点，它们接收的报文数量比其它节点高 20%，其它节点之间按照 *uniform random* 模式通信。*Random permutation* 是从总共 $36!$ 个可能的置换（permutation）模式中随机得到的 1000 个置换的平均^[59]。

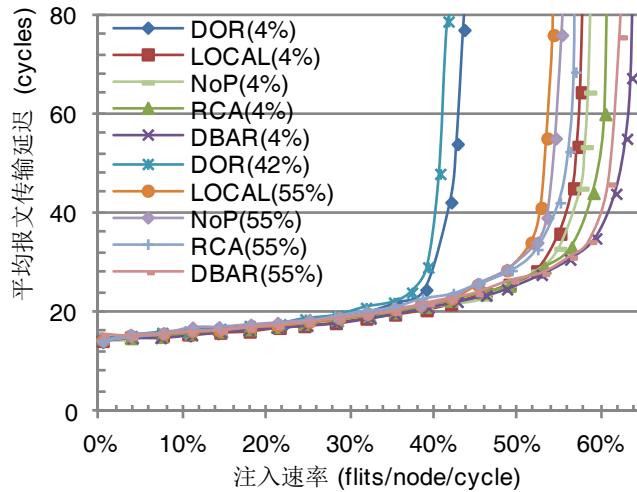


图 3.20 路由算法在非规则区域上的性能

DBAR 在所有流量模式下都获得了最好性能。与 4×4 mesh 不同，LOCAL 和 NoP 的短视性在中型规模网络中开始限制它们的性能，导致 RCA 的性能优于 LOCAL 和 NoP。尽管图3.19中的区域间干扰依然会降低 RCA 的性能，中型规模网络更高的平均跳数和拥塞加权机制缓解了它的影响，因此，RCA 的性能下降没有图3.14中明显，最大的性能下降是 random permutation 模式下的 7.3%。

3.5.2.3 非规则区域结果

在图3.18的非规则区域配置中，区域 R0 和 R1 的通信隔离边界是包含它们的最少矩形，这两个区域注入的报文共享某些网络链路，因此，区域 R1 的通信会影响区域 R0 的性能。图3.20给出了当 R2 和 R3 的注入率维持 4%，R1 的注入率从 4% 上升到 55% 时 R0 的性能，所有区域都传输 uniform random 流量模式。

在区域 R1 的各种注入率下，DBAR 的性能始终最好。当区域 R1 中的注入率较低时，RCA 的性能仅次于 DBAR。区域 R0 有两行具有 5 个路由器，此时 LOCAL 和 NoP 的短视性不能有效避免远端拥塞。随着 R1 注入率的上升，所有算法的性能都会下降。DBAR 为 R0 和 R1 提供了最好的隔离性，其性能下降最少，当 R1 的注入率从 4% 上升到 55% (DOR 是 42%) 时，DOR、LOCAL、NoP、RCA 和 DBAR 的性能分别出现了 7%、7%、6.8%、6.7% 和 4% 的性能下降。在非规则区域配置中，区域 R0 和 R1 并没有完全隔离，R0 的通信会受到 R1 通信的影响，DBAR 正确地考虑了这种区域间的相关性。

3.5.2.4 Mesh 网络实验评估小结

应用程序根据自身并行度映射到不同规模的区域上，RCA 的干扰信息在小型区域中降低了它的性能，而 LOCAL 和 NoP 的短视性在大型区域中限制了它们的性能。在所有评估的配置下，DBAR 的性能最好，同时，在非规则多区域配置下，

表 3.2 DBAR 平均性能提升

| 网络规模 | LOCAL | NoP | RCA | RCA_multi |
|--------------|-------|-------|-------|-----------|
| 4×4 | 7.2% | 8.8% | 10.4% | 25.2% |
| 8×8 | 12.6% | 14.9% | 4.7% | - |
| 非规则 | 16.5% | 14.3% | - | 6.8% |

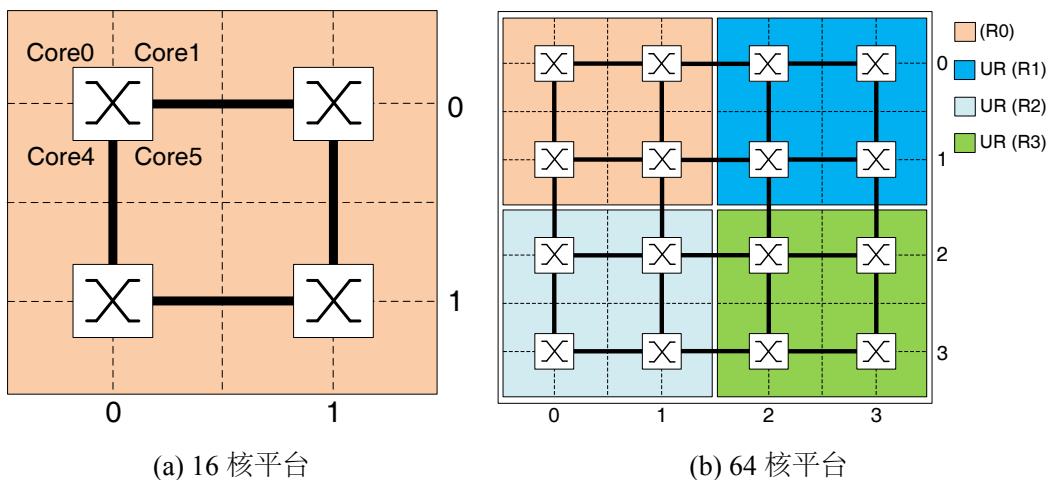


图 3.21 CMesh 网络的配置

其性能下降最低。表3.2给出了 DBAR 相对于其它路由算法获得性能提升。

3.5.3 Concentrated Mesh 性能

3.5.3.1 实验配置

上述各节讨论了路由算法在 mesh 网络上的性能，本节将讨论扩展到 concentrated mesh (CMesh) 网络^[27, 156] 上。作为一个研究案例，本节使用了基数为 4 的 CMesh 网络^[27, 156]，即每个路由器上集成 4 个处理器核，如图3.21(a)所示。在该图中，Core0、Core1、Core4 和 Core5 四个核聚集在路由器 (0,0) 上，每个核都有专门的 injection/ejection 链路连接到路由器上。基于一个 CMesh 链路延迟模型^[156]，本节假设网络链路的延迟是 2 个时钟周期，injection/ejection 链路的延迟是 1 个时钟周期，路由器的流水线与第3.4节中的描述相同。实验评估了 16 核 CMesh 网络和 64 核 CMesh 网络（图3.21(b)），在 64 核 CMesh 网络中，实验既测试了单区域配置的性能，又测试了多区域配置的性能。对于多区域配置情况（图3.21(b)），区域 R1、R2 和 R3 传输 uniform random 通信，实验测试区域 R0 的性能。

3.5.3.2 性能

图3.22给出了路由算法在 CMesh 上的性能。 2×2 CMesh 的每个维度上只有两个路由器，LOCAL、DBAR 和 RCA 算法在选择输出端口时使用了相同的网

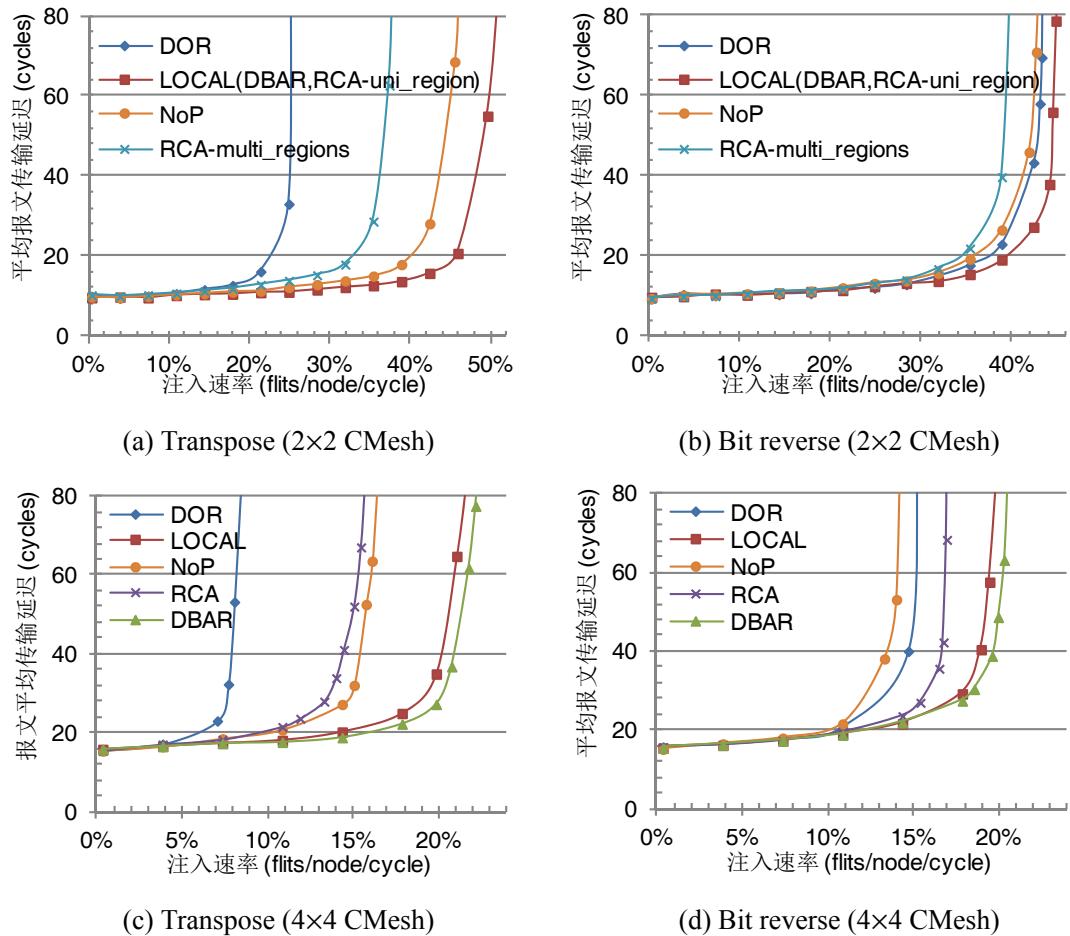


图 3.22 路由算法在 CMesh 网络（区域）上的性能

络状态信息，因此它们的性能相同。比如说，在图3.21(a)中，当报文需要从路由器(0,1)发送到路由器(1,0)时，这三种算法都是考虑路由器(0,0)的东输入端口和路由器(1,1)的北输入端口的状态。NoP 有所不同，它比较路由器(1,0)的北输入端口和南输入端口的状态。由于 NoP 忽略了邻居节点的状态，其性能比 LOCAL、DBAR 和 RCA 要低。DOR 在 transpose 模式下的性能低于其它路由算法（图3.22(a))。在 4×4 mesh 网络的 bit reverse 模式下，DOR 与自适应路由算法的性能差异是最好的自适应路由算法的 1/3（图3.14(b))，DOR 与自适应路由算法的性能差异在 2×2 CMesh 下有所下降（图3.22(a))。这是因为 CMesh 在路由器上集成了多个核，降低报文的传输跳数，从而减少了自适应路由算法实现负载均衡的机会。

在多区域配置的 4×4 CMesh 中，区域间干扰降低了 RCA 的性能，RCA 的性能下降在 transpose 模式下是 26.2% (*RCA-uni_region* vs. *RCA-multi_regions*)。在 4×4 mesh 的 bit reverse 模式下，由于区域内干扰已很大程度地降低了 RCA 的性能，区域间干扰对 RCA 性能的影响较少（图3.14(b))，但是 2×2 CMesh 没有区域内干扰，因此，区域间干扰给 RCA 在 bit reverse 模式下带来了 9.2% 的性能下降。

路由算法在 4×4 CMesh 的单区域配置下的性能趋势（图3.22(c)和图3.22(d)）与 4×4 mesh 中的情况基本相同（图3.14(a)和图3.14(b)），DBAR 和 LOCAL 的性能最好，RCA 的性能受限于区域内部干扰。CMesh 的网络链路延迟是 injection/ejection 链路延迟的两倍，因此在 CMesh 中做出明智的输出端口选择更为重要，其导致区域内部干扰在 CMesh 中对 RCA 的影响更为严重。DBAR 在 4×4 CMesh 的 transpose 模式下的性能比 RCA 高 29.4%，而在 4×4 mesh 的 transpose 模式下（图3.14(a)），DBAR 的性能只比 RCA 高 7.4%。上述实验结果表明良好的选择策略在 CMesh 网络中一样重要。

3.6 硬件开销讨论

3.6.1 连线资源

自适应路由器使用一些额外连线资源传输网络拥塞信息。假设 8×8 mesh 网络的每个端口配置 8 条虚通道，DBAR 在每个维度上使用 8 位连线传输拥塞信息；RCA 在每个方向上使用 8 位连线，在每个维度上需要 16 位连线；NoP 在每个方向上使用 12 位连线，在每个维度需要 24 位连线；LOCAL 在每个方向上使用 3 位连线，其每个维度上需要 6 位连线。DBAR 具有适中的额外连线开销。假设 128 位的切片位宽^[109]，DBAR、RCA、NoP 和 LOCAL 的额外连线资源开销分别是 3.125%、6.25%、9.375% 和 2.34%。片上丰富的连线资源可以满足这些需求。

3.6.2 路由器开销

相比于传统虚通道路由器，DBAR 路由器增加了 SMC 模块、DP 模块和 3 个寄存器。在 8×8 mesh 中，DBAR 路由器使用两个 7 位移位器对齐从 *congestion_X* 和 *congestion_Y* 读出的信息。*Congestion_X* 和 *Congestion_Y* 是 9 位寄存器，*out_dim* 是 64 位寄存器，与路由器中已经存在的缓存资源（5 个输入端口、每个输入端口上设置 8 条虚通道、每个虚通道包含 5 个缓存单元、128 位切片）相比，这些寄存器只带来了 0.3% 的额外存储开销。

3.6.3 功耗和能量延迟积

本节使用一个片上网络功耗模型^[205] 计算链路、输入缓存和路由器逻辑的功耗，实验也测试了拥塞信息传播网络的功耗。网络各模块的活跃指数通过 Booksim 获得，实验使用 ITRS 32nm 工艺参数^[12]，假设网络频率是 1GHz。

图3.23给出了 8×8 mesh 中 transpose 模式的功耗结果，由于 DOR 不支持高于 20% 的注入率，其没有 30% 和 35% 注入率下的实验结果。自适应路由算法需要

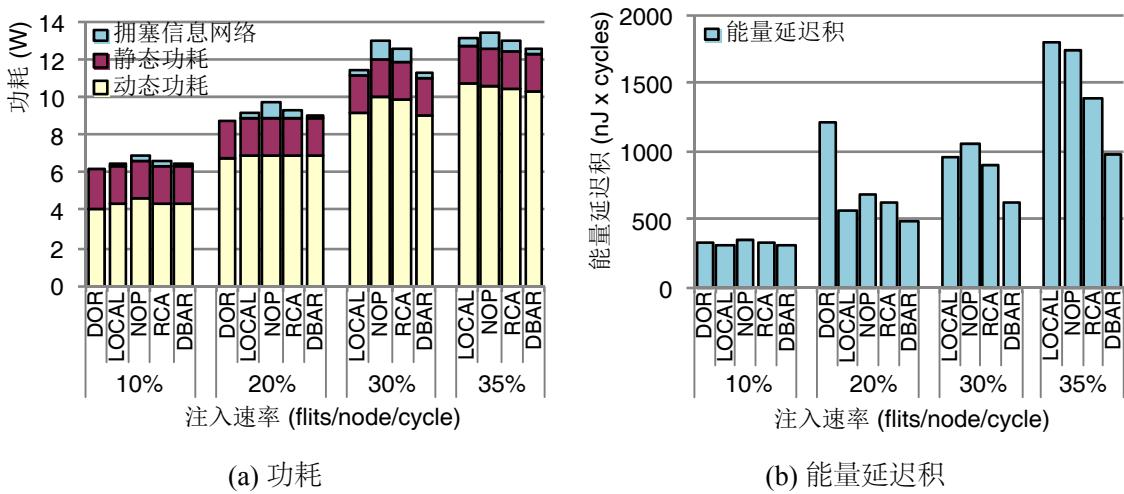


图 3.23 8×8 mesh 网络 transpose 模式的功耗和能量延迟积

一些的额外硬件开销（如拥塞信息传播网络），因此，它们的功耗高于 DOR。由于 LOCAL 和 DBAR 使用的额外连线资源最低，它们的功耗在自适应算法中最低，NOP 的功耗最高。LOCAL 使用的连线资源较低，但是这些连线的翻转率较高。在 20% 注入率时，DBAR 的拥塞信息传播链路的翻转率是 15.8%，而 LOCAL 的拥塞信息传播链路的翻转率是 17.5%。DBAR 的较低翻转率降低了功耗开销，使得 LOCAL 的功耗在 35% 注入率时略高于 DBAR。自适应路由算法可以加速报文的传输，它们在能量延迟积（Energy-delay product）中的优势更为明显，如图3.23(b)所示，DBAR 在中等（20%）或高注入率（30% 和 35%）时获得了较大的能量延迟积下降。

3.7 进一步讨论

3.7.1 拥塞信息传播网络带宽

表3.3测试了 DBAR 采用 1 位、2 位和 3 位宽度拥塞信息传播网络时的性能。拥塞信息传播网络带宽的增加只能带来少量性能提升，同时这些提升随着网络规模的增大而降低。选择策略只需要得到两个候选端口的相对拥塞程度，而不需要精确计算空闲虚通道数。为了验证这一点，本节在模拟器中同时配置 1 位、2 位和 3 位带宽的拥塞信息传播网络，采用 uniform random 模式，测试选择策略在 1 位或 2 位配置时所选择的端口与 3 位配置时所选择的端口的差异比例。当注入率非常低时（2%），它们的差异分别是 8.8% 和 7.6%，这些差异主要是由随机选择两个具有相同拥塞程度的候选端口带来的。随着注入率的提高，不同配置的差异逐渐降低，当网络饱和时，它们的差异分别是 1.8% 和 1.1%。如此小的差异也说明了没有必要精确计算空闲虚通道数。

表 3.3 DBAR 在不同带宽拥塞信息传播网络下的性能

| | | bitcomp | transpose | bitrev | shuffle | uniform | bitrot | tornado | randperm | Avg Imp |
|-----|--------|---------|-----------|--------|---------|---------|--------|---------|----------|---------|
| 4×4 | 1-bit | 39.6% | 74.0% | 78.5% | 74.8% | 71.2% | 81.2% | 72.8% | 52.9% | - |
| | 2-bits | 41.2% | 74.5% | 78.6% | 76.8% | 71.3% | 82.4% | 72.9% | 53.1% | 1.33% |
| | 3-bits | 42.0% | 74.8% | 78.7% | 77.4% | 71.3% | 82.6% | 73.4% | 53.1% | 1.92% |
| 8×8 | 1-bit | 21.2% | 35.4% | 36.0% | 40.8% | 35.6% | 43.2% | 25.2% | 28.6% | - |
| | 2-bits | 21.4% | 36.4% | 38.5% | 40.9% | 36.4% | 43.3% | 25.5% | 28.6% | 1.12% |
| | 3-bits | 21.5% | 36.6% | 38.7% | 41.6% | 36.6% | 43.4% | 25.7% | 28.6% | 1.72% |

3.7.2 DBAR 的可扩展性

DBAR 的额外连线资源随着网络规模的增大而线性上升, 在一个 $N \times N$ mesh 中, 每个维度上需要使用 N 位拥塞信息传播线。在 128 位切片宽度的 16×16 mesh 中, 连线资源的开销是切片网络的 6.25%。DBAR 路由器中所使用的寄存器位宽也与网络规模成线性关系。DP 模块采用树形结构, 其延迟与网络规模成对数关系, 但是 DP 模块不在关键路径上。因此, DBAR 具有比较好的可扩展性。

3.7.3 拥塞信息传播延迟

除了消除干扰以外, DBAR 拥塞信息传播网络每跳只需要 1 个时钟周期, 而 RCA 的拥塞信息传播网络每跳需要 2 个时钟周期。为了分析传输延迟对性能的影响, 本节测试了 DBAR 在拥塞信息传播每跳需要 2 个时钟周期时的性能。相比于每跳 1 个周期, DBAR 在 shuffle 模式的性能下降了 5%。

3.8 本章小结

本章主要研究了负载整合模式下的路由算法设计, 首先对已有设计的局限性进行了分析, 这些设计包括局部自适应路由算法和全局自适应路由算法。局部自适应路由算法对网络状态的短视性限制了它的性能, 而全局自适应路由算法的性能受限于干扰信息, 在负载整合模式下, 这些干扰信息耦合了多个同时运行的应用程序。本章针对这两方面的问题, 提出了基于目标的自适应路由 (Destination-Based Adaptive Routing, DBAR) 算法, 该算法将目标节点位置信息集成到输出端口选择过程中。通过使用一个低开销的拥塞信息传播网络, DBAR 一方面提供了较高的适应性, 另一方面为应用程序提供了动态隔离性。应用程序之间的动态隔离性在众核平台同时运行多个应用程序时至关重要, 它是保证应用程序性能可预测性的关键。实验结果表明 DBAR 在所有配置下都获得了较好的性能。DBAR 的额外连线资源需求比较低, 同时也降低了能量延迟积。

第四章 面向 Cache 一致性通信的完全自适应路由算法

由于面积和功耗的限制，片上网络路由算法只能使用少量的虚通道，在虚通道数目受限环境中设计完全自适应路由算法面临着许多新的挑战。第一，基于之前的死锁避免理论设计的完全自适应路由算法需要采用保守虚通道分配策略：只有空虚通道才能被重新分配，该要求严重限制了路由算法的性能。本章提出了全报文发送（Whole Packet Forwarding, WPF）虚通道分配策略，其允许一条非空虚通道被重新分配，因此带来了性能提升。本章证明了如果完全自适应路由算法在采用保守虚通道分配时不存在死锁，则采用 WPF 策略一样不会存在死锁。因此，WPF 策略对之前的死锁避免理论进行了重要的扩展。第二，虚通道受限环境下的完全自适应路由算法应该提供较高的路由灵活性。基于 WPF 策略，本章进一步给出了一种在较低硬件开销条件下支持较高路由灵活性的路由算法设计。

4.1 引言

路由算法对片上网络性能具有至关重要的影响，它不仅决定了报文的传输延迟，而且还决定了片上网络的吞吐率，人们已提出了多种高效的片上网络路由算法^[50, 57, 59, 61, 63, 144, 148]。路由算法设计除了需要考虑性能以外，还需要考虑正确性。具体而言，所设计的路由算法必须是无死锁的。Cache 一致性片上网络有两种类型的死锁：网络层死锁和协议层死锁。一种通用的消除网络层死锁的方法是按照死锁避免理论设计路由算法。当前存在许多不同的死锁避免理论，包括针对部分自适应路由的死锁避免理论^[144, 159, 160, 198] 和针对完全自适应路由的死锁避免理论^[162, 199, 206–209]。尽管这些理论大部分是面向片外网络提出的，它们依然被大量应用于今天的片上网络设计^[50, 57, 59, 61, 63, 144, 148]。

然而，片上网络的报文特性与片外网络有很大不同。大量的片上连线资源增加了切片的宽度，降低了报文的切片数，导致片上网络传输的大部分是短报文。与之相反，片外网络的连线资源受限于芯片的引脚数目。典型片外网络路由器（如 Alpha 21346^[210]）的切片位宽是 32 位，而片上网络的切片位宽一般是 128 位^[109] 或者 256 位^[36]。在 128 位切片宽度下，大量一致性控制消息由于只包含地址和控制信息，而不携带数据，可以编码成单切片报文；携带 64 字节 cache 行的数据消息编码成 5 切片报文。图4.1评测了 PARSEC 测试集^[201] 在目录 MOESI 协议下的单切片报文比例，这 10 个应用程序的平均单切片报文比例是 78.7%。

由于面积和功耗的限制，片上缓存资源比片外缓存资源要珍贵得多^[45, 170]，因此，片上网络一般采用基于切片的虫孔交换机制^[165] 来降低对缓存的需求。尽

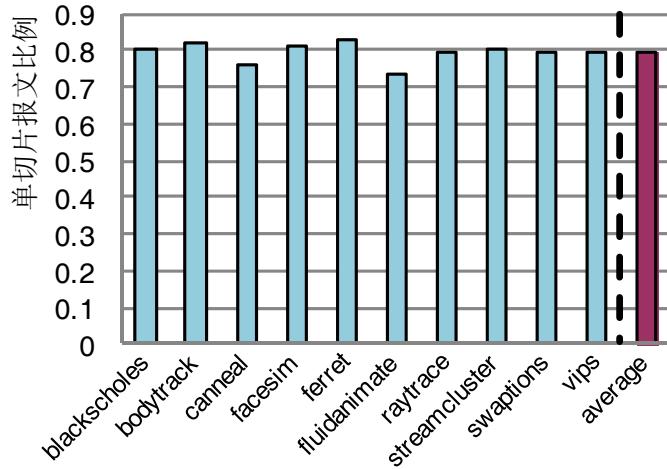


图 4.1 PARSEC 测试集单切片报文比例

表 4.1 物理或虚拟网络数目

| 工业界产品 | | | GEMS 模拟器中 cache 一致性协议 | | |
|--------------|-----------------|------------------------------|-----------------------|-----------------|-------------|
| Alpha 21364 | TILE64 | TRIPS | MESI directory | MOESI directory | MOESI token |
| 1 PN (7 VNs) | 5 PNs (1 VN/PN) | 2 PNs. OCN: 4 VNs, OPN: 1 VN | 5 VNs | 4 VNs | 4 VNs |

管片上网络的缓存资源相当有限，其仍然需要配置多个独立的物理或虚拟网络来传输不同类型的消息以避免协议层死锁。表4.1左侧给出多个工业界片外或片上网络产品中所使用的物理 (Physical network, PN) 和虚拟网络 (Virtual network, VN) 数，该表右侧给出了 GEMS 模拟器^[188] 中多种一致性协议的虚拟网络数；消除协议层死锁一般需要使用 4 到 5 个虚拟网络。由于更多的虚通道需要更高的缓存资源，同时也增加了路由器复杂性，此时每个虚拟网络只能配置少量虚通道，即片上网络路由算法能使用虚通道数一般比较受限。比如说，片上网络芯片 TILE64^[22] 和 TRIPS^[109] 的每个虚拟网络都只配置了一条虚通道。

大量短报文和受限虚通道数这两个特性给片上网络完全自适应路由算法设计提出了新的挑战。基于已有的虫孔交换网络死锁避免理论设计的完全自适应路由算法需要采用保守虚通道分配策略：虚通道只有在其为空的时候才能被重新分配^[162, 199, 206–209]。该保守分配策略是为了防止死锁，但是它在网络中存在大量短报文时会给路由算法的性能带来了很大的负面影响^[144]。图4.2给出了三种路由算法在配置两条虚通道时的性能，PSF 是完全自适应路由算法，DOR 是确定性路由算法，Odd-even 是部分自适应路由算法。尽管完全自适应路由具有更大灵活性，同时也能获得更好的负载均衡，但是它的性能却低于确定性路由和部分自适应路由。确定性路由和部分自适应路由所采用的积极虚通道分配策略提升了它们的性

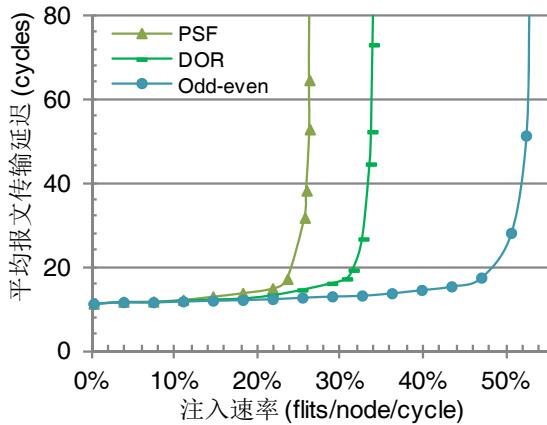


图 4.2 路由算法在 bit reverse 流量模式下的性能

能。因此，需要扩展已有的死锁避免理论以提高完全自适应路由算法的性能。

本章为完全自适应路由算法设计了全报文发送 (Whole Packet Forwarding, WPF) 虚通道分配策略，其允许非空虚通道被重新分配。具体而言，如果一条非空虚通道有足够的空闲缓存来接收整个报文，则允许其被重新分配。全报文发送策略可以被视为在虫孔交换网络中采用基于报文的交换机制，这种混合流控机制解决了保守虚通道分配策略的缺陷，并获得了较大性能提升。本章证明了如果完全自适应路由算法在采用保守分配策略时没有死锁，则采用全报文发送也不会出现死锁，因此，全报文发送对已有的死锁避免理论进行了重要扩展。

基于全报文发送策略，本章进一步给出了一种高效的完全自适应路由算法，其能在较低硬件开销条件下支持较高的路由灵活性。实验结果表明，与保守分配策略相比，全报文发送在合成流量模式下平均获得了 88.9% 的性能提升；同时，其在网络负载较重的 PARSEC 程序上的平均加速比是 21.3%，最大加速比是 37.8%；本章设计的算法也优于确定性路由算法和部分自适应路由算法。

4.2 相关研究

本节主要讨论死锁避免理论和完全自适应路由算法设计的相关研究。

4.2.1 死锁避免理论

片上网络一般采用虫孔交换^[165]降低对缓存的需求^[1, 166, 167]，本节主要讨论虫孔交换网络的死锁避免理论。Dally 和 Seitz 提出了一种开创性的死锁避免理论^[198]，该理论可以用于设计无死锁的确定性路由算法和部分自适应路由算法；Duato 给出了路由子函数的概念，并提出一种设计完全自适应路由算法的方法^[162, 199]；Lin 等采用消息流模型分析死锁特性^[207]；Schwiebert 和 Jayasimha 提出了通道等待图的概念，并且将其用于路由算法的死锁分析^[208]；最近 Verbeek 和

Schmaltz 基于静态分析方法给出了判定路由算法无死锁的充分必要条件^[209, 211]。这些理论^[162, 199, 206–209]可以用于设计无死锁的完全自适应路由算法。

这些完全自适应路由死锁避免理论的一个局限性是它们都限制只有空虚通道才能被重新分配^[162, 199, 206–209]。这个限制的目的是保证当报文进入虚通道时一定能到达虚通道头部，从而有机会进入一条“无死锁”路由路径。然而，由于片上网络上大量的短报文和受限的虚通道数，严格遵守这个要求将会带来很大的性能损失。已有的死锁恢复设计^[212]或理论^[213]能够解决这个局限性，它们无需采用保守虚通道分配策略，而是允许先出现死锁，然后恢复到正常状态^[212, 213]。本章提出的全报文发送与这些死锁恢复机制不一样，全报文发送扩展了已有的死锁避免理论，它从设计上消除了死锁。就我们所知，基于已有的死锁避免理论设计的虫孔网络完全自适应路由算法都需要采用保守分配策略^[162, 199, 206–209]，而本章所提出的全报文发送是第一个允许多个报文同时存在于一条虚通道的虫孔网络完全自适应路由算法设计。

4.2.2 完全自适应路由算法设计

Duato 理论^[162, 199]是一种广泛使用的完全自适应路由算法设计理论，该理论使用两种类型的虚通道：适应性虚通道和逃逸虚通道。当适应性虚通道之间出现循环等待时，报文必须有机会“逃逸”到无死锁的虚通道，即逃逸虚通道上。逃逸虚通道采用更具约束性的路由子函数以保证它们之间不会出现循环依赖，该路由子函数一般是维序路由；报文只有在其输出端口符合维序路由时才能使用逃逸虚通道。

许多基于 Duato 理论设计的路由算法包含两部分：路由函数和输出端口选择策略^[50, 63, 148, 210]。如果输出端口选择策略选择了某个输出端口，报文只能请求该输出端口的虚通道。这种设计带来一个限制：一旦报文进入逃逸虚通道，其后续路由只能使用继续逃逸虚通道，该报文将丧失路由灵活性。这个限制在虚通道数受限的 cache 一致性片上网络中会带来较大性能损失。然而，根据 Duato 理论，如果能保证报文在任何时候都有机会使用逃逸虚通道，则允许进入逃逸虚通道的报文重新回到适应性虚通道^[162, 199]。基于这些观察，本章提出了一种具有较低硬件开销的，能维持路由灵活性的设计。

4.3 研究动机

本节论述完全自适应路由算法的一些要求，同时也分析这些要求对性能的影响，并给出本章的研究动机。

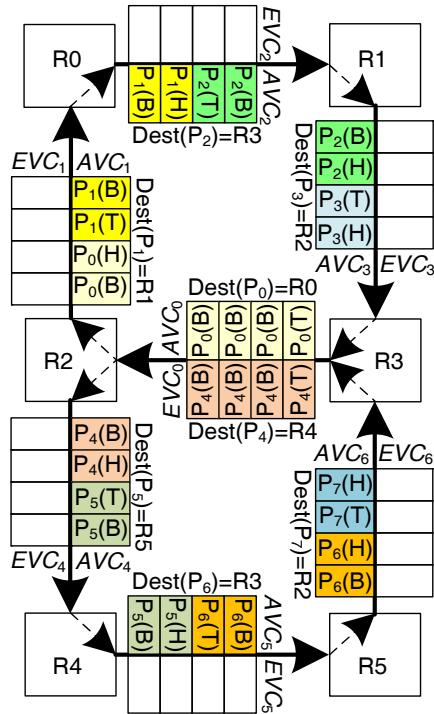


图 4.3 完全自适应路由算法的死锁实例

4.3.1 虚通道分配策略

完全自适应路由算法的一个局限是：一条虚通道在任何时候只能存储属于同一报文的切片，即只有空虚通道才能被重新分配。由于完全自适应路由算法允许虚通道之间出现循环等待，这个要求是合理的，它用于防止死锁的出现。比如说，基于 Duato 理论的完全自适应路由算法的适应性虚通道间可能存在循环等待，如果允许多个报文存在同一条虚通道内，则会出现如图4.3所示的死锁^[162, 199, 213]。这个例子配置了两条虚通道：一条适应性虚通道（Adaptive VC, AVC）和一条逃逸虚通道（Escape VC, EVC）。死锁是由适应性虚通道之间的循环等待导致的，配置更多的虚通道并不能消除这种死锁。

图4.3共包含 8 个报文：报文 P_0 到 P_7 。处于 AVC_1 内的 P_0 的头切片在 P_1 的尾切片之后，报文 P_1 、 P_2 、 P_4 、 P_5 和 P_6 的情况类似。 P_3 和 P_7 的头切片位于 AVC_3 和 AVC_6 的头部，但是由于 AVC_0 和 EVC_0 已被占用， P_3 和 P_7 也不能移动。出现该死锁的原因是：某些报文的头切片没有到达虚通道头部，使得它们不能传输到“无死锁”路径；同时，这些报文的尾切片占用了其它虚通道的头部，阻止了后续报文的传输，导致报文之间出现循环阻塞。比如说在图4.3中， P_0 的尾切片位于 AVC_0 中，其阻止 P_3 发送到 AVC_0 中。而 P_3 又通过循环关系阻止了 P_0 的发送。

如果报文长度大于虚通道深度，允许多个报文的切片存储于同一条虚通道中很容易带来死锁，这是因为当报文进入到一条非空虚通道时，其尾切片依然会占

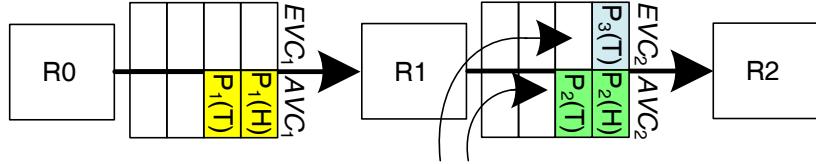
根据保守分配策略，报文 P_1 不能使用这些空闲缓存

图 4.4 保守分配策略降低了虚通道利用率

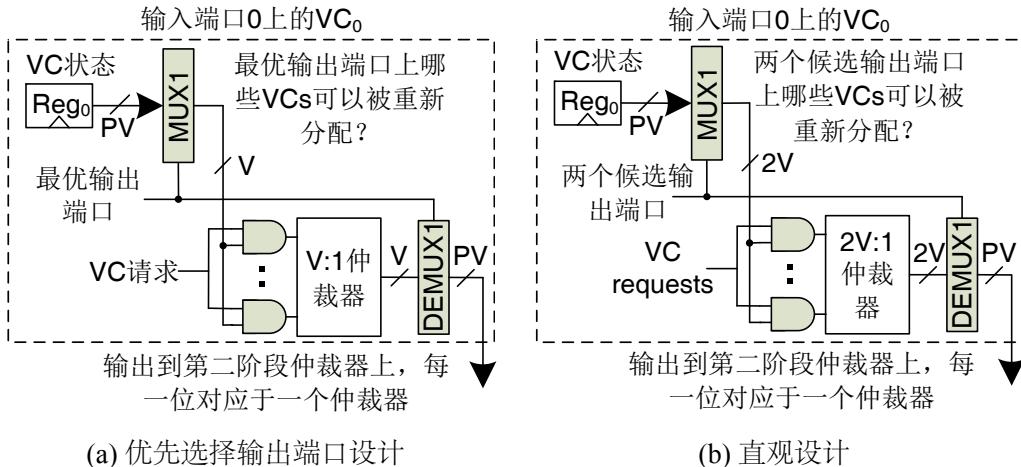


图 4.5 虚通道分配器的第一级仲裁器结构

用另一条虚通道的头部。然而，丰富的连线资源导致片上网络传输的大部分是短报文，此时要求每条虚通道只能存储同一报文的切片会降低虚通道的利用率，从而带来性能损失。如图4.4所示， EVC_2 和 AVC_2 不能被重新分配，报文 P_1 只能在 AVC_1 等待，直到 EVC_2 或 AVC_2 中的切片被完全发送出去。由于 P_1 只包含两个切片， EVC_2 和 AVC_2 的空闲缓存都能存储整个报文，此时将 P_1 发送到这两条虚通道中不会阻止后续报文到达 AVC_1 的头部。也就是说，这里存在性能优化的可能性，第4.4.1节证明此时 P_1 可以发送到 EVC_2 或 AVC_2 中，而不会带来死锁。

4.3.2 路由灵活性

本节分析虚通道受限网络中的路由灵活性。许多基于 Duato 理论的完全自适应路由算法由两部分组成：路由函数和输出端口选择策略^[50, 63, 148, 210]。路由函数计算所有可能的候选输出端口，选择策略从它们中间选取一个。如果选择策略选择了某个输出端口，报文只能申请该端口的虚通道，本章将这种类型的路由算法称为优先选择输出端口设计。对于由两级仲裁器组成的可分离虚通道分配器，优先选择输出端口设计在第一级结构上只需使用 $V:1$ 的仲裁器，如图4.5(a)所示。

优先选择输出端口设计要求一旦报文进入逃逸虚通道，其后续路由只能使用逃逸虚通道，该报文将丧失路由灵活性。违反这个限制可能导致如图4.6所示的死

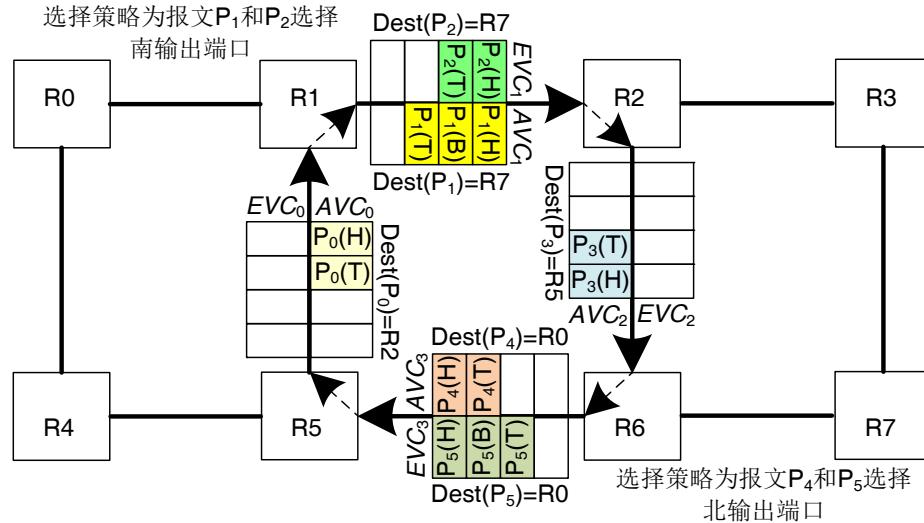


图 4.6 优先选择输出端口设计中的死锁

锁。在这个例子中，报文 P_1 和 P_2 可以使用南端口或东端口到达它们的目标节点路由器 $R7$ 。如果选择策略选择了南端口， P_1 和 P_2 只能请求 AVC_2 ；逃逸虚通道只有在输出端口符合维序路由算法时才能被使用，它们不能请求 EVC_2 。类似的，如果选择策略为 P_4 和 P_5 选择了北端口，它们只能请求 AVC_0 。此时没有报文可以移动，网络中出现死锁。因此，在优先选择输出端口设计中，有必要要求报文在进入逃逸虚通道后只能继续使用逃逸虚通道。然而，报文在虚通道数受限网络中有较大可能性进入逃逸虚通道，这个要求会带来性能损失。

由于 Duato 理论的逃逸虚通道依赖图无圈^[162, 199]，当报文申请到逃逸虚通道后，总能找到一条没有循环依赖的传输路径^[162, 199]。因此，如果报文任何时候都能申请逃逸虚通道，Duato 理论支持报文在进入逃逸虚通道后重新回到适应性虚通道上^[162, 199]。满足这一条件的直观设计是让报文申请所有候选输出端口^[162, 199]，这些输出端口中至少有一个是符合维序路由算法的，报文可以申请该端口的逃逸虚通道。但是这种直观设计会带来额外的硬件开销，如图4.5(b)所示，虚通道分配器第一级结构必须采用 $2V : 1$ 的仲裁器。基于这些观察，本章提出了一种在较低硬件开销条件下能维持路由灵活性的设计。

4.4 基于全报文发送策略的完全自适应路由算法

本节首先描述全报文发送策略，并证明其无死锁特性，然后给出一个在较低硬件开销下能维持路由灵活性的设计，最后讨论硬件实现和开销。

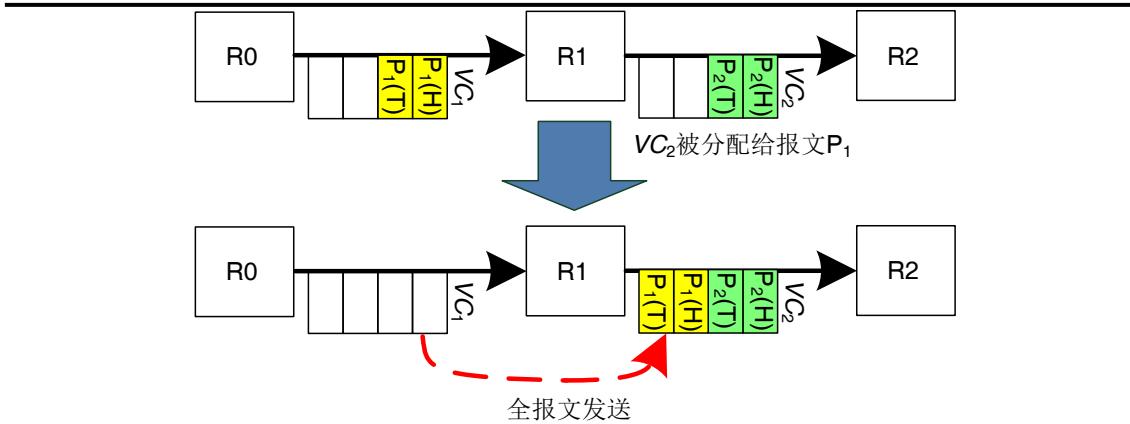


图 4.7 全报文发送实例

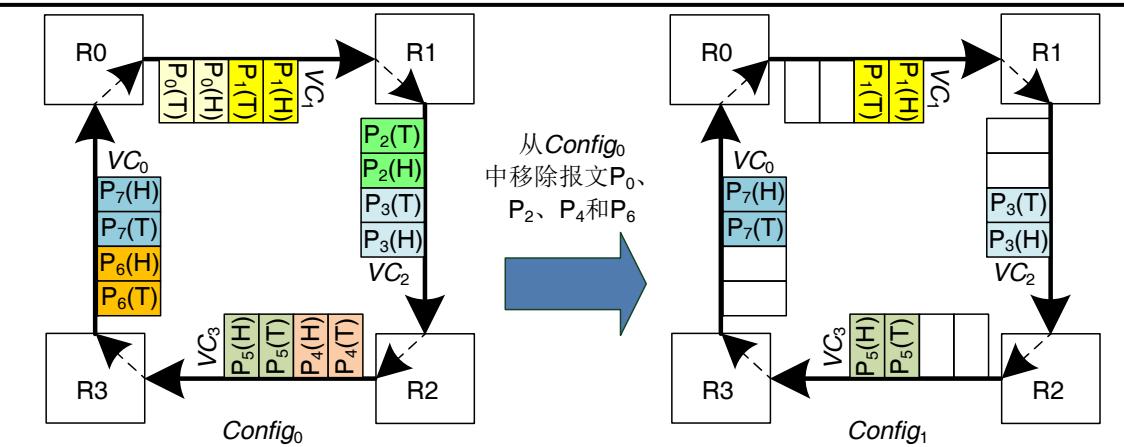
4.4.1 全报文发送

如第4.3.1节所分析，保守虚通道分配策略降低了虚通道利用率，影响了路由算法的性能。本节提出了一种允许非空虚通道被重新分配的策略，该策略能提高虚通道利用率，同时也不会带来死锁。假设报文 P_k 的长度是 $length(P_k)$ ，其当前位于虚通道 VC_i 中， VC_j 是 VC_i 的下游虚通道，假设路由算法允许 P_k 使用 VC_j 。在保守分配策略下， VC_j 只有在为空时，才能分配给 P_k 使用。在所提出的策略下，只要 VC_j 已经接收到最近分配的报文的尾切片，并且剩余空闲缓存数量 ($free_slots(VC_j)$) 大于或等于 P_k 的长度， VC_j 可以分配给 P_k 使用。由于 $free_slots(VC_j) \geq length(P_k)$ ， P_k 的所有切片在有限的时间内一定会全部发送到 VC_j 中，这种分配策略称为全报文发送 (Whole Packet Forwarding, WPF)。

图4.7给出了全报文发送的一个实例，假设路由算法允许报文 P_1 使用虚通道 VC_2 ， VC_2 已经接收到了其最近分配的报文 P_2 的尾切片。 VC_2 的 2 个空闲缓存单元足够容纳 P_1 的 2 个切片，因此，全报文发送策略允许 VC_2 分配给 P_1 使用。当非空虚通道的空闲缓存足够存储整个报文时，全报文发送允许该虚通道被重新分配；从这个角度说，全报文发送与基于报文的存储转发^[7]或虚切通流控机制比较类似^[164]。然而，空虚通道依然采用虫孔交换机制；也就是说，全报文发送并没有要求虚通道深度一定要大于最长报文的切片数。全报文发送可以被视为在虫孔交换网络中采用基于报文的流控机制，这种混合设计解决了完全自适应路由算法必须采用保守分配策略的局限性。

本章的论点是：如果路由算法采用保守虚通道分配策略没有死锁，则采用全报文发送策略向非空虚通道发送报文也不会带来死锁。为方便起见，我们将使用保守分配策略的路由算法记为 Alg ，将在 Alg 基础上使用全报文发送策略的路由算法记为 $Alg + WPF$ 。

定理 4.1：如果 Alg 是无死锁的，则 $Alg + WPF$ 也是无死锁的。

图 4.8 基于 $Config_0$ 构造一个 $Alg + WPF$ 的网络配置

证明思路：采用反证法，我们证明如果 $Alg + WPF$ 存在一个死锁配置，则 Alg 也存在一个死锁配置。以图4.8中的死锁配置 $Config_0$ 为例，我们将那些头切片不处于虚通道头部的报文移除，得到一个新的配置 $Config_1$ 。我们证明 Alg 路由算法可以产生 $Config_1$ ，同时 $Config_1$ 是一种死锁配置，而 Alg 是无死锁的，因此不存在这样的死锁配置。

证明：反证法。如果 $Alg + WPF$ 不是无死锁的，则其存在一个死锁配置 ($Config_0$)，该配置中的报文集合被记为 P_{set_0} 。根据死锁配置的定义， P_{set_0} 中的报文所等待的虚通道当前都被 P_{set_0} 中的其它报文占用^[198]。我们证明 Alg 也存在一个死锁配置。证明包括三步：

步骤 1：我们在 $Config_0$ 的基础上构造一个新的网络配置。考察 $Config_0$ 的报文 P_i ，如果 P_i 的头切片没有位于虚通道头部，则该虚通道是按照全报文发送分配给 P_i 的，因此，报文 P_i 的所有切片都位于一条虚通道内。我们将报文 P_i 从 $Config_0$ 中移除，这些被移除报文的集合被记为 P_{subset_0} ，所得到的新配置被记为 $Config_1$ ，处于 $Config_1$ 中的剩余报文集合被记为 P_{subset_1} 。

步骤 2：我们证明如果网络采用 Alg 进行路由，所有 P_{subset_1} 中的报文能被发送到它们在配置 $Config_1$ 中的虚通道上。考查 P_{subset_1} 中的每一个报文 P_j ，我们进一步考查当网络采用 $Alg + WPF$ 路由时，报文 P_j 的每一跳 hop_k 。不失一般性，假设报文 P_j 的头切片在第 hop_k 跳从虚通道 VC_k 发送到虚通道 VC_{k+1} 上，此时 VC_{k+1} 有两种情况。

2.1). 当报文 P_j 的头切片发送时， VC_{k+1} 为空，即 VC_{k+1} 是按照保守策略分配给 P_j 使用的。因此，如果网络采用 Alg 路由，报文 P_j 仍然可以使用 VC_{k+1} 。

2.2). 当报文 P_j 的头切片发送时， VC_{k+1} 非空，即 VC_{k+1} 是按照全报文发送分配给 P_j 使用的。 P_j 可以发送到 VC_{k+1} 上，表示路由算法允许 P_j 使用 VC_{k+1} 。然而，当网络采用 Alg 路由时， P_j 只有在 VC_{k+1} 为空时才能发送。由于 Alg 是无死

锁的，当前处于虚通道 VC_{k+1} 中的报文一定会在有限的时间内被发送出去。然后， VC_{k+1} 可以按照保守策略分配给报文 P_j 使用。因此，如果网络被 Alg 路由， P_j 依然可以使用 VC_{k+1} 。

综合 2.1) 和 2.2)，当网络采用 $Alg + WPF$ 路由时，如果报文 P_j 使用了某条虚通道，则该虚通道在采用 Alg 路由时依然可以被报文 P_j 使用。因此，按照 Alg 路由算法， P_{subset_1} 中的报文都可以被发送到它们在 $Config_1$ 中的虚通道上。

步骤 3：我们证明 $Config_1$ 是 Alg 路由算法的一种死锁配置。考虑被移除报文集合 P_{subset_0} 中每一个报文 P_i ， P_i 的所有切片都位于同一虚通道内，同时 P_i 的头切片不处于该虚通道头部，因此将 P_i 从配置 $Config_0$ 中移除并不会产生空虚通道。此时，每条虚通道中只有属于同一报文的切片。 Alg 所使用保守分配策略只允许空虚通道被重新分配，因此，剩余报文集合 P_{subset_1} 中的报文依然在等待那些被同一集合内其它报文占用的虚通道。也就是说， $Config_1$ 是 Alg 的一个死锁配置。但是根据假设， Alg 是无死锁的，因此不存在这样的配置。因此， $Alg + WPF$ 也是无死锁的。 ■

上述证明中并没有对路由算法做出任何假设，因此，全报文发送可以被应用到任何路由算法上。全报文发送允许非空虚通道被重新分配这一特性是对已有的死锁避免理论的重要扩展。

4.4.2 完全自适应路由算法

如图4.2所示，完全自适应路由算法在虚通道受限网络中的性能可能低于确定性和部分自适应路由算法。为了解决这个问题，本节采用全报文发送策略设计了一种完全自适应路由算法，该设计基于 Duato 理论^[162, 199] 进行死锁避免。在虚通道数受限的环境下，路由算法应该尽量维持路由的灵活性，即路由算法应允许报文在进入逃逸虚通道后，将来依然可以使用适应性虚通道。否则，一旦报文进入逃逸虚通道，其后续传输将丧失路由灵活性。如第4.3.2节的分析，为了实现这个目标，设计应保证报文在任何时候都能申请逃逸虚通道^[162, 199]。

本节设计对优先选择输出端口设计进行了修改。在优先选择输出端口设计中，当选择策略获得的输出端口不符合维序路由时，报文不能申请逃逸虚通道，此时，本节设计允许报文申请没有被选择的端口上的逃逸虚通道。以图4.6中报文 P_1 为例，当选择策略选择南端口时，本节设计允许 P_1 申请东端口的逃逸虚通道。这个设计使得报文在任何时候都能申请逃逸虚通道，因此，它支持报文在使用逃逸虚通道后返回适应性虚通道。该设计的虚通道分配器第一级结构采用 $V : 1$ 仲裁器，与图4.5(b)的直观设计相比，获得了更低的硬件开销和关键路径延迟。

4.4.3 路由器微结构

传统片上网络路由器流水线包括 4 个阶段：路由计算 (RC)、虚通道分配 (VA)、交叉开关分配 (SA) 和交叉开关传输 (ST)^[7, 38, 39, 163]。传统片上网络路由器可以采用一些优化技术以提高基准性能：猜测交叉开关分配在较低负载时能并行执行 VA 和 SA^[38]；超前路由计算将 RC 段从关键路径移除，路由算法在前一跳计算本级路由器的两个候选输出端口，本级路由器完成输出端口选择^[60, 63, 148]。基准路由器的流水线延迟为 2 个时钟周期，链路传输还需要 1 个时钟周期。

全报文发送和第4.4.2节提出的路由算法都只需对虚通道分配器进行少量修改。它们可以使用任何类型分配器，本章采用可分离虚通道分配器^[38, 39, 44]，这种分配器结构简单，支持更高频率。可分离虚通道分配器的第一级仲裁器完成同一输入虚通道申请的多条输出虚通道之间的仲裁，第二级仲裁器完成申请同一输出虚通道的多条输入虚通道之间的仲裁，设计只需修改第一级仲裁器。

首先需要知道下游虚通道是否满足全报文发送条件，判断标准是下游虚通道已经接收到最近分配报文的尾切片，同时其空闲缓存足够存储整个新的报文。Cache 一致性协议只包含两种长度的报文，长报文长度一般会大于虚通道深度，本章只考虑在短报文（单切片报文）上采用全报文发送。因此，如果下游虚通道已经接收到上一个报文的尾切片，并且当前还有空闲缓存资源，则允许该虚通道被重新分配给一个单切片报文。

图4.9给出了本章设计的虚通道分配器结构，寄存器 Reg_0 记录下游虚通道是否可以按照保守策略分配，其条件是下游虚通道当前处于空状态。寄存器 Reg_1 记录下游虚通道是否可以按照全报文发送策略分配。根据输入报文的类型，选择器 MUX0 选择 Reg_0 或 Reg_1 的值输出。如果输入报文是单切片报文，则采用全报文发送，MUX0 选择 Reg_1 的值，否则，选择 Reg_0 的值。上游路由器采用信元记录下游路由器的状态^[163]，因此 Reg_0 和 Reg_1 的更新不处于关键路径上，全报文发送增加的额外延迟是一个两输入选择器：MUX0。

第4.4.2节提出的路由算法需要修改 MUX1 和 DEMUX1 单元。如图4.9所示，MUX1 增加了两个输入信号： DOR 和 *the other output port*， DOR 表示当前所选择的输出端口是否符合维序路由，*the other output port* 记录没有被选择的候选输出端口编号，这两个信号由路由计算逻辑得到。如果 DOR 值为 0，则选择的输出端口不符合维序路由，此时，*the other output port* 端口的逃逸虚通道的状态被发送到 $V : 1$ 仲裁器，这个功能通过一个选择信号是 DOR 的两路选择器完成。类似的，如果 DOR 为 0，第一级 $V : 1$ 仲裁器的结果被发送到 *the other output port* 逃逸虚通道上的第二级仲裁器，一个两输出的多路分配器用于实现这个功能。所提出的

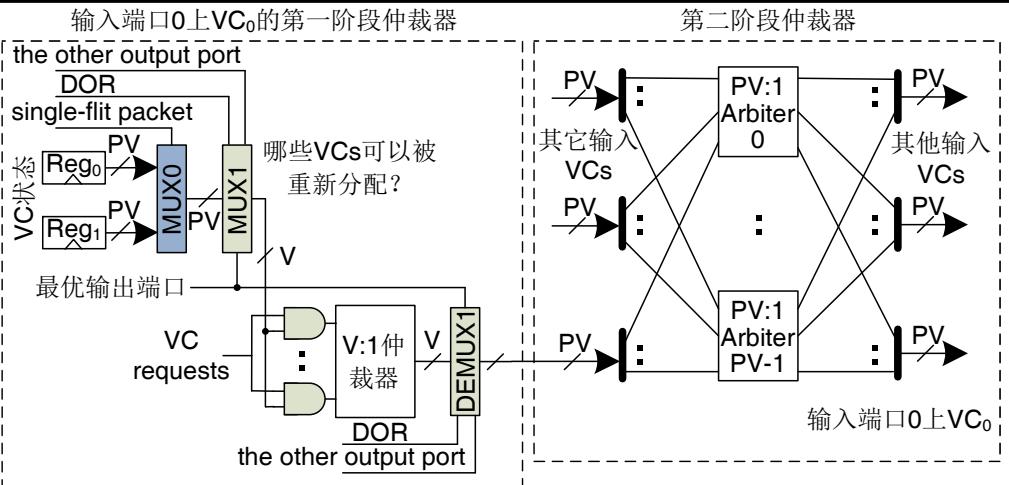


图 4.9 所提出的虚通道分配器结构图

表 4.2 虚通道分配器关键路径延迟和面积开销

| 设计类型 | 延迟 (ns) | 面积 (μm^2) |
|----------------------|---------|------------------------|
| 优先选择输出端口设计 (图4.5(a)) | 1.78 | 49437.4 |
| 直观设计 (图4.5(b)) | 1.92 | 56045.4 |
| 本章提出的设计 (图4.9) | 1.79 | 49512.6 |

路由算法的额外延迟是一个两输入的多路选择器和一个两输出多路分配器。

为评测硬件开销，本节在一个开源 RTL Verilog 片上网络路由器^[44] 上实现了图4.5和图4.9中的三种虚通道分配器，并采用 Synopsys Design Compiler 进行综合，综合使用 TSMC 65nm 工艺，综合目标设置为普通环境 (1.2V, 25°C) 达到 500 MHZ。分配器中使用轮转仲裁器^[163]，每个路由器有 5 个输入端口，支持 4 个虚拟网络，每个虚拟网络上包含 2 条虚通道。表4.2给出了关键路径延迟和面积结果，直观设计 (图4.5(b)) 的第一级结构使用 4:1 仲裁器，导致其关键路径延迟和面积比优先选择端口设计分别高 7.9% 和 13.4%；本章所提出的设计在第一级结构上依然采用 2:1 仲裁器，因此其关键路径延迟和面积只增加了 0.5% 和 0.2%。

4.5 实验评估

实验对时钟精确模拟器 Booksim^[163] 进行修改，以模拟第4.4.3节中描述的路由器流水线和微结构，实验评估了所设计的路由算法采用保守策略 (FULLY) 和全报文发送策略 (FULLY+WPF) 时的性能，同时也评估了优先选择输出端口设计采用保守策略 (PSF) 和全报文发送策略 (PSF+WPF) 时的性能。实验比较对象包括确定性路由算法和部分自适应路由算法，所采用的确定性路由算法是维序路由 (DOR)，部分自适应路由算法包括西方向优先 (West-first)、负方向优先

表 4.3 实验配置参数

| 属性 | 基准 | 变量 |
|-----------------|-------------------|-------------------|
| 网络拓扑 | 4×4 mesh | 8×8 mesh |
| VCs/VN | 2 | 4 |
| VC 深度 | 4 | 3, 2 |
| 报文切片数 | 长: 5, 短: 1 | - |
| SFP 比例 | 80% | 60%, 40% |
| Warmup 时钟, 测量时钟 | 10000, 100000 | - |

表 4.4 全系统模拟参数

| 参数名 | 参数值 |
|------------------|--|
| 核数 | 16 |
| 一级 cache (数据或指令) | private, 4-way, 32KB each |
| 二级 cache | private, 8-way, 512KB each |
| Cache 一致性协议 | 分布式目录 MOESI 协议 |
| 网络拓扑 | 4×4 Mesh, 4 个虚拟网络 (VN), 2VCs/VN |

(Negative-first) 和奇偶模型 (Odd-even)。所有自适应路由算法中都采用了一种局部自适应输出端口选择策略：当存在两个候选输出端口时，报文选择具有较多空闲缓存的端口。

实验过程既使用了合成流量模式，又使用了真实应用程序。虚拟网络之间是相互独立的，因此，合成流量模式实验只配置一个虚拟网络。基准实验配置是： 4×4 mesh 网络，每个虚拟网络配置 2 条虚通道，每条虚通道有 4 个缓存单元，网络中包含单切片报文和 5 切片报文，基准单切片报文 (Single Flit Packet, SFP) 比例是 80%。表4.3给出了基准网络配置和用于敏感性分析的参数变化值。

实验使用两个模拟器评估全系统性能：FeS2^[185] 模拟 x86 体系结构，Booksim 模拟 FeS2 产生的报文在片上网络中的传输。实验模拟了一个 16 核，拓扑结构是 4×4 mesh 的 CMP 平台，并评估了 PARSEC 测试集^[201] 在 16 线程运行时的性能。实验假设处理器频率是网络频率的 5 倍。实验使用了分布式目录 MOESI 协议，其需要 4 个虚拟网络以避免协议层死锁。Cache 行包括 64 字节，网络切片大小是 16 字节。PARSEC 应用程序采用 *simsmall* 输入集，全系统性能指标是程序运行时间，表4.4给出了全系统模拟的配置参数。

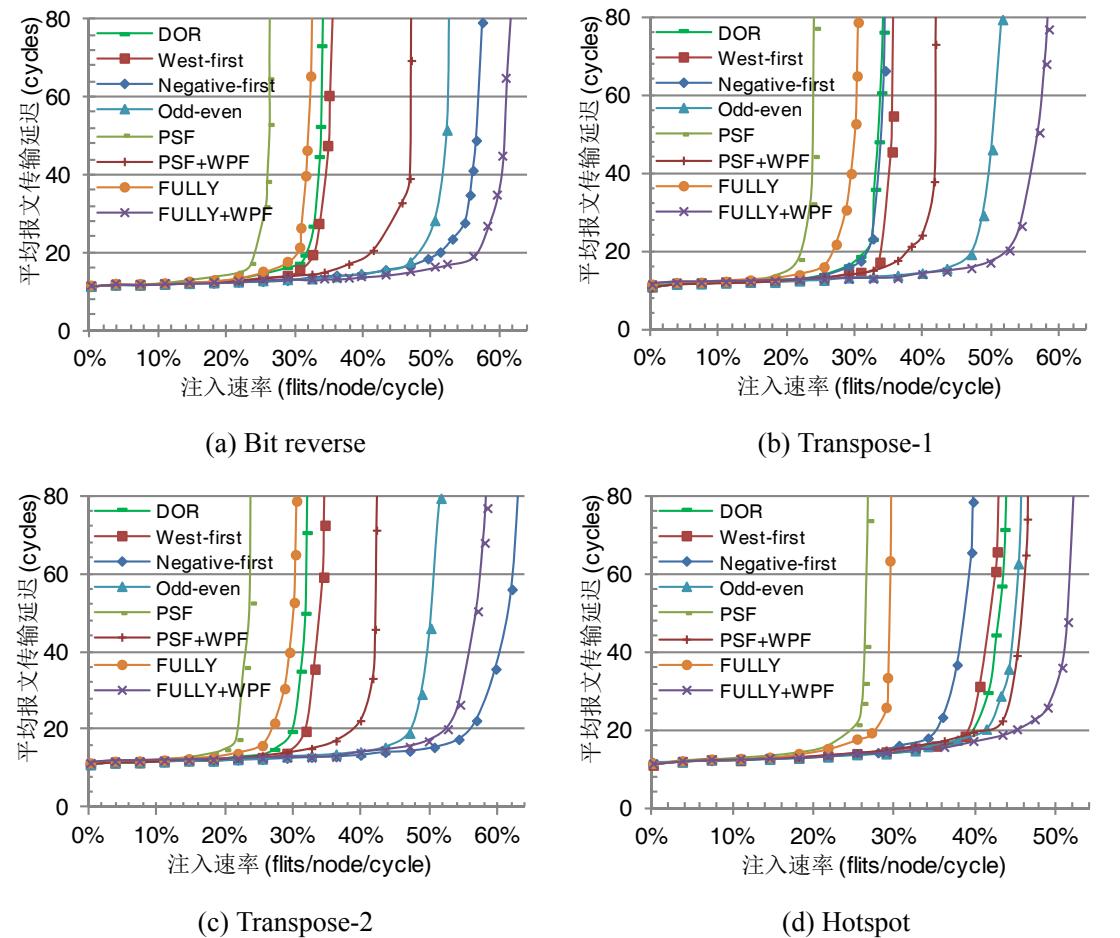


图 4.10 路由算法在基准网络配置下的性能

4.5.1 合成流量模式结果

图4.10给出了路由算法在基准网络配置中的性能，评估使用了4种合成流量模式：bit reverse、hotspot 和 2 种 transpose 模式。尽管 PSF 和 FULLY 能为所有的报文提供适应性，它们的性能受限于所采用的保守虚通道分配策略，因此，在这4种流量模式下，PSF 和 FULLY 的性能最差。在 PSF 算法中，一旦报文进入逃逸虚通道，其后续路由过程只能按照维序路由通过逃逸虚通道传输，这种较低路由灵活性进一步制约了它的性能，导致 PSF 性能低于 FULLY。

在 bit reverse 模式下，位地址为 $\{S_3, S_2, S_1, S_0\}$ 的源节点向 $\{S_0, S_1, S_2, S_3\}$ 节点发送报文；62.5% 的报文是在东北和西南象限之间传输的，negative-first 为这些报文提供了适应性；只有 37.5% 的报文是向东发送的，west-first 为这些报文提供了适应性。Negative-first 提供的适应性高于 west-first，因此，其性能优于 west-first。类似的，odd-even 提供的适应性低于 negative-first，导致其性能也低于 negative-first。尽管 PSF+WPF 采用 WPF 策略提升了虚通道利用率，PSF+WPF 较

低的路由灵活性导致其性能依然低于 odd-even 和 negative-first。FULLY+WPF 既取得了较高的虚通道利用率，又具有较大的灵活性，因此它的性能最好。

在 transpose-1 模式下，节点 (i, j) 向节点 $(3-i, 3-j)$ 发送报文，此时 negative-first 退化成维序路由，它不能为该模式的报文提供任何适应性。West-first 依然能为 37.5% 的报文提供适应性，因此 west-first 的性能优于 negative-first。Odd-even 提供的适应性高于前两种部分自适应算法，它的性能也优于 negative-first 和 west-first。FULLY+WPF 为所有的报文都提供了适应性，它的饱和吞吐率比 odd-even 高 15.7%。

Transpose-2 是一种对 negative-first 友好的流量模式，在该模式下，节点 (i, j) 向节点 (j, i) 发送报文，negative-first 能够为这种模式的所有报文提供适应性，因此获得了最好的性能。尽管 FULLY+WPF 也能为所有报文提供适应性，它的性能受限于逃逸虚通道的使用限制：只有当输出端口符合维序路由时，才能使用逃逸虚通道。Transpose-1 和 transpose-2 是两种对称的流量模式，FULLY+WPF 和 odd-even 为它们提供相同的适应性，因此，它们在这两种模式下的性能基本类似。

Hotspot 流量模式将网络中的 4 个节点设置成热点，它们比其它节点多接收 20% 的报文，这种流量模式模拟由存储控制器造成的非均衡通信。由于 FULLY+WPF 和 odd-even 为该模式提供了更高适应性，它们的性能最好。Odd-even 的适应性低于 FULLY+WPF，它的性能也低于 FULLY+WPF。Hotspot 模式的背景通信是 uniform random 模式，DOR 能够将 uniform random 模式的通信更为均衡地分布，因此，DOR 的性能优于 negative-first 和 west-first。

由上述实验结果可以看出，保守虚通道分配策略严重制约了完全自适应路由算法的性能，使得它们的性能在虚通道数受限环境下低于确定性路由算法和部分自适应路由算法。Negative-first 和 west-first 的适应性在不同流量模式下不均衡，比如说，negative-first 能够为 transpose-2 模式的所有报文提供适应性，但是其不能为 transpose-1 模式的任何报文提供适应性。片上网络的流量模式在运行时可能发生改变，同时不同虚拟网络的流量模式也可能不相同，使得这些部分自适应算法不适合这样的场景。Odd-even 为不同模式提供了均衡的适应性。

WPF 策略提高了完全自适应算法的虚通道利用率，因此提升了它们的性能。但是当虚通道数受限时，获得较高的性能还需要提供足够的路由灵活性，路由算法应允许报文在使用逃逸虚通道后仍然可以使用适应性虚通道。PSF+WPF 的路由灵活性较低，导致其性能低于某些部分自适应算法。FULLY+WPF 既获得了较高的虚通道利用率，同时也具有较好的路由灵活性，因此，其性能最好。表4.5给出了 FULLY+WPF 相对于其它路由算法在 4 种流量模式下的平均饱和吞吐率提升。FULLY+WPF 的性能比 FULLY 高 88.9%，这是由 WPF 策略带来的性能提升；

表 4.5 FULLY+WPF 的平均饱和吞吐率提升

| 算法 | 性能提升 | 算法 | 性能提升 |
|----------------|-------|----------|--------|
| FULLY | 88.9% | Odd-even | 16.3% |
| DOR | 64.5% | PSF | 130.9% |
| West-first | 58.6% | PSF+WPF | 31.3% |
| Negative-first | 26.6% | - | - |

FULLY+WPF 和 PSF+WPF 的性能差异反应了路由灵活性的影响，较高的路由灵活性带来了 31.3% 的性能提升。

4.5.2 PARSEC 测试集结果

图4.11给出了各种路由算法在 PARSEC 程序下相对于 PSF 的全系统性能加速比。PARSEC 测试集包括两种类型的应用程序：低网络负载应用程序和高网络负载应用程序。优化设计的路由算法能够提高网络的饱和吞吐率，但是其全系统性能取决于每个应用程序产生的负载和流量模式。具有较高网络负载和较多猝发流量的应用程序能够从优化的路由算法设计上获得性能提升。不同路由算法在 blackscholes、fluidanimate、raytrace 和 swaptions 上的性能基本类似，这些应用程序的工作集能够载入私有二级 cache，降低了网络负载，导致支持高网络吞吐率的设计并不能提升它们的性能。其它 6 个应用程序具有大量的猝发性流量，增强了网络层优化对系统性能的影响，支持高网络吞吐率的路由算法能给它们带来性能提升。比如说，FULLY+WPF 在 facesim 和 streamcluster 下分别获得 48.5% 和 43.0% 的加速比。vips 有大量东方向通信，因此，west-first 在该程序下的性能最好。由于 negative-first 为 facesim 和 streamcluster 的猝发性通信提供更高的适应性，其性能在这两个应用程序中比 odd-even 好。在除 vips 之外的高网络负载程序中，FULLY+WPF 的性能最好。在这些程序中，FULLY+WPF 的平均加速比是 21.3%，最大加速比是 37.8%。FULLY+WPF 的路由灵活性高于 PSF+WPF，因此 FULLY+WPF 的性能平均比 PSF+WPF 高 12.1%。与 PSF、DOR、west-first、negative-first 和 odd-even 算法相比，FULLY+WPF 分别获得 29.3%、15.0%、10.1%、9.9% 和 10.4% 的性能提升。

4.6 敏感性分析

不同系统的具体网络配置可能与表4.3中的基准配置不同，本节对网络配置参数进行敏感性分析，在每个小节中，除了被分析的参数外，其它参数都采用基准配置中的值。

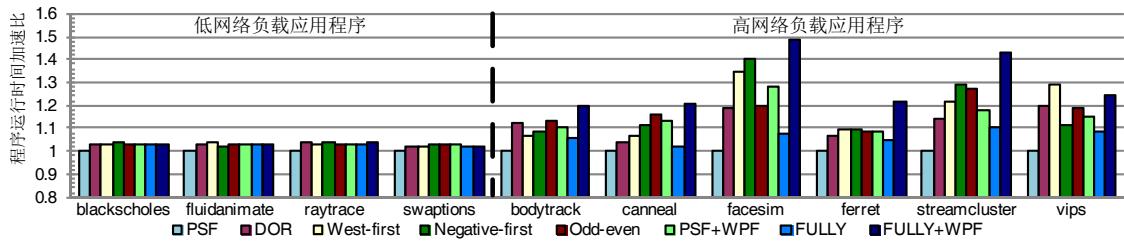


图 4.11 PARSEC 测试集的全系统加速比

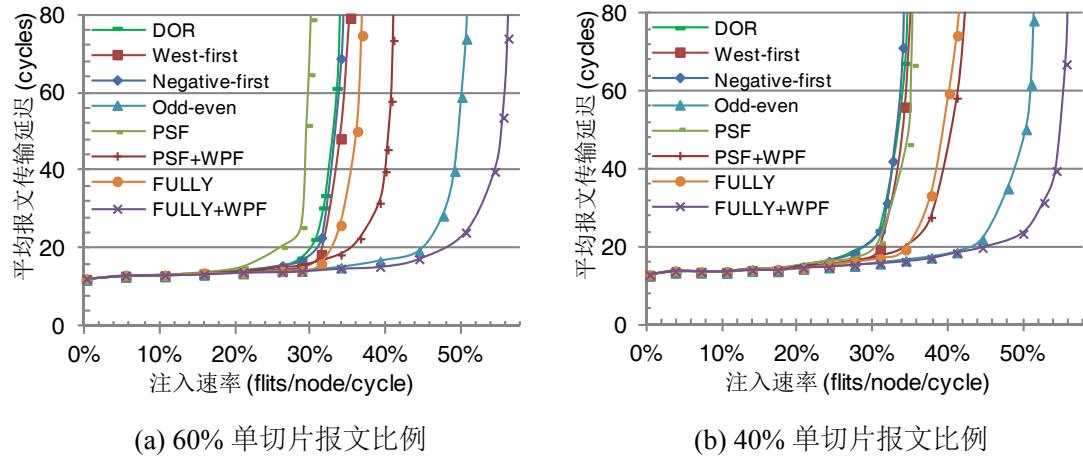


图 4.12 路由算法在不同单切片报文比例下的性能

4.6.1 单切片报文比例

单切片报文比例取决于应用程序特性、cache 层次结构和 cache 一致性协议。为了测试设计的鲁棒性，图4.12评估了 60% 和 40% 单切片报文比例时 transpose-1 模式的性能。DOR、west-first、negative-first 和 odd-even 采用的积极虚通道分配策略对报文长度不敏感，因此，这些算法的性能在不同单切片报文比例时基本相同。由于长报文在保守虚通道分配策略下能提高虚通道利用率，PSF 和 FULLY 的性能随着单切片报文比例的下降而上升。使用全报文发送的概率在越低的单切片报文比例下越少，这使得 FULLY+WPF 和 FULLY (或 PSF+WPF 和 PSF) 的性能差异随着单切片报文比例的下降而减少。但是即使在 40% 单切片报文比例下，FULLY+WPF 的性能依然比 FULLY 高 53.1%。在各种单切片报文比例下，FULLY+WPF 都获得了最好的性能。

4.6.2 虚通道深度

不同片上网络实现面临不同的面积和功耗限制，它们的虚通道 (VC) 深度可能会发生变化。为测试设计的灵活性，图4.13评估了每条 VC 包含 3 个缓存单元 (buffer slots) 和 2 个缓存单元时的性能。当 VC 深度从 4 个 slots (图4.10(a)) 下

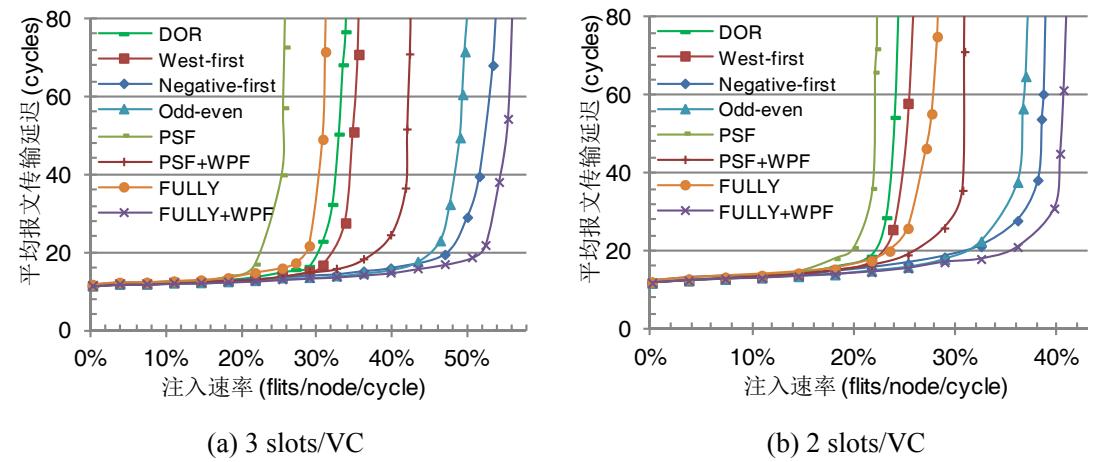


图 4.13 路由算法在不同虚通道深度下的性能

降到 3 个 slots (图4.13(a)) 时, DOR 和 west-first 的性能基本保持不变, FULLY 和 PSF 的性能出现轻微下降。DOR 和 west-first 的性能瓶颈是它们有限的适应性, 因此, 当 VC 深度从 4 个 slots 下降到 3 个 slots 时, 它们的性能几乎没有变化; FULLY 和 PSF 的性能瓶颈是所采用的保守虚通道分配策略, 由于网络 80% 报文是单切片报文, VC 深度从 4 个 slots 下降到 3 个 slots 只给它们的性能带来轻微的影响。FULLY+WPF、PSF+WPF、odd-even 和 negative-first 的性能随着 VC 深度的减少而下降。对这些设计而言, VC 深度是它们性能瓶颈, 较浅的虚通道提高了阻塞报文分布的路由器数目, 从而增加了链式阻塞对性能的影响^[214]。

当每条 VC 配置 2 个 slots 时, 所有算法的性能都出现了下降, FULLY 此时的性能优于 DOR 和 west-first。保守分配策略和积极分配策略的差异随着虚通道深度变浅而减少, 因此, FULLY 相比于 DOR 和 west-first 获得了一定性能提升。网络中存在的大量短报文使得 WPF 策略在 2 slots/VC 时依然带来了很大的性能提升, FULLY+WPF 的性能比 FULLY 高 46.2%。FULLY+WPF 的饱和吞吐率在 2 slots/VC 时是 40.3% (图4.13(b)), 而 FULLY 的饱和吞吐率在 4 slots/VC 时 (图4.10(a)) 是 32.3%; 也就是说, WPF 策略在使用保守分配策略一半的缓存资源时获得了高于保守分配策略的性能。类似的, PSF+WPF 在 2 slots/VC 时的性能高于 PSF 在 4 slots/VC 时的性能。

4.6.3 虚通道数目

随着工艺的进一步发展, 每个虚拟网络可能配置更多的虚通道 (VC)。此外, 优化 cache 一致性协议可以减少所需虚拟网络数目, 从而允许每个虚拟网络配置更多虚通道。DOR、west-first 和 odd-even 在 2 条 VCs (图4.10(a)) 和 4 条 VCs 时 (图4.14(a)) 的性能基本相同。这些算法的性能在 VC 数目从 1 条增加到

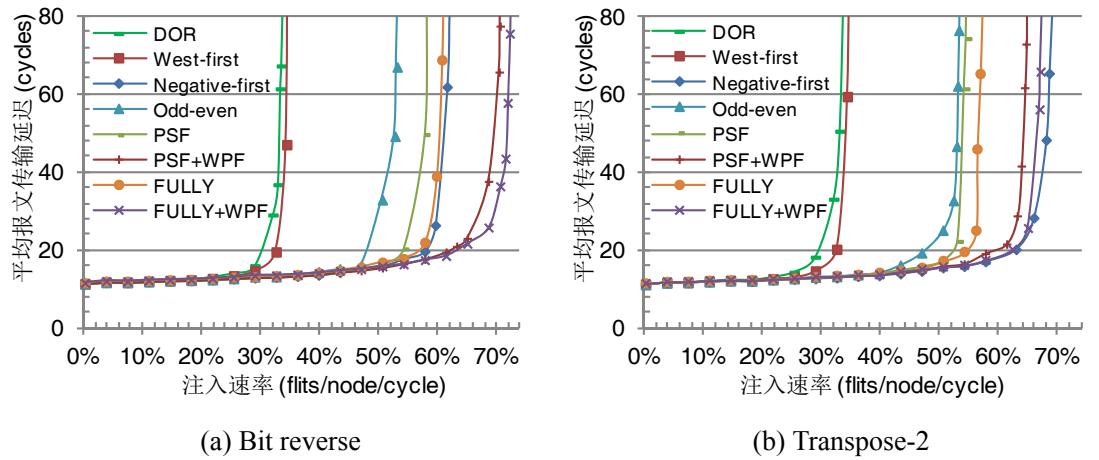


图 4.14 路由算法在 4 条虚通道配置下的性能

2 条时获得了较大提升，但是当 VC 数目继续增加时，它们的性能受限于有限的物理路径多样性，因此没有进一步提升。Negative-first 具有较高的物理通道多样性，当 VC 数目从 2 条增加 4 条时，其获得了一定程度的性能提升。由于更多的虚通道降低了保守分配策略对性能的影响，当 VC 数目从 2 条增加 4 条时，PSF、FULLY、PSF+WPF 和 FULLY+WPF 都得到了显著性能提升。报文进入逃逸虚通道的概率随着虚通道数的增加而下降，因此，PSF 和 FULLY（或 PSF+WPF 和 FULLY+WPF）的性能差异随着 VC 数目的增加而减少。

图4.14(b)给出了 transpose-2 模式的性能，negative-first 在该模式的性能最好。随着虚通道数目的增加，报文使用逃逸虚通道的概率逐渐下降，因此，FULLY+WPF 在配置 4 条 VCs 时的性能与 negative-first 相当。由于报文在配置更多虚通道时找到空虚通道的概率会上升，FULLY 和 FULLY+WPF（或 PSF 和 PSF+WPF）的性能差异随着虚通道数目的增加而下降。WPF 策略向非空虚通道发送报文，这会带来头报文阻塞^[163]。尽管如此，在图4.14的两种模式中，FULLY+WPF 依然获得了平均 19.8% 的饱和吞吐率提升。与虚通道深度分析类似，FULLY+WPF 在采用 2 条 VCs 时的性能（图4.10(a)和图4.10(c)）与 FULLY 采用 4 条 VCs 时（图4.14(a)和图4.14(b)）的性能相当；WPF 使用保守分配策略一半虚通道数时获得了与保守分配策略类似的性能。

4.6.4 网络规模

图4.15研究了路由算法在 8×8 mesh 上的可扩展性。网络通信主要由流量模式决定，而与网络规模关系不大，因此，不同路由算法在 8×8 mesh 中的性能趋势与 4×4 mesh 中（图4.10）基本相同。由于流量模式的平均跳数随着网络规模的增大而提高，大规模网络中的虚通道面临更高的压力。WPF 策略能提高虚通

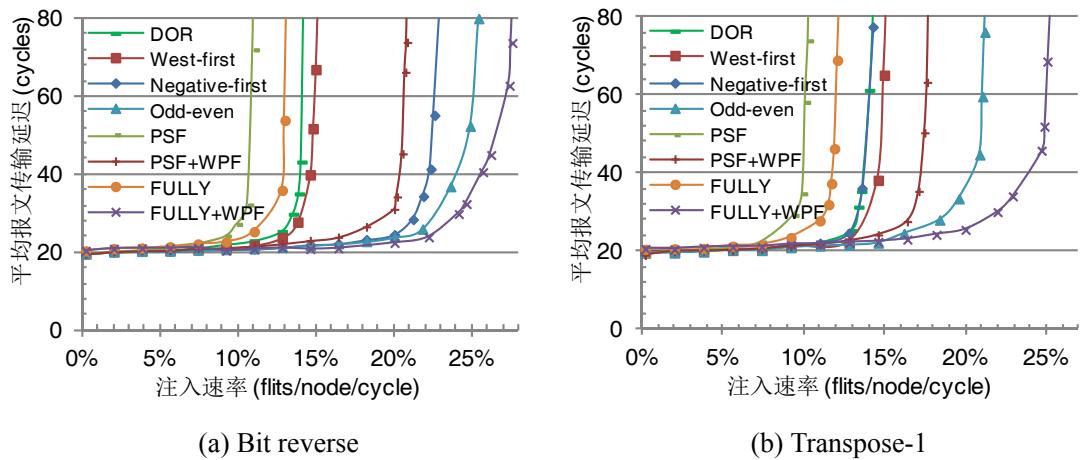


图 4.15 路由算法在 8×8 mesh 网络中的性能

道利用率，因此，其在越大规模网络中获得了越高的性能提升。在 8×8 mesh 中，**FULLY+WPF** 在 bit reverse 和 transpose-1 模式下的性能平均比 **FULLY** 高 108.2%，而在 4×4 mesh 中，该性能提升是 93.1%。报文传输跳数在大规模网络中会增加，此时报文进入逃逸虚通道的概率也会上升。在 **PSF** 和 **PSF+WPF** 中，一旦报文进入逃逸虚通道，其后续传输只能使用逃逸虚通道，该报文将丧失路由灵活性。因此，**FULLY+WPF** 和 **PSF+WPF**（或 **FULLY** 和 **PSF**）的性能差异随着网络规模的增加而上升。在大规模网络中，提供充足的路由灵活性变得更为重要。

综上所述，虽然更低的单切片报文比例、更浅的虚通道深度、或更多的虚通道数会降低 WPF 策略的效果，但是在所评估的各种配置中 WPF 策略都获得了明显的性能提升。随着网络规模的增大，WPF 策略和路由灵活性对性能的影响更为明显。由于 WPF 策略提高了虚通道利用率，其可以在使用保守分配策略一半缓存资源或虚通道数目时获得与保守分配策略类似的性能。

4.7 进一步讨论

4.7.1 报文长度和虚通道深度

Cache 一致性协议报文的长度一般表现出双峰分布，即只包含两种长度的报文。然而，cache 行压缩之类的优化设计^[203, 215] 会产生其它长度的报文，此时网络报文长度可能分布在单切片和体系结构所能支持的最大报文切片数之间。为了在这些网络上使用全报文发送，需要在图4.9的第一级仲裁器上增加记录下游 VC 状态的寄存器。设计中需要考虑对哪些报文采用全报文发送，可以使用全报文发送的报文的长度必须至少比虚通道深度少 1。由于长报文使用全报文发送的概率较低，设计者可以忽略它们，具体权衡取决于网络报文长度分布、硬件实现开销

和预期性能提升。本章假设虚通道深度小于网络最长报文长度，如果虚通道深度大于最长报文长度，虫孔交换网络采用保守虚通道分配策略的性能会低于虚切通网络，因为后者允许多个报文同时存储在同一虚通道中，此时，在虫孔交换网络中使用 WPF 的性能与虚切通网络相同。WPF 策略在虚通道深度小于最长报文长度时，依然允许多个报文同时存储于一条虚通道上，因此提高了设计灵活性。

4.7.2 DAMQ 和混合流控机制

为提高缓存的利用率，人们在片外^[216] 和片上网络^[49, 50] 中提出了动态分配多队列（Dynamically Allocated Multi-Queue, DAMQ）缓存结构。在虫孔交换网络中，如果完全自适应路由算法允许不同报文的切片同时存在于一条虚通道中，DAMQ 结构也有可能出现类似于图4.3 中的死锁。本章研究出发点是期望通过扩展死锁避免理论以提高虚通道的利用率，因此，WPF 策略与 DAMQ 技术是正交的。WPF 策略可以被视为一种混合流控机制，当前也存在很多混合流控机制，包括 hybrid switching^[171]、buffered wormhole^[172] 和 layered switching 机制^[78] 等。Hybrid switching^[171] 和 buffered wormhole^[172] 将虫孔交换网络中处于阻塞状态的报文移除到路由器中央缓存或处理器的缓存中，从而释放该报文所占用的物理链路。Layered switching^[78] 将虫孔交换网络中的长报文划分成多个分组，并且在这些分组内部保持交叉开关分配器的结果。本章提出的 WPF 策略与这些设计的出发点不同，WPF 策略主要用于提升虫孔交换网络中完全自适应路由算法的性能。

4.8 本章小结

本章主要研究了面向 Cache 一致性通信的完全自适应路由算法设计。基于已有的虫孔交换网络死锁避免理论设计的完全自适应路由算法需要采用保守虚通道分配策略，该策略在 cache 一致性片上网络中会给路由算法带来大量性能损失。因此，本章提出了面向虫孔交换网络完全自适应路由算法的全报文发送虚通道分配策略，该策略允许多个报文同时存在于一条虚通道内部，从而在虚通道受限的环境中带来了很大的性能提升。全报文发送策略获得的性能提升得益于 cache 一致性片上网络传输的报文大部分长度较短。本章证明如果路由算法在采用保守虚通道分配策略时没有死锁，则其采用全报文发送策略同样也不会出现死锁。基于全报文发送策略，本章进一步给出了一种完全自适应路由算法设计，该设计能够在较低硬件开销情况下提供较高的路由灵活性。与保守虚通道分配策略相比，全报文发送策略在合成流量模式下平均获得了 88.9% 的饱和吞吐率提升。在网络负载较重的 PARSEC 应用程序中，全报文发送策略最高获得了 37.8% 的全系统性能

提升。与保守策略相比，全报文发送策略能够在使用一半的缓存资源或者虚通道数目时获得类似的性能。

第五章 面向 Torus 片上网络的切片气泡流控机制

Cache 一致性协议需要同时传输长报文和短报文，已有的 torus 网络死锁避免理论不能高效地处理这种混合长度报文的传输。一种传统的设计需要使用两条虚通道实现死锁避免，其提高了对缓存资源的需求，增大了分配器的规模，降低了路由器的频率。虚切通网络的一些优化设计只使用一条虚通道，但是它们需要将每个短报文视为长报文，从而降低了缓存利用率，给性能带来了负面影响。本章针对这些设计的局限性，提出了切片气泡流控（Flit Bubble Flow Control, FBFC）。该理论的关键点是：虫孔交换网络可以通过在 ring 上维持一个空闲缓存单元来避免死锁。切片气泡流控只需使用一条虚通道，因此支持更高的路由器频率；同时，它无需将短报文视为长报文，从而提高了缓存利用率。基于此死锁避免理论，本章提出了两种实现：本地切片气泡策略（FBFC-L）和关键切片气泡策略（FBFC-C），它们的性能都显著优于已有设计。

5.1 引言

随着集成核数的增加，片上网络^[1]应该采用高带宽、低延迟的拓扑结构。Torus 网络是一种较好的候选对象，其回绕链路有效地将片上丰富的连线资源转化成带宽^[1]，同时也降低了网络的跳数和延迟^[34]。传统 mesh 网络的通信拥塞在中间区域^[33]，容易导致该区域成为系统性能的瓶颈。相反，torus 结构的节点对称性有利于在整个网络上实现负载均衡^[34, 59]，因此，许多片外^[154, 155, 165, 210]和片上网络^[17, 152]都采用 torus 拓扑结构。

尽管 torus 网络具备许多优秀的特性，其回绕链路引入的循环依赖增加了死锁避免的复杂性。一种高效的死锁避免机制应该在较低硬件开销下支持较高的性能。死锁避免机制应该尽量使用最少虚通道^[168]；更多的虚通道不仅增加了设计的复杂性，也降低了路由器的频率。另一方面，缓存是一种非常宝贵的片上资源^[45, 49]；一种高效的死锁避免机制应该在有限缓存条件下获得尽可能高的性能。然而，已有的死锁避免机制不能满足这些目标。

一种传统的设计^[163] 使用了两条虚通道来移除 torus 网络链路的循环依赖关系；两条虚通道增大了分配器的规模，由此降低了路由器的频率。虚切通网络^[164]的一些优化设计^[82, 217]通过禁止报文占用 ring 上的最后一个报文大小的缓存来避免死锁；它们只使用一条虚通道。然而，当网络中出现不同长度的报文时，它们需要将每个报文都视为最长报文^[155]。这个要求是为了避免死锁，但是当网络中存在大量的短报文时，该要求会降低缓存利用率和性能。

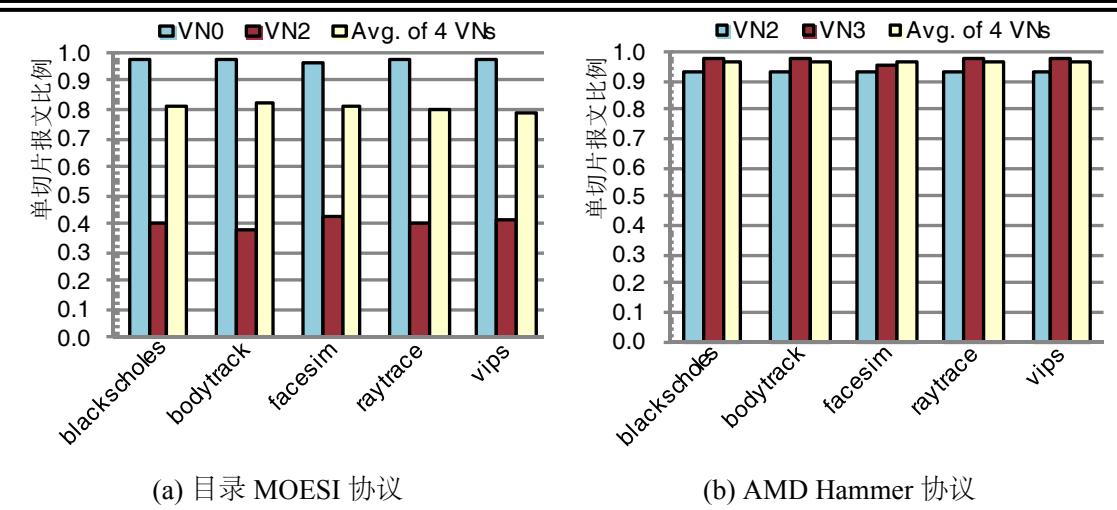


图 5.1 PARSEC 测试集单切片报文分布

Cache 一致性片上网络一般传输两种长度的消息：短控制消息和长数据消息。即使系统采用多个虚拟网络（Virtual Network, VN）^[163] 来防止协议层的死锁，在每个虚拟网络中依然会存在混合长度的报文。比如说，在 AlphaServer GS320 机器^[183] 中，短 read request 消息和长 write-back 消息都是通过 VN0 传输，而短 write-back acknowledgement 消息和长 read response 消息都是通过 VN1 传输；这两个虚拟网络上都传输混合长度的报文。类似的，DASH 机器^[218]、Origin 2000 机器^[219] 和 Piranha 机器^[220] 的所有虚拟网络都传输混合长度的报文。

基于典型的 128 位片上网络切片宽度^[45, 49, 150]，网络中占绝大部分的短消息只包含一个切片；剩余的消息包含一个 64 字节的 cache 行，它们的长度是 5 个切片。图 5.1 给出了一些具有代表性 PARSEC 应用程序^[201] 在两种 cache 一致性协议^[185, 221] 下的报文长度分布。这两种协议都需要采用 4 个虚拟网络来避免协议层的死锁，对于每种协议，其中 2 个虚拟网络上传输混合长度的报文，另外 2 个虚拟网络只传输单切片报文。MOESI 目录协议^[185] 的 VN0 和 AMD Hammer 协议^[221] 的 VN2 和 VN3 中的单切片报文比例都高于 90%。在此如此高的单切片报文比例下，将每个短报文视为长报文严重限制了缓存的利用率。如第 5.6.2 节所示，即使在网络饱和时，这些设计的最大缓存利用率依然低于 40%。这种现象带来了大量的带宽和性能损失，因此，很有必要对 torus 网络的死锁避免机制进行改进。

本章提出了一种 torus 网络死锁避免理论：切片气泡流控（Flit Bubble Flow Control, FBFC）。切片气泡流控采用虫孔交换机制^[165]，其死锁避免的原理是在每个 ring 上维持一个空闲缓存单元。切片气泡流控只需要使用一条虚通道，因此降低了分配器大小，提升了路由器的频率。此外，与之前的气泡机制不同，每个短报文无需被视为长报文，因此带来了较高的缓存利用率。基于切片气泡理论，本

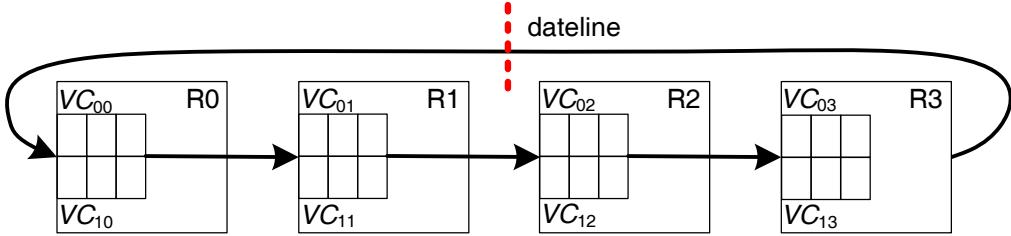


图 5.2 Dateline 需要使用两条虚通道 (VC)

章进一步给出了两种实现：本地切片气泡策略和关键切片气泡策略。

实验结果表明，切片气泡流控的性能显著高于已有设计。切片气泡流控的路由器频率比 dateline 机制^[163] 高约 30%；在 4×4 torus 网络的合成流量模式测试中，切片气泡流控的性能分别比本地气泡策略^[217] 和关键气泡策略^[82] 高 92.8% 和 34.2%；在 8×8 torus 网络中，这些性能提升分别是 107.2% 和 40.1%；在 PARSEC 测试集中，切片气泡流控相对于本地气泡策略平均获得了 13.0% 的性能提升，最大性能提升是 22.7%。切片气泡流控的性能优势在越少的缓存数量下越明显。

5.2 传统设计的局限

本节分析已有的 torus 网络死锁避免机制的局限性。实现 torus 网络死锁避免的通用做法是先在一个 ring 上避免死锁，然后与维序路由算法结合。因此，本节使用 ring 网络进行讨论。

5.2.1 Dateline

如图5.2所示，dateline^[163] 使用 VC_{0i} 和 VC_{1i} 两条虚通道实现死锁避免。为了消除循环通道依赖，dateline 要求报文在越过网络中配置的‘日期变更线 (dateline)’之后必须使用 VC_{1i} 。Dateline 机制既可以用于基于报文的虚切通交换网络，又可被用于基于切片的虫孔交换网络。这种设计需要使用两条虚通道，因此增大了分配器的规模，降低了路由器的频率。

5.2.2 本地气泡策略 (LBS)

气泡流控 (Bubble Flow Control)^[217, 222] 是面向虚切通 torus 网络的一种死锁避免理论，它通过禁止报文占用 ring 上最后一个报文大小的缓存资源（报文气泡）实现死锁避免，该机制只需使用一条虚通道。理论上，ring 上任何位置存在一个空闲气泡即允许报文移动，因此可以避免死锁^[217, 222]。然而，由于获得网络全局缓存使用状态和协调所有节点的缓存分配比较困难，已有实现采用一种本地策略^[217, 222]。具体来说，只有在接收虚通道中的空闲缓存数量大于等于两个报文大小时才允许报文注入。如图5.3所示，报文 P_0 、 P_1 和 P_2 在等待注入，理论上这

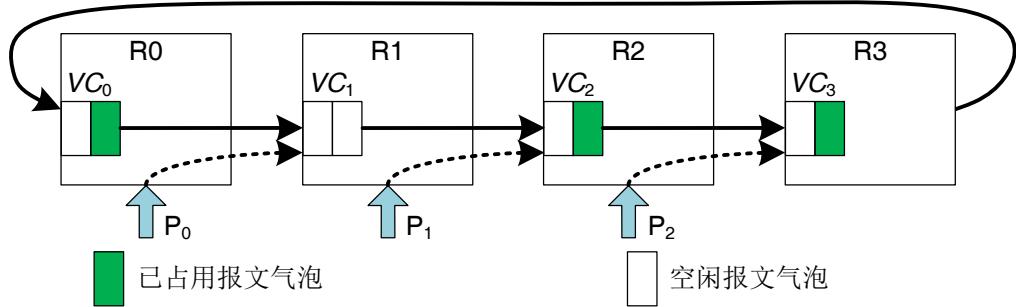


图 5.3 LBS 要求每条虚通道至少能容纳两个报文

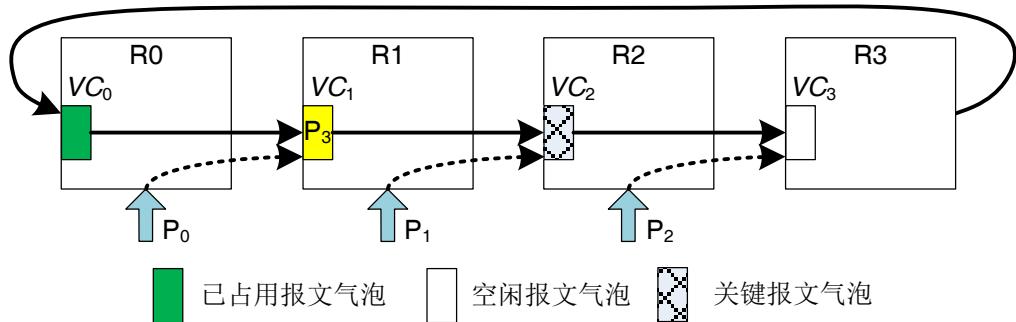


图 5.4 CBS 要求每条虚通道能容纳一个报文

些报文都可以注入，因为在它们注入之后 ring 的 VC_1 依然剩余一个空闲气泡。但是根据本地实现策略，由于只有 VC_1 有两个报文大小的空闲缓存，只有报文 P_0 可以注入。本地气泡策略（Localized Bubble Scheme, LBS）要求每条虚通道的深度至少能容纳两个报文。

5.2.3 关键气泡策略 (CBS))

基于气泡流控理论，Chen 等提出了关键气泡策略（Critical Bubble Scheme, CBS）^[82]。CBS 在每个 ring 上将至少一个报文气泡标记为关键气泡。报文允许注入的条件是其注入不会占用关键气泡。在图5.4给出的例子中， VC_2 上的气泡被标记为关键气泡。报文 P_2 可以注入；报文 P_1 的注入会占用关键气泡，因此其不能注入。限制关键气泡只能被当前已处于 ring 中的报文占用可以保证 ring 上维持一个空闲的报文气泡；邻居节点通过控制信号记录关键气泡的移动。在图5.4中，如果报文 P_3 从路由器 R1 发送到 VC_2 中，则关键气泡会移动到 VC_1 中，此时 VC_1 维持一个空闲的报文气泡。CBS 要求每条虚通道的深度能容纳一个报文。

5.2.4 处理变长报文的低效性

LBS 和 CBS 都是面向虚切通网络提出的，它们能高效处理等长报文。然而，正如 BlueGene/L 所观察到的，LBS 在处理变长报文时会出现死锁，这是因为短报

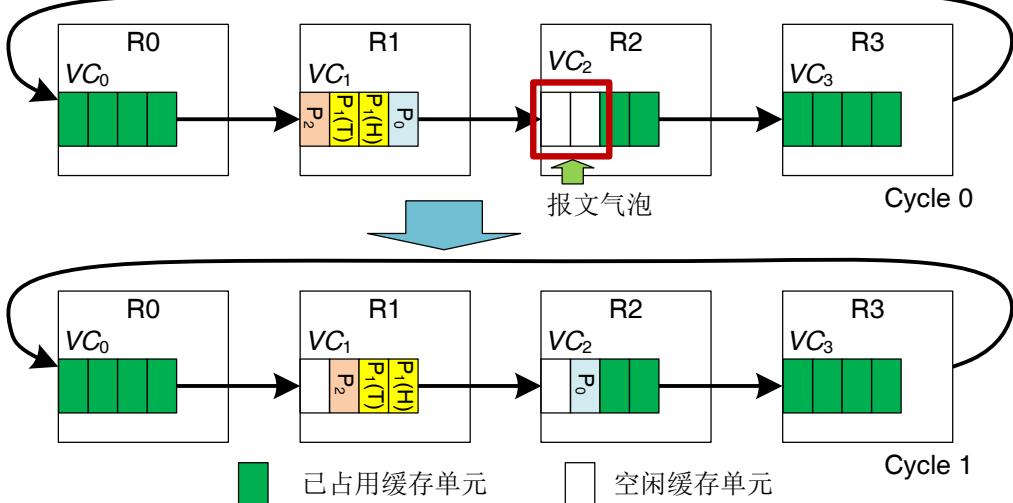


图 5.5 LBS 在处理变长报文时面临的死锁

文的传输会将一个完整的报文气泡切分^[155]。图5.5给出了一个实例，该图假设了两种大小的报文：单切片报文和具有两个切片的报文。请注意，从本图开始，每个方框代表一个切片大小的缓存单元，而在图5.3和图5.4中，每个方框代表代表报文大小的缓存单元数。在 Cycle 0 时，一个完整的报文气泡位于 VC_2 中；单切片报文 P_0 在 Cycle 1 发送到 VC_2 中。此时，该完整报文气泡被切分成两部分，分别位于 VC_1 和 VC_2 中。虚切通交换机制要求报文发送时下游虚通道必须拥有足够容纳整个报文的缓存资源。此时 VC_2 的空闲缓存资源小于报文 P_1 的长度， P_1 不能发送，因此，网络中出现了死锁。CBS 和 LBS 是基于同一理论设计的，它在处理混合长度报文时也有类似的问题。

为了解决这个问题，BlueGene/L 将网络中的每个报文都视为最长报文^[155]，此时，不会有气泡切分情况出现。但是这种设计在大部分报文是短报文的 cache 一致性片上网络中（图5.1）会降低缓存利用率，同时带来性能损失。为了解决已有设计的缺陷，本章在虫孔交换机制^[165]的基础上提出了切片气泡流控，该流控通过在 ring 上维持一个空闲缓存单元避免死锁。与 LBS 和 CBS 不同，切片气泡流控不需要将每个报文都视为最长报文，因此带来了性能的提升；此外，切片气泡流控只需使用一条虚通道，因此提高了路由器的频率。

5.3 切片气泡流控机制

本节首先证明切片气泡流控是无死锁的，然后给出两种切片气泡流控的实现策略，最后讨论饿死现象的处理。

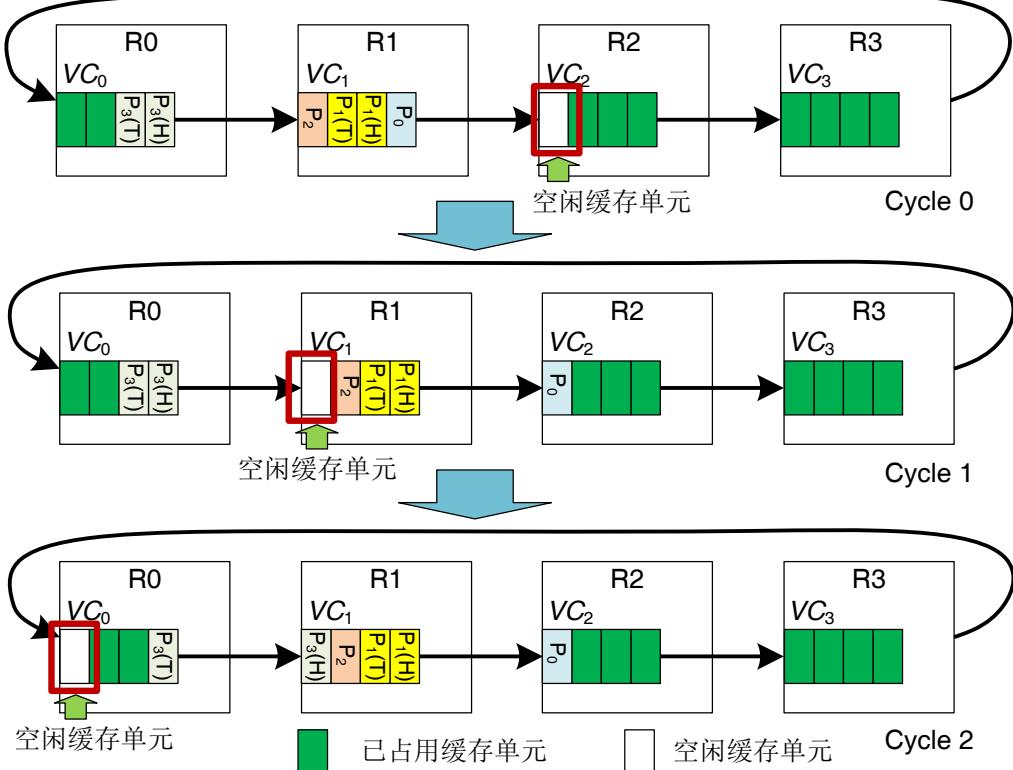


图 5.6 虫孔交换网络中的报文路由实例

5.3.1 理论描述

虚切通网络的死锁避免需要维持报文大小的空闲缓存；与之不同，虫孔交换网络的死锁避免只需维持切片大小的空闲缓存，这是因为虫孔交换网络在下游虚通道有一个空闲缓存单元时就可以发送报文^[165]。图5.6给出了一个例子，报文 P_0 和 P_2 是单切片报文。在 Cycle 0， VC_2 有一个空闲缓存单元。报文 P_0 在 Cycle 1 时被发送；由于 P_0 的移动， VC_1 中出现了一个空闲缓存单元。类似的，报文 P_3 的头切片在 Cycle 2 被发送至 VC_1 ，从而给 VC_0 创造了一个空闲缓存单元。这个空闲缓存单元在 ring 中不停地循环，使所有切片都能获得转发机会，因此网络中不存在死锁。

对于已注入的报文，它们的传输在占用一个缓存的同时会释放另一个缓存。因此，这些报文的传输不会减少整个网络上的空闲缓存数目；只有报文注入会减少网络上的空闲缓存数目。本章提出了如下理论。

定理 5.1：如果报文注入在虫孔交换 ring 网络上维持一个空闲缓存单元，则不会出现死锁。

证明思路：虫孔交换网络的死锁状态包含一个循环等待的切片集合，并且集合中的任何切片都不能移动^[198]。Ring 网络上的循环等待关系需要所有虚通道参

与，因此，只需证明任意虚通道上的某个切片可以移动即可。

证明：假设网络中唯一的空闲缓存单元位于虚通道 VC_{i+1} 中，其它虚通道都被完全占用。记 VC_{i+1} 的上游虚通道为 VC_i ，考查当前处于 VC_i 头部的切片 f ，有两种情况：

1) f 是一个头切片。如果 f 已经到达了目标节点，则 f 可以排出 (ejection)；如果 f 需要发送到 VC_{i+1} 中，则考查最近使用 VC_{i+1} 的报文 P_k ，有两种情况：

1.1) P_k 是从 VC_i 发送到 VC_{i+1} 中的。由于此时另一个报文的头切片已经位于 VC_i 头部，报文 P_k 的尾切片一定已经进入 VC_{i+1} 。根据虫孔交换机制， f 可以发送到 VC_{i+1} 中。

1.2) P_k 是注入到 VC_{i+1} 中的。此时报文 P_k 的尾切片一定已经进入 VC_{i+1} 。否则，其尾切片会占用唯一的空闲缓存单元，这违反了报文注入应该在 ring 中维持一个空闲缓存单元的前提条件。因此， f 可以发送。

2) 如果 f 是体切片或尾切片，其也可以排出或发送到 VC_{i+1} 。

在所有可能情况下，至少有一个切片可以移动，因此不存在死锁。 ■

上述理论被称为切片气泡流控 (Flit Bubble Flow Control, FBFC)，其通过维持一个空闲缓存单元 (切片气泡) 避免了 ring 中的死锁。维序路由消除了维度间的循环依赖，因此将维序路由与 FBFC 结合使用可以避免 torus 网络的死锁。FBFC 的气泡是切片大小的，其不存在气泡切分的情况；与 LBS 和 CBS 不同，FBFC 无需将每个报文视为最长报文，因此比较适合于 cache 一致性片上网络。FBFC 只使用一条虚通道，其路由器频率高于 dateline 设计。FBFC 使用虫孔交换机制发送已处于 ring 中的报文，它要求报文注入维持一个空闲缓存单元，下面给出满足这一条件的两种实现策略。

5.3.2 本地切片气泡策略 (FBFC-L)

实现切片气泡流控的关键是保证报文注入在 ring 中维持一个空闲缓存单元。理论上，如果报文注入之后，ring 上任意位置依然存在一个空闲缓存单元，则允许该报文注入。但是由于收集和分配全局缓存使用情况比较复杂，本节给出一种本地实现策略：本地切片气泡 (FBFC-Localized, FBFC-L)。

与维序路由算法一起使用时，对转维报文的处理与注入报文相同。FBFC-L 的规则如下：(1). 对于在同一维度内传输的报文，只要下游虚通道有 1 个空闲缓存单元，则允许报文发送；这与虫孔交换相同。(2). 对于需要注入或转维的报文，只有当下游虚通道的空闲缓存数多于报文切片数时，才允许报文注入或转维。该要求保证报文注入后，下游虚通道至少剩余一个空闲缓存单元以避免死锁。

图5.7给出了 FBFC-L 的实例；报文 P_2 、 P_3 和 P_4 等待注入。虚通道 VC_2 和

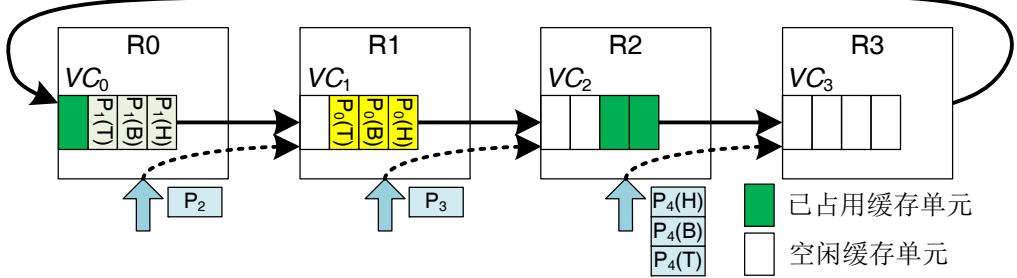


图 5.7 FBFC-L 报文路由实例

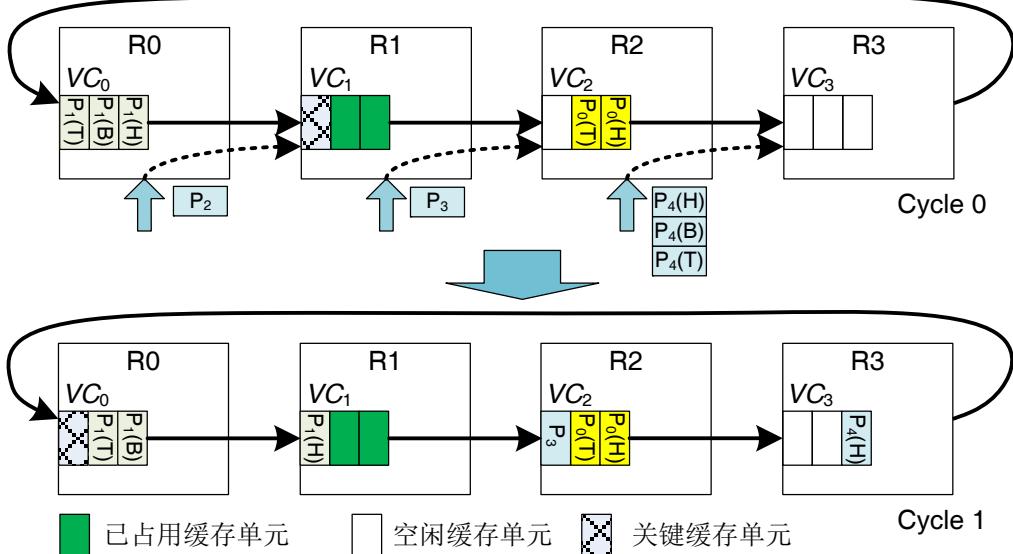
VC_3 的空闲缓存数分别是 2 个和 4 个，它们分别比报文 P_3 和 P_4 的切片数大 1；因此， P_3 和 P_4 可以注入。在它们注入之后，ring 中至少存在两个空闲缓存单元。由于虚通道 VC_1 只有一个空闲缓存单元，报文 P_2 不能注入。然而，虫孔交换允许 VC_1 的空闲缓存单元用于 P_1 的头切片发送。FBFC-L 要求虚通道深度至少比最长报文的切片数大 1。

5.3.3 关键切片气泡策略 (FBFC-C)

为了进一步降低对虚通道深度的需求，本节提出了关键切片气泡策略 (FBFC-Critical, FBFC-C)。FBFC-C 将至少一个缓存单元标记为关键缓存单元 (关键切片气泡)，并限制关键缓存单元只能被同一维度内传输的报文使用。FBFC-C 的规则如下：(1). 对于在同一维度内传输的报文，只要下游虚通道有 1 个空闲缓存单元，无论该单元是关键缓存单元或普通缓存单元，都允许报文发送。(2). 对于需要注入或转维的报文，只有当下游虚通道的空闲普通缓存数大于等于报文切片数时，才能允许报文注入或转维；也就是说，报文注入不能占用关键缓存单元。这个要求保证 ring 上始终存在一个空闲缓存单元。

图 5.8 给出了 FBFC-C 的实例。关键缓存单元在 Cycle 0 时处于虚通道 VC_1 上。虚通道 VC_2 和 VC_3 的空闲普通缓存数分别等于报文 P_3 和 P_4 的切片数，因此这两个报文可以注入。它们的注入没有占用关键缓存单元，则关键缓存单元一定处于空闲状态；也就是说，ring 网络至少存在一个空闲缓存单元。虚通道 VC_1 的唯一空闲缓存是关键缓存单元，报文 P_2 不能注入。但是该关键缓存单元允许报文 P_1 的头切片在 Cycle 1 时发送到 VC_1 。此时，关键缓存单元被移至虚通道 VC_0 中，即将由 P_1 头切片移动产生的空闲缓存标记为关键缓存单元，第 5.4 节给出了更多实现细节。FBFC-C 要求虚通道深度至少等于最长报文的切片数，这个要求与 CBS 相同，是 LBS 的一半，比 FBFC-L 少 1 个缓存单元。

FBFC-L 和 FBFC-C 的报文注入与虚切通交换机制类似：报文只有在下游虚通道有足够的空闲缓存时才能注入。FBFC-L 和 FBFC-C 可以被视为对虫孔交换网络的注入或转维报文采用虚切通交换机制，这些混合流控机制是实现切片气



泡流控的直观方式，第5.8节讨论了其它实现方式。

5.3.4 饿死现象

与 LBS^[217] 和 CBS^[223] 类似，FBFC-L 和 FBFC-C 需要处理饿死现象。FBFC-L 的饿死现象本质上与 LBS 的饿死现象是一样的：它们是由于报文注入需要的缓存资源高于报文在网络中传输需要的缓存资源造成的^[217]。图5.9给出了 FBFC-L 的饿死实例：如果节点 R0 持续向节点 R3 发送类似于 P_0 的报文，则报文 P_1 无法注入。FBFC-L 使用的饿死预防机制与 LBS 类似^[217]：当节点检测到饿死时，会通知 ring 中其它节点停止注入报文。饿死预防信号（图 5.9 中的 starve 信号）通过一个边带网络传输，网络中配置了饿死判定阈值。当报文 P_1 不能注入的时钟数超过这个阈值时，节点 R1 会将输出 starve 信号置位。节点 R0 接收到 starve 信号后会停止注入报文，同时将该信号转发给节点 R3。最后，除了 R1 以外的其它节点都会停止注入报文，此时，R1 可以将报文 P_1 注入，然后 R1 将 starve 信号清空以通知其它节点恢复报文注入。为处理多个节点同时检测到饿死的情况，starve 信号携带了用于排序的 router ID 和 time stamp 字段。只有当输入的 starve 信号的顺序高于节点正服务的 starve 信号时，输入的 starve 信号才会被转发到邻居节点。

除了上述饿死现象以外，FBFC-C 还面临着由关键切片气泡停滞造成饿死现象，CBS 也面临类似的问题^[223]。图5.10给出了 FBFC-C 的饿死实例。关键（切片）气泡在 Cycle 0 时处于虚通道 VC_3 上。关键气泡移动取决于报文发送；如果所有到达虚通道 VC_2 的报文（图中 P_0 ）都是以节点 R2 为目标节点，这些报文将会排出，而非发送到 VC_3 。此时，关键气泡停滞在 VC_3 上，导致报文 P_1 无法注

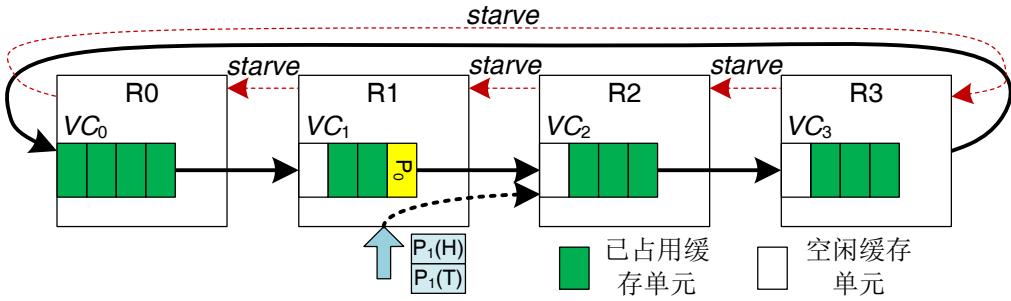


图 5.9 FBFC-L 饿死实例

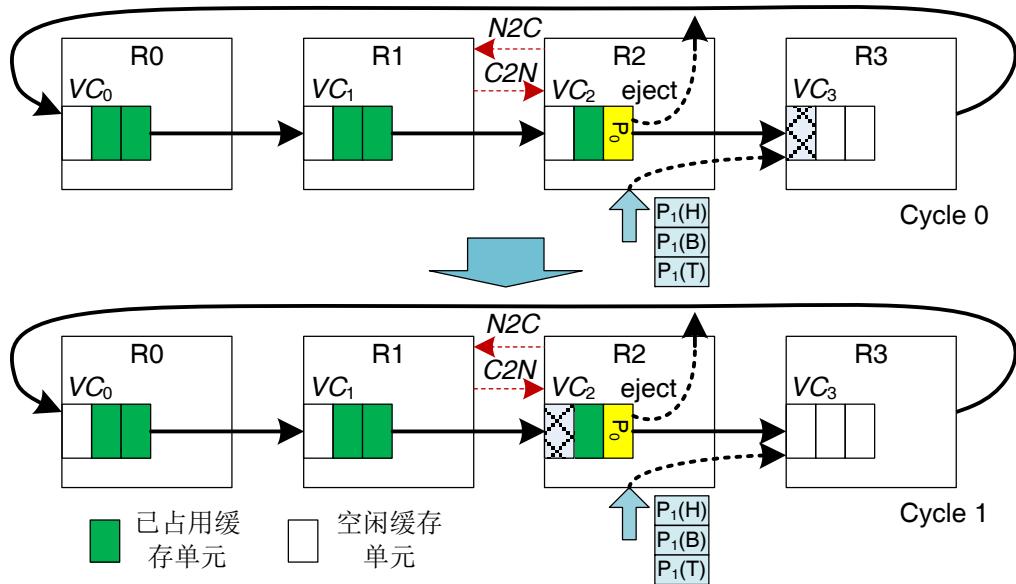


图 5.10 FBFC-C 饿死实例

入。主动移动关键气泡可以预防这种饿死现象^[223]。具体而言，如果上游虚通道有空闲普通气泡，则可以将关键气泡移至上游虚通道。如图5.10所示，邻居路由器之间配置了一对 $N2C$ 和 $C2N$ 信号。如果节点 R_2 检测到 VC_3 的关键气泡阻止了报文 P_1 的注入，则置位输出 $N2C$ 信号。如果此时 VC_2 有空闲普通缓存单元， R_1 将该普通缓存单元更改为关键缓存单元，同时置位 $C2N$ 信号，以通知 R_2 将 VC_3 中的关键气泡更改为普通气泡。此时，报文 P_1 可以注入。注意，输入虚通道的气泡状态是由上游路由器记录的。

5.4 路由器流水线和微结构

本节讨论 FBFC 使用的虫孔交换路由器结构，以及 LBS 和 CBS 使用的虚切通交换路由器结构。

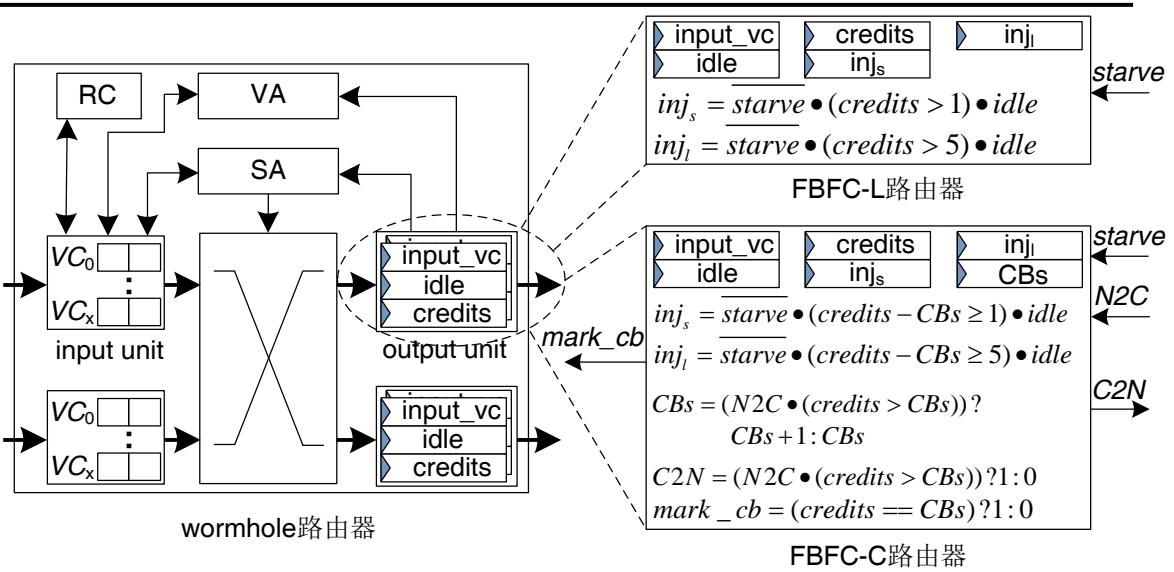


图 5.11 FBFC 路由器微结构

5.4.1 FBFC 路由器

图5.11左侧给出了传统虫孔交换路由器的微结构，其主要模块包括输入单元（input units）、路由计算（RC）、虚通道分配器（VA）、交叉开关分配器（SA）、交叉开关（crossbar）和输出单元（output units）^[7, 163]。流水线阶段包括路由计算（RC）、虚通道分配（VA）、交叉开关分配（SA）和交叉开关传输（switch traversal）^[7, 163]。输出单元记录下游虚通道的状态，其中 *input_vc* 寄存器记录下游虚通道被分配的输入虚通道号，*idle* 寄存器记录是否已经将报文尾切片发送至下游虚通道，*credits* 寄存器记录下游虚通道的信元数，VA 和 SA 需要访问这些寄存器以获得下游虚通道的状态。为了获得更高的基准性能，虫孔交换路由器采用超前路由计算^[7, 163]将 RC 与 VA 段并行执行；为了与虚切通交换路由器公平比较，虫孔交换路由器试图为整个报文传输过程维持交叉开关分配结果，即将交叉开关优先分配给已获得授权的虚通道^[41]；

FBFC 路由器需要修改输出单元。如图5.11右上侧所示，为了支持一致性协议双峰分布的报文长度，FBFC-L 路由器使用 *inj_s* 和 *inj_l* 寄存器分别记录是否允许短报文和长报文的注入（或转维）。当报文注入时，VA 模块根据其长度检查相应寄存器的值。假设短报文和长报文分别包含 1 个和 5 个切片，短报文要求下游虚通道至少有 2 个空闲缓存单元，长报文要求至少有 6 个空闲缓存单元。如果输入 *starve* 信号被置位，*inj_s* 和 *inj_l* 被清零以禁止报文注入。图5.11中给出了相应的实现逻辑，这些逻辑可以被预算算，因此不影响关键路径。

图5.11右下侧给出了 FBFC-C 路由器的输出单元结构。与 FBFC-L 路由器相比，FBFC-C 路由器还需要一个额外寄存器（*CBs*）来记录关键气泡数目。*Inj_s* 和

inj_l 的计算逻辑根据 FBFC-C 的注入规则进行相应修改：只有当下游虚通道的普通缓存数 ($credits - CBs$) 不小于报文切片数时，才允许报文注入。FBFC-C 路由器通过主动移动关键气泡来预防饿死。当输入 $N2C$ 信号置位时，输出单元检查是否有空闲普通缓存单元；如果有，则将 CBs 加 1，同时置位输出 $C2N$ 信号以通知邻居路由器将关键气泡更改为普通气泡。当某个输出切片需要占用下游虚通道的关键气泡时，输出单位会置位 $mark_cb$ 信号，该信号通知上游路由器将新释放的缓存单元标记为关键气泡。以图5.8为例，当报文 P_1 的头切片输出时，路由器 R0 将 $mark_cb$ 信号置位以通知 R3 将 VC_0 中的新释放的缓存标记为关键气泡。与 FBFC-L 路由器类似，这些逻辑也不在关键路径上。

5.4.2 VCT 路由器

为了给实验部分提供足够的背景知识，本节讨论 LBS 和 CBS 使用的虚切通交换路由器。传统虚切通交换路由器^[163, 224]与图5.11中的虫孔交换路由器非常相似，两者的区别主要是虚通道分配 (VA) 过程：只有当能存储整个报文时，虚切通路由器才允许虚通道重新分配。报文的发送只返回一个信元，其表示报文大小的缓存资源被释放。本章优化设计了虚切通路由器：路由器为整个报文传输过程保持交叉开关分配结果；由于 VA 保证了下游虚通道有足够的缓存资源，只要报文头切片开始发送，则后续切片可以不间断地发送出去；也就是说，报文当前占用的缓存资源一定会在有限时间内被释放；因此，当头切片开始发送时，就可以向上游路由器回复信元信号，该信元表示报文占用所有缓存将被释放。这种超前信元回复机制允许输入报文对虚通道缓存的使用与输出报文对虚通道缓存的释放重叠进行^[224]，该优化有利于报文注入，第5.7.2节评估了其对性能的影响。LBS 路由器的输出单元与 FBFC-L 路由器的输出单元类似，但是 LBS 路由器不区分长报文和短报文，其只需一个 inj 寄存器，同时 $credits$ 寄存器记录的信元数目是以报文长度而非切片长度为单位的。CBS 路由器的输出单元与 FBFC-C 路由器的输出单元也有上述两个区别。此外，CBS 路由器只有由关键气泡停滞带来的饿死，因此 CBS 路由器没有 $starve$ 信号输入。本章第5.7.7节讨论了 FBFC 路由器和 VCT 路由器的硬件开销。

5.5 实验方法

实验对时钟精确模拟器 Booksim^[163]进行修改，以模拟第5.4节中描述的路由器流水线和微结构。实验实现了 FBFC-L 和 FBFC-C，同时也实现 dateline、LBS 和 CBS 以进行性能比较；实验过程既使用了合成流量模式，又使用了真实应用程序。虚拟网络之间是相互独立的，因此，合成流量模式实验只使用一个虚拟网

表 5.1 延迟实验结果 (单位 FO4)

| | Ring (3 端口) | | | Torus (5 端口) | | | |
|-----|-------------|----------|------|--------------|----------|------|-----|
| | bubble | dateline | 增量 | bubble | dateline | 增量 | |
| 1VN | VA | 8.4 | 12.2 | 45% | 10.0 | 13.8 | 38% |
| | SA | 6.9 | 11.7 | 69% | 8.5 | 13.3 | 57% |
| 2VN | SA | 11.7 | 16.5 | 41% | 13.3 | 18.1 | 36% |
| 3VN | SA | 14.5 | 19.3 | 33% | 16.1 | 20.9 | 30% |
| 4VN | SA | 16.5 | 21.3 | 29% | 18.1 | 22.9 | 27% |

络。报文长度呈现双峰分布：网络中包含单切片报文和 5 切片报文。基准单切片报文 (Single Flit Packet, SFP) 比例是 80%。模拟器的预热 (warmup) 时间是 10,000 个时钟周期，性能测试时间是预热之后的 100,000 个时钟周期。

由于 1 维 torus (ring) 和 2 维 torus 在片上网络中被大量应用，实验评估主要面向这两种拓扑。实验使用维序路由算法。缓存是非常宝贵的片上资源^[45, 49]，因此，在大部分实验中，每个虚拟网络配置了 10 个缓存单元。所有的气泡设计 (LBS、CBS、FBFC-L 和 FBFC-C) 都在虚拟网络上配置了一条虚通道。Dateline 需要使用两条虚通道，每条虚通道包含 5 个缓存单元，这个深度可以覆盖信元往返延迟^[163]。与传统设计^[163] 中要求注入报文首先使用 VC_{0i} ，在越过 ‘日期变更线 (dateline)’ 后切换到 VC_{1i} 不同，实验对 dateline 采用了一种负载均衡优化^[154, 163]：报文在注入时根据其是否会越过 ‘日期变更线 (dateline)’ 决定使用的虚通道类型，如果报文会越过 ‘dateline’，使用 VC_{1i} ，否则，使用 VC_{0i} 。CBS 和 FBFC-C 在每个 ring 上设置了一个关键气泡，CBS 将 5 个缓存单元标记为一个关键报文气泡，FBFC-C 将 1 个缓存单元标记为关键切片气泡。LBS 和 FBFC-L 中的饿死阈值为 30 个时钟周期，CBS 和 FBFC-C 处理关键气泡停滞导致的饿死的阈值是 3 个时钟周期。

虚通道分配器 (VA) 和交叉开关分配器 (SA) 的延迟决定了路由器的频率^[38, 44]。Dateline 使用 2 条虚通道，增大了分配器规模，导致了更长的关键路径。表 5.1 使用一个与工艺无关的延迟模型^[38] 评测了采用矩阵仲裁器^[38, 163] 的可分离分配器^[38] 的延迟。VA 是为每个虚拟网络 (VN) 独立配置^[44]，因此，SA 在多虚拟网络时成为关键路径。当配置 4 个虚拟网络时，dateline 的 SA 延迟比气泡设计约高 30%。Becker 和 Dally 使用 45 nm 工艺的综合结果也表明，在不同类型的分配器上，配置 8 条虚通道的 SA 延迟比配置 4 条虚通道的 SA 延迟高 15%~26%^[44]。

实验使用两个模拟器评估全系统性能：FeS2^[185] 模拟 x86 体系结构，Booksim 模拟 FeS2 产生的报文在片上网络中的传输。实验模拟了一个 16 核，拓扑结构是 4×4 torus 的 CMP 平台，并评估了 PARSEC 测试集^[201] 在 16 线程运行时的性能。

表 5.2 全系统模拟参数

| 参数名 | 参数值 |
|------------------|---------------------------------|
| 核数 | 16 |
| 一级 cache (数据或指令) | private, 4-way, 32KB each |
| 二级 cache | private, 8-way, 512KB each |
| Cache 一致性协议 | 分布式目录 MOESI 协议 |
| 网络拓扑 | 4×4 Torus, 4 个虚拟网络 (VN), 1VC/VN |

由于 dateline 的频率与气泡设计不同，全系统模拟没有评测 dateline 的性能。实验假设处理器频率是网络频率的 2 倍。Cache 行包括 64 字节，网络切片大小是 16 字节，因此，长报文和短报文分别包含 5 个和 1 个切片。实验使用分布式目录 MOESI 协议，其需要 4 个虚拟网络以避免协议层死锁，每个虚拟网络的缓存数目是 10 个。PARSEC 应用程序采用 *simsmall* 输入集，全系统性能指标是程序运行时间，表5.2给出了全系统模拟的配置参数。

5.6 一维 Torus 网络 (Ring 网络) 性能测评

5.6.1 性能

实验评估从一个 8 节点的 ring 网络开始。图5.12给出了 4 种合成流量模式的性能。尽管 FBFC-C 的报文注入所需缓存单元比 FBFC-L 少 1 个，但是实验使用了 10 个缓存单元，因此，这个优势的影响很少，导致 FBFC-L 与 FBFC-C 的性能基本相当。FBFC (包括 FBFC-L 和 FBFC-C) 的性能显著优于 LBS 和 CBS。LBS 和 CBS 需要将短报文视为长报文，这个局限性严重限制了它们的性能：FBFC-C 的短报文发送只需占用一个缓存单元，而 LBS 和 CBS 则需占用 5 个缓存单元。在这 4 种模式下，FBFC-C 的饱和吞吐率比 LBS 和 CBS 平均高 73.5% 和 33.9%。LBS 的报文注入比 CBS 需要更多的缓存资源，进一步制约了 LBS 的性能；与 LBS 相比，CBS 平均获得了 29.6% 的性能提升。

Dateline 以 *flits/node/cycle* 表示的性能比 LBS 和 CBS 高。但是根据表5.1，dateline 的频率比气泡设计低 30%，因此，dateline 以 *second* 表示的零负载延迟要比 LBS 和 CBS 高 30%，同时 dateline 以 *flits/node/second* 表示的饱和吞吐率也比 CBS 低。比如说，虽然 dateline 在 uniform random 模式下以 *flits/node/cycle* 表示的饱和吞吐率比 CBS 高 20.3%，但是其以 *flits/node/second* 表示的饱和吞吐率比 CBS 低 7.5%。除此之外，dateline 的另外一个缺点是需要将缓存资源划分到 2 条虚通道上，降低了虚通道深度，较浅的虚通道增加了被阻塞报文分布的节点数。当报文在一个维度上传输时，dateline 固定了报文所使用的虚通道类型。这两个因

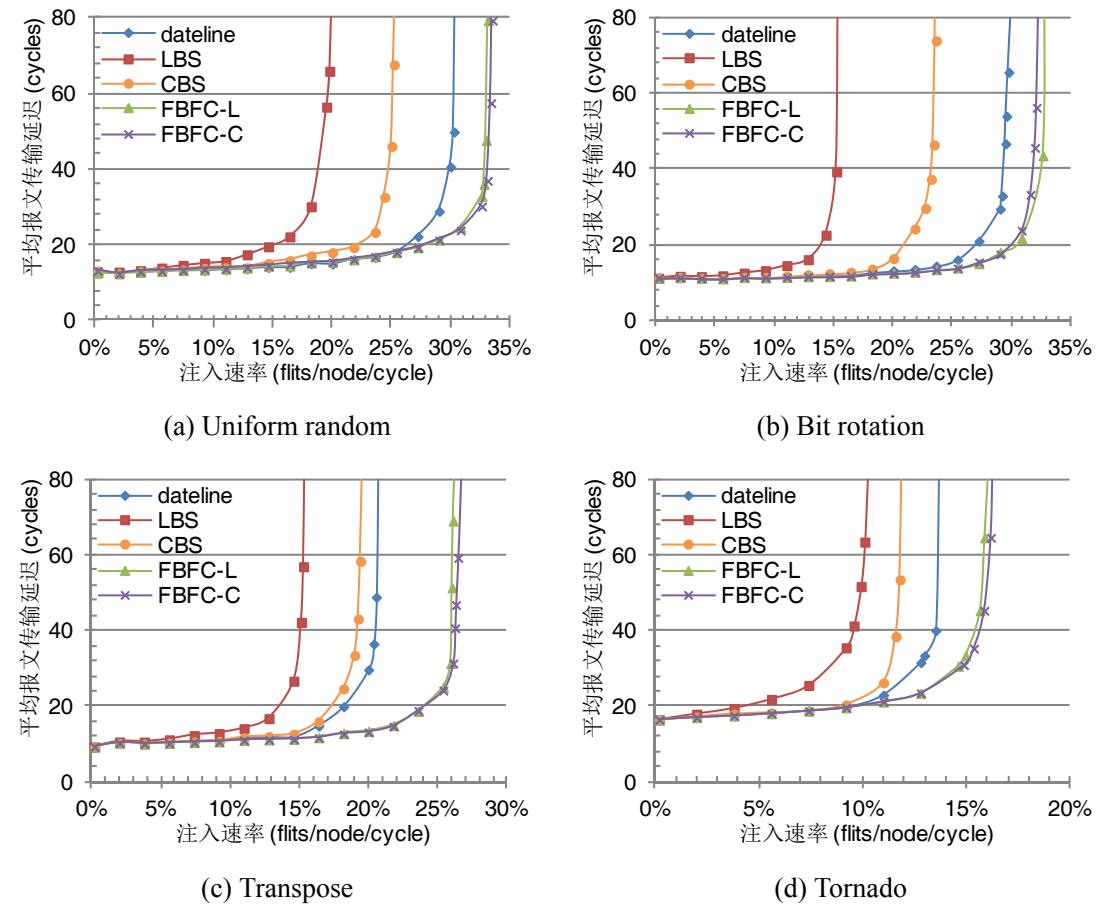


图 5.12 八节点 ring 网络上的性能

素增加了链式阻塞对性能的影响^[214]，导致 dateline 在报文发送过程中存在一些局限。然而，dateline 允许报文在下游虚通道中只有一个空闲缓存时就开始注入（或转维），它在处理报文注入（或转维）时优于 FBFC。因此，dateline 与 FBFC 之间的性能趋势取决于流量模式的报文平均注入次数（包括转维次数）和报文平均跳数。在这 4 种模式下，即使以 $flits/node/cycle$ 表示性能，FBFC-C 的性能依然优于 dateline。Ring 网络上没有转维过程，所有模式的平均注入次数都小于等于 1，隐藏了 dateline 的报文注入优势对性能的影响。与 dateline 相比，FBFC-C 获得的最大性能提升分别是在 transpose 和 tornado 模式下的 29.2% 和 18.8%。Transpose 模式的平均注入次数是 0.75（该模式的一些报文是节点发送给自己的，这些报文不需要注入到 ring 中），平均跳数是 2.25；tornado 模式的平均注入次数和平均跳数分别是 1 和 4。这两种模式揭露了 dateline 在报文发送上的局限性。

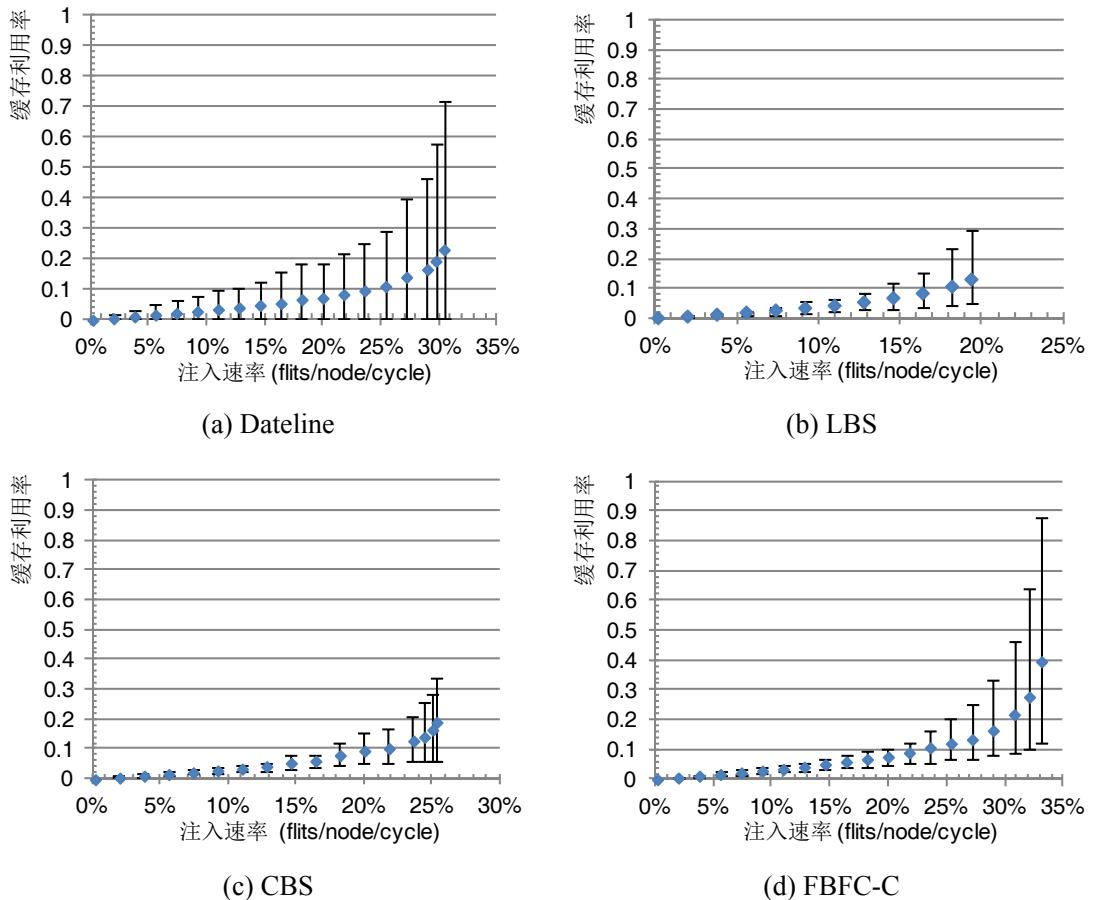


图 5.13 Uniform random 流量模式下的缓存利用率

5.6.2 缓存利用率

为了深入理解上述性能趋势，图5.13测试了网络虚通道在模拟器稳定状态下的平均利用率，图中的误差线表示最高和最低利用率，FBFC-L 与 FBFC-C 类似，因此图5.13没有给出它的结果。缓存可以提高网络吞吐率，更高的缓存利用率会带来更好的性能。随着负载的增加，缓存利用率及其变化幅度都会上升。LBS 和 CBS 将短报文视为长报文，严重限制了它们对缓存资源的有效利用。当网络饱和时，LBS 和 CBS 的平均利用率分别是 13.0% 和 19.2%。LBS 的报文注入需要更多的缓存资源，因此其利用率低于 CBS。Dateline 设计中至少有一条虚通道不会被使用，因此其最低缓存利用率总是 0。以图5.2为例，由于所有越过 ‘dateline’ 的报文都使用 VC_{10} , VC_{00} 不会被使用。这一现象和链式阻塞结合在一起限制了 dateline 的缓存利用率。当网络饱和时，dateline 的平均利用率和最大利用率分别是 23.2% 和 71.8%。FBFC-C 在网络饱和时的平均利用率和最大利用率分别是 39.5% 和 89.8%。网络出现饱和是由于某些资源，如利用率为 89.8% 的虚通道，出现了饱和^[163]。表5.3给出网络饱和时的虚通道利用率分布情况，LBS 的大部分

表 5.3 网络饱和时的缓存利用率分布

| | [0.0, 0.2] | (0.2, 0.4] | (0.4, 0.6] | (0.6, 0.8] | (0.8, 1.0] |
|----------|------------|------------|------------|------------|------------|
| Dateline | 41.7% | 33.3% | 12.5% | 12.5% | 0% |
| LBS | 66.7% | 33.3% | 0% | 0% | 0% |
| CBS | 33.3% | 66.7% | 0% | 0% | 0% |
| FBFC-L | 33.3% | 33.3% | 12.5% | 8.3% | 12.5% |
| FBFC-C | 33.3% | 16.7% | 25.0% | 12.5% | 12.5% |

虚通道的利用率低于 0.2， dateline 的链式阻塞在高负载情况下更为严重，导致其大部分虚通道的利用率低于 0.4， FBFC-C 和 FBFC-L 都有一些虚通道的利用率高于 0.8。

5.6.3 短报文和长报文的传输延迟

FBFC-C 和 FBFC-L 的长报文注入比短报文注入需要更多的缓存资源。图5.14分析了气泡设计中长报文和短报文的传输延迟，延迟包括三部分：*InjVC* 表示报文在注入虚通道 (Injection VC) 上的延迟；*NI* 是报文在网络接口 (Network Interface) 队列中的延迟，小到中规模网络中的拥塞会很快传输到网络接口上，此时报文会在这个队列中等待^[163, 214]；*Network* 是所有其它的延迟。短报文和长报文在 LBS 和 CBS 中被同等对待，它们在注入虚通道和网络中的延迟基本相同。由于长报文比短报文多 4 个切片，长报文的网络接口延迟比短报文多 4 个时钟周期。在低到中等负载（直到 20% 注入率）下，FBFC-C 中的长报文和短报文在注入虚通道中的延迟基本相同；在更高负载时，它们的差异会增加。然而，即使网络饱和时，长报文和短报文在 uniform random 和 bit rotation 模式下的延迟差异也只有 3.4 和 3.2 个时钟周期，也就是说 FBFC-C 并没有牺牲长报文的性能。FBFC-C 能够较快地发送短报文，也有利于长报文的发送。事实上，与 FBFC-C 相比，LBS 和 CBS 牺牲了短报文的性能。比如说，在 20% 注入率的 uniform random 模式下，LBS 和 CBS 的短报文分别在注入虚通道上消耗了 11.7 和 5.4 个时钟周期，而 FBFC-C 的短报文只需要 4.3 个时钟周期。

5.7 二维 Torus 网络性能测评

5.7.1 4x4 Torus 网络性能

本节开始关注二维 torus 网络的性能，图5.15给出了 4×4 torus 网络上的性能，图5.15(a)中的高误差线和低误差线分别是长报文和短报文的平均传输延迟。由于 FBFC-C 和 FBFC-L 的报文注入差异相对较少，它们的性能基本相似。FBFC-C 的

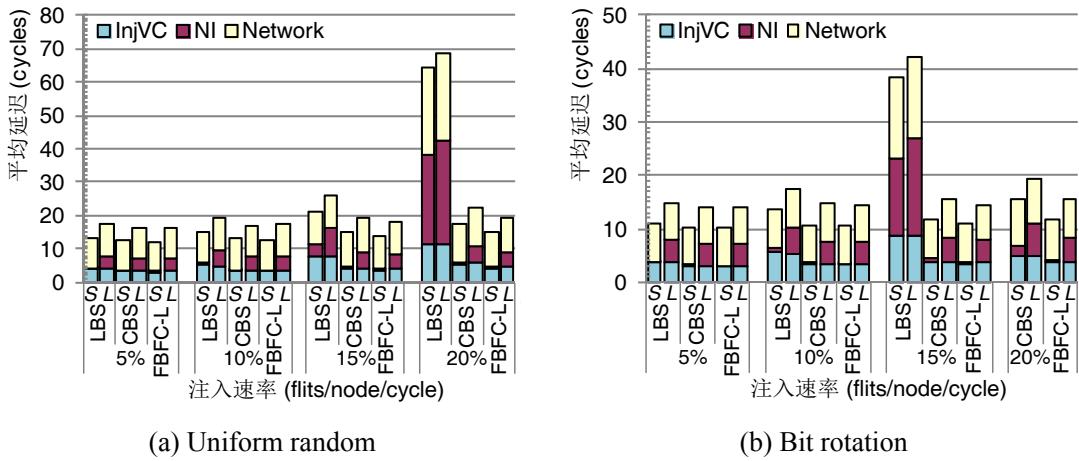


图 5.14 短报文 (图中 S) 和长报文 (图中 L) 的传输延迟

高缓存利用率使得其性能显著优于 LBS 和 CBS；相比于 LBS 和 CBS，FBFC-C 的平均性能提升分别是 92.8% 和 34.2%。CBS 的报文注入所需缓存资源低于 LBS，因此其性能优于 LBS；CBS 相对于 LBS 的平均性能提升是 45.7%，最大性能提升是 transpose 模式中的 100%。Transpose 模式的许多报文从同一行发送到同一列，这些报文需要在相同路由器的相同输出端口上转维，从而造成了这些端口的拥塞；CBS 相对较低的报文转维缓存数需求带来了上述性能提升。

由于 FBFC 对转维报文的处理与注入报文一样，它和 dateline 在二维 torus 中的性能趋势与 ring 中不同。在 uniform random 和 transpose 模式下，dateline 的性能与 FBFC-C 相当；这两种模式的平均注入次数都是 1.5，它们的平均跳数都是 3，此时报文注入和转维对性能的影响增加了，因此 dateline 的报文注入优势带来了性能提升。在 hotspot 模式下，dateline 的性能比 FBFC-C 高 5.7%；该模式的许多来自不同行的报文需要转维到 4 个热点节点所在的列，FBFC-C 的长报文注入局限性降低了性能。在 bit rotation 模式下，FBFC-C 的性能高于 dateline 6.4%；该模式的所有报文转维都是申请不同输出端口，降低了这些端口上的拥塞，由此减轻了 FBFC-C 长报文注入局限性对性能的影响。注意这里的性能比较并没有考虑路由器的频率，但是如表5.1所示，dateline 的频率要比 FBFC 低 30%。

从图5.15(a)可以看出，LBS 和 CBS 的长报文和短报文的延迟差异在所有负载情况下基本相同，长报文在网络接口上多消耗 4 个时钟周期，导致其延迟比短报文约高 4 个时钟周期。当网络饱和时，dateline 的长报文延迟比短报文高 6.2 个时钟周期。FBFC-L 的延迟差异更为明显：转维过程使得二维 torus 中的长短报文延迟差异比 ring 中约多 3 个时钟周期，在网络饱和时，FBFC-L 的长报文延迟比短报文高 9.8 个时钟周期。饱和之后的网络状态是一个值得关注的问题。图5.16给出了在 uniform random 和 hotspot 模式下，注入率一直上升到 1.0 flits/node/cycle

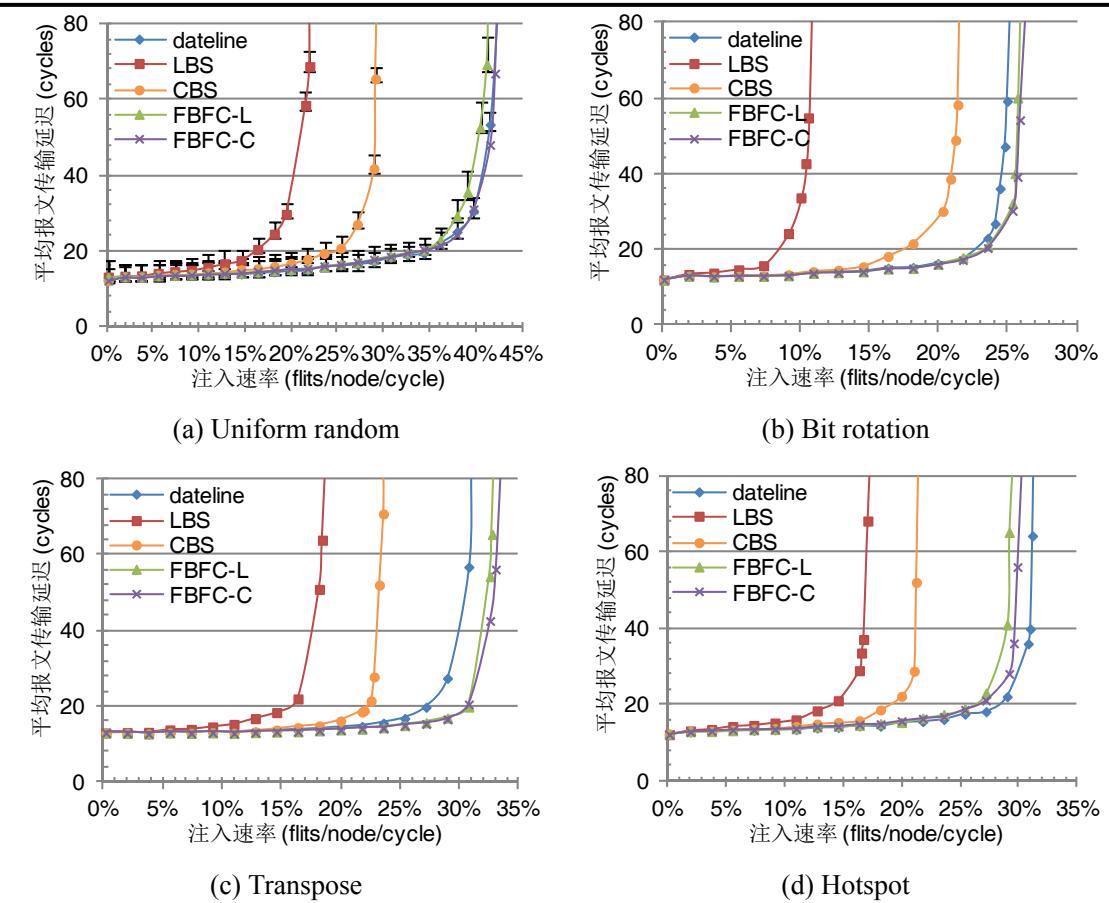


图 5.15 4×4 torus 网络上的性能

时的网络吞吐率，其它流量模式的趋势与之类似。维序路由保证已注入报文一定可以顺利转发，因此所有设计在饱和之后都维持自己的性能。在自适应路由算法或死锁恢复设计中，由于逃逸通道的排出率低于报文的注入率，它们在网络饱和后的性能可能会出现下降^[59, 82]。此时可以采用源节流（source throttling）技术来维持它们的性能^[82]。图5.16(a)和图5.16(b)中的稳定吞吐率略高于图5.15(a)和图5.15(d)；图5.16(a)和图5.16(b)中的平均延迟已经达到了几千个时钟周期。

5.7.2 单切片比例敏感性分析

如第5.4.2节所讨论的，LBS 和 CBS 设计中采用了超前信元回复优化，其重叠了输入报文使用虚通道和输出报文释放虚通道的过程，因此报文可以在下游虚通道只有 1 个空闲缓存单元时开始输入。该优化使得 CBS 的长报文注入过程优于 FBFC：CBS 允许报文在下游虚通道只有 1 个缓存单元时开始注入，而 FBFC-C 的长报文注入需要 5 个普通缓存单元。图5.17分析了各种设计对单切片报文比例的敏感性。LBS 和 CBS 将短报文视为长报文，更多的长报文能成比例提升它们的性

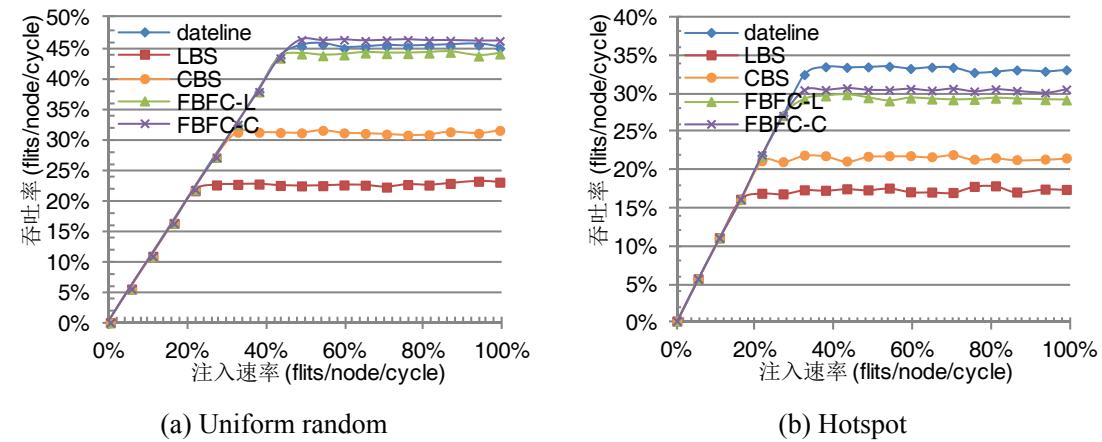


图 5.16 网络达到饱和之后的性能

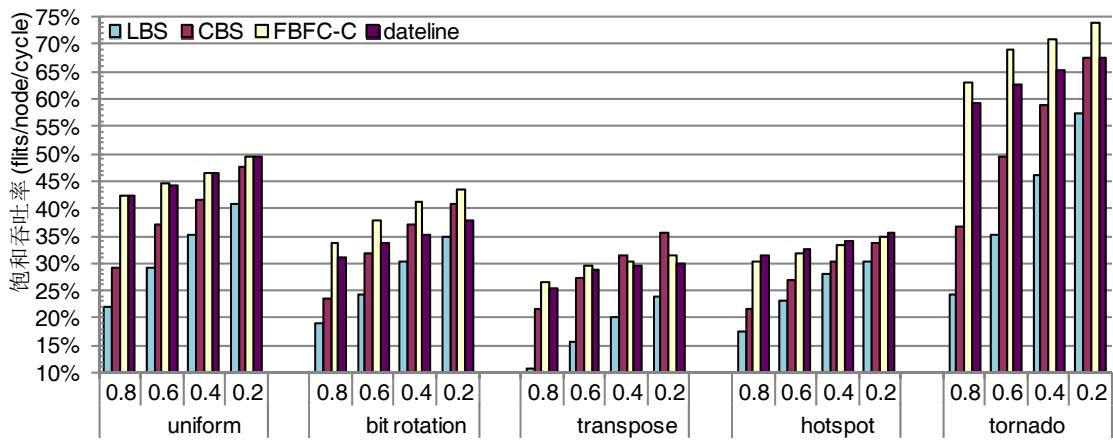


图 5.17 不同单切片报文比例下的饱和吞吐率

能，它们的性能随着单切片报文比例的降低而线性增长。由于 FBFC-C 和 dateline 试图为整个报文维持交叉开关分配的结果，更多的长报文能降低交叉开关分配的冲突，因此，它们的性能随着单切片报文比例的降低而轻微上升。

CBS 相对于 LBS 能更高效地使用缓存资源，它在所有场景下的性能都优于 LBS。随着单切片报文比例的下降，FBFC-C 相对于 CBS 的性能提升逐渐减少。但是由于 CBS 的性能受限于将短报文视为长报文，即使在 0.2 的单切片比例下，FBFC-C 在 uniform random、bit rotation、hotspot 和 tornado 模式下依然优于 CBS。在 0.2 单切片比例的 tornado 模式中，FBFC-C 的性能比 CBS 高 10.1%。在 4×4 torus 网络中，tornado 模式从节点 (i, j) 发送报文到节点 $((i + 1)\%4, (j + 1)\%4)$ ，该模式每条网络链路只传输一个通信对的报文，网络不存在拥塞，缓解 FBFC-C 的长报文注入局限性对性能的影响。Transpose 模式的结果有所不同：在 0.4 和 0.2 的单切片报文比例下，CBS 的性能优于 FBFC-C；该流量模式的转维路由器存在拥塞，因此 CBS 的注入优势能带来性能提升。Dateline 和 FBFC-C 之间的性能趋

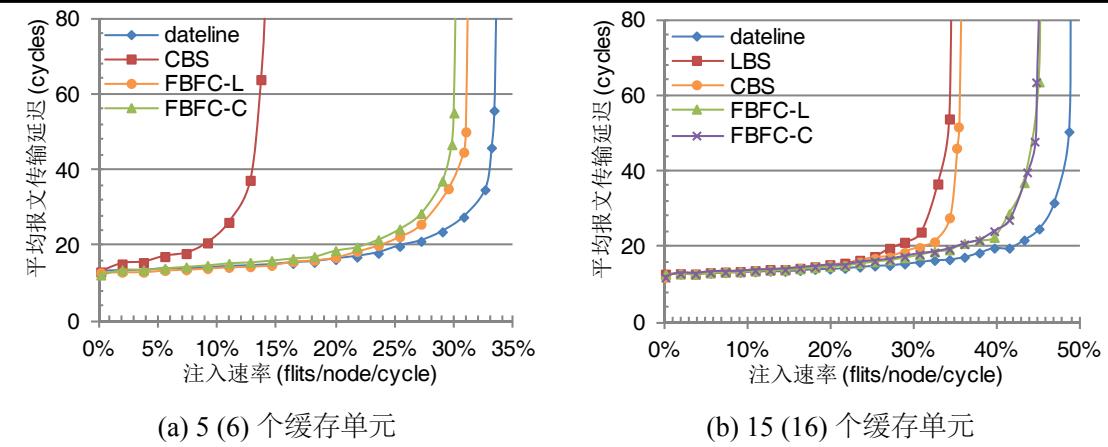


图 5.18 Uniform random 模式在其它缓存配置下的性能

势与图5.15类似：除 hotspot 模式外，FBFC-C 的性能即使以 $flits/node/cycle$ 表示依然不逊于 dateline。上述实验假设切片是 128 位宽度的，本节同时也研究了切片是 64 位宽度时的性能，此时长报文和短报文分别包含 9 个和 1 个切片。随着长报文长度的增加，LBS 和 CBS 将短报文视为长报文会更进一步限制它们的性能，同时，FBFC-C 的长报文注入也需要更多的空闲缓存数目。这两方面因素使得 64 位切片位宽下的性能趋势与图5.17类似。

5.7.3 缓存数量敏感性分析

本节分析缓存数量对性能的影响，在图5.18(a)中，CBS、FBFC-C 和 FBFC-L 配置了能够保证它们正确运行的最少缓存数目，即 5 个、5 个和 6 个；dateline 使用 6 个缓存单元 (slots)，并将它们划分到两条虚通道内；由于 LBS 至少需要 10 个缓存单元，图5.18(a)没有给出它的性能。在图5.18(b)中，所有气泡设计都使用一条 15 缓存单元的虚通道，dateline 使用两条 8 缓存单元的虚通道。实验使用 uniform random 模式。尽管 FBFC-C 的长报文注入局限性在 5 slots/VC 时更为严重，dateline 以 $flits/node/cycle$ 表示的性能依然只比 FBFC-C 高 7.6%。此时，dateline 较浅的虚通道 (3 slots/VC) 无法覆盖信元往返延迟，使得链路层流控成为性能瓶颈。提高缓存利用率在缓存越少时越重要，因此，FBFC-C 相对于 CBS 的性能提升在缓存越少时越明显：其性能提升在 15 slots/VC、10 slots/VC 和 5 slots/VC 时分别是 26.6%、41.4% 和 121.8%。比较图5.15(a)和图5.18(a)，CBS 在 10 slots/VC 时的性能与 FBFC-C 在 5 slots/VC 时的性能基本相当，即 FBFC-C 只需使用 CBS 一半的缓存资源就可获得了与 CBS 相似的性能。此外，LBS 在 15 slots/VC 时的性能只比 FBFC-C 在 5 slots/VC 时的性能高 5.2%。

更多的缓存资源能缓解 LBS 的高注入缓存需求对性能的影响，因此，LBS 的

性能在 15 slots/VC 时与 CBS 相似。本节还测试了缓存数量进一步增长时的性能。当网络拥有丰富的缓存时，大量缓存单元将处于空闲状态，导致不同气泡设计的性能趋于相似，它们的性能收敛点取决于具体的流量模式。例如，由于 transpose 模式加剧了转维路由器的拥塞，LBS 至少需要配置 30 slots/VC 才能赶上 CBS。在 uniform random 模式下，LBS 和 CBS 赶上 FBFC-C 的性能收敛点是 50 slots/VC。这些气泡设计的性能只有在配置大量的缓存资源时才会相似，这也使得具有较高缓存利用率的 CBS 和 FBFC-C 成为合理缓存配置下的优胜者。

5.7.4 8×8 Torus 网络可扩展性分析

图5.19分析了不同设计在 8×8 torus 网络上的可扩展性。该图使用了两种缓存配置：第一种为每个设计配置了 10 个缓存单元，第二种与图5.18(a)相同，其为每种设计配置了 5 个或 6 个缓存单元。随着网络规模的增大，每种流量模式的平均跳数都会增加，因此，dateline 的报文发送局限性对性能的影响在越大规模网络中越明显。同时， 8×8 torus 中流量模式的平均注入次数与 4×4 torus 基本相当。这两方面因素在一定程度上缓解了 FBFC-C 的长报文注入局限性对性能的影响。当配置 10 个缓存单元时，FBFC-C 即使以 *flits/node/cycle* 表示的性能依然在所有模式中都优于 dateline。FBFC-C 获得了最大的性能提升分别是在 bit rotation 和 tornado 模式下的 26.5% 和 18.0%。Bit rotation 模式的转维路由器的拥塞程度较轻，从而缓解了 FBFC-C 的长报文注入局限性对性能的影响。Tornado 模式的平均跳数是 7 跳，其恶化了 dateline 的报文发送局限性对性能的影响。当配置 6 个缓存单元时，FBFC-L 的性能在 4 种模式下优于 dateline，FBFC-L 在 bit rotation 模式下获得了最大 18.4% 的性能提升。缓存使用压力随着网络规模的增大而上升，因此，LBS 和 CBS 的低缓存利用率对性能的影响在越大规模网络中越明显。在 8×8 torus 的 uniform random 模式下，FBFC-C 的性能在配置 10 个缓存单元时比 CBS 高 82.5%，而这个性能提升在 4×4 torus 中是 41.4%（图5.15(a))。当配置 10 个缓存单元时，FBFC-C 在 8 种模式下的性能平均比 LBS 和 CBS 高 107.2% 和 40.1%；当配置 5 个缓存单元时，FBFC-C 相对于 CBS 的平均性能提升是 78.7%。

5.7.5 饿死现象对性能的影响

本节分析 4×4 torus 中的饿死现象，由于较少的缓存数量有利于观察饿死对性能的影响，实验采用与图5.18(a)相同的缓存配置（LBS 使用 10 个缓存单元），流量模式是 uniform random，更大规模网络或其它流量模式具有类似的趋势。LBS 和 FBFC-L 的饿死本质上是相同的，图5.20(a)给出了它们在三个饿死阈值下的性能。一个较小阈值导致频繁置位 starve 信号（图5.9）禁止报文注入，从而影响了

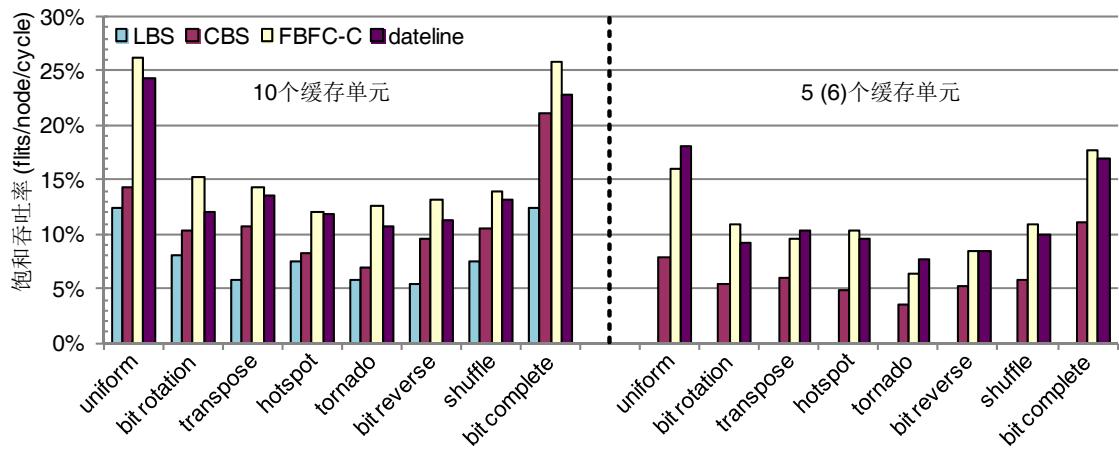


图 5.19 8x8 torus 网络中的性能

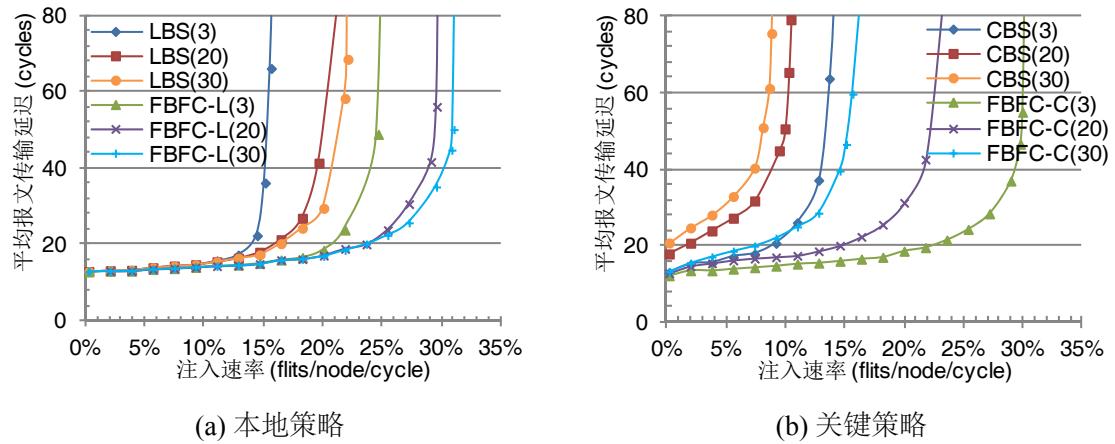


图 5.20 不同饿死阈值下的性能

网络整体性能，LBS 和 FBFC-L 的性能在阈值为 3 个时钟周期时最差。图 5.21 给出了饿死阈值从 3 增长到 50 时的饱和吞吐率。随着阈值的增加，LBS 和 FBFC-L 的性能逐渐上升，但是阈值过大时会牺牲发生饿死的节点，因此它们的性能在阈值大于 30 时基本保持稳定。本章将 LBS 和 FBFC-L 的饿死阈值设置为 30。

关键气泡停滞会给 CBS 和 FBFC-C 带来饿死，此外，FBFC-C 还存在与 FBFC-L 相同类型的饿死，FBFC-C 使用两个阈值分别处理这两种类型的饿死。这里将 FBFC-C 处理与 FBFC-L 相同类型饿死的阈值设置为 30，并调整另一个阈值的大小。如图 5.20(b) 所示，CBS 和 FBFC-C 的性能在阈值越小时越好。主动移动关键气泡没有限制节点注入报文，因此，即便很高的饿死误检测率也不会导致总体性能下降。相反，在阈值较大时，发生饿死的节点长时间无法注入报文会导致性能的下降。例如，当饿死阈值为 20 时，被关键气泡阻止注入的节点只有在 20 个时钟周期后才会尝试移动这个关键气泡，这种缓慢处理方式不仅限制了饱和吞吐率，而且增加了零延迟负载（图 5.20(b)），图 5.21 也证实上述观察。由于主动移

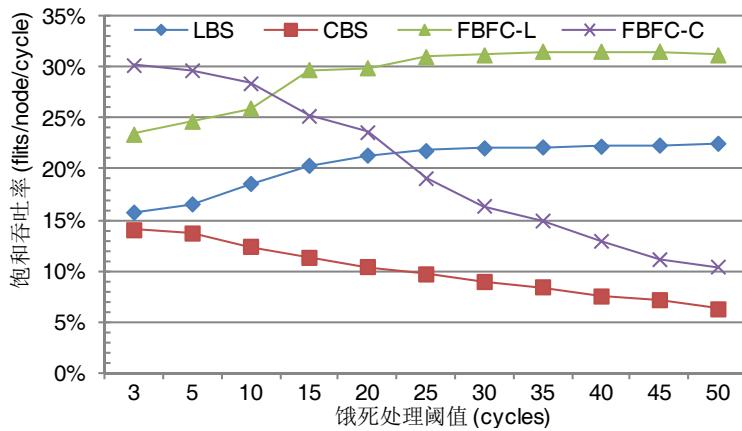


图 5.21 不同饿死阈值下的饱和吞吐率

动关键气泡需要 2 个时钟周期，本章将处理关键气泡停滞饿死的阈值设置为 3。

5.7.6 PARSEC 测试集实验结果

图 5.22 给出了各种设计在 PARSEC 测试集下相对于 LBS 的性能加速比。FBFC 能支持更高的网络吞吐率，但是全系统性能提升取决于应用程序产生的负载和流量模式，具有较高网络负载和较多猝发通信的应用程序能够从 FBFC 中获得性能提升。不同设计在 `blackscholes`、`fluidanimate` 和 `swaptions` 上性能相当，这些程序的计算过程具有很少的栅栏同步，同时它们的工作集可以放入 2 级 cache，导致它们产生的网络负载较低，因此，网络层优化几乎不会影响它们的性能。然而，其它 7 个程序的性能受到网络层优化的影响。CBS 和 FBFC（包括 FBFC-L 和 FBFC-C）相对于 LBS 都获得了一定性能提升，FBFC 在 `canneal` 上获得了最大 22.7% 的性能提升。带来这些性能提升的原因有两个：第一，这些程序的网络负载相对较重，同时它们也包含一些猝发通信，因此，支持较高网络吞吐率的设计能够带来性能提升。第二，VN0 和 VN2 传输了 70.8% 的网络负载，包括 `read request`、`write-back request`、`read response`、`write-back acknowledgement` 和 `invalidation acknowledgement` 消息，因此，FBFC 高效处理混合长度报文能带来性能提升。在所有的程序下，FBFC 和 CBS 相对于 LBS 的平均性能提升分别是 13.0% 和 7.5%。由于所配置的 CMP 平台对网络缓存的压力较低，全系统模拟性能提升比合成流量模式性能提升要少。使用更少的缓存资源或在单个路由器上集成多个处理核^[163] 会增加对网络缓存的压力，FBFC 在这些情况下会获得更大的性能提升。例如，FBFC-C 在配置 5 个缓存单元时相对于 CBS 的能获得了平均 9.8%、最大 20.2% 的性能提升。同时，FBFC-C 在 5 个缓存单元时的程序执行时间与 LBS 在 10 个缓存单元时的程序执行时间相当。

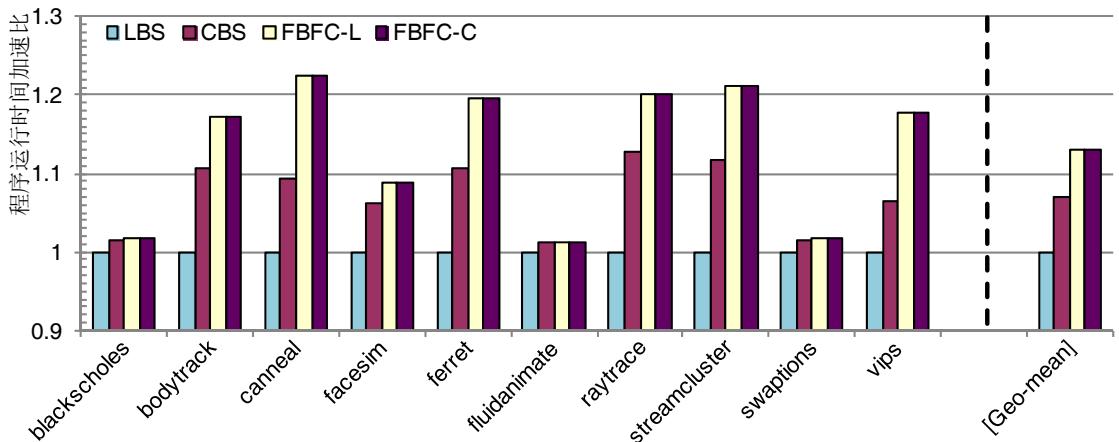


图 5.22 PARSE 测试集的全系统加速比

5.7.7 FBFC 硬件开销

如第5.4节所述，FBFC 路由器需要比对应的虚切通路由器多使用了 1 个一位 *inj* 寄存器。假设 128 位切片宽度，每条虚通道包含 10 个缓存单元，这个额外寄存器只引入 0.7% 的存储开销。FBFC 区别对待长报文和短报文，因此，VA 在访问 *inj* 之前需要先访问一个两输入多路选择器。基于延迟模型分析^[38]，该多路选择器只给 VA 段增加了 1.2% 的延迟。如表5.1所示，SA 段在多虚拟网络时是关键路径。因此，多路选择器所增加的延迟不在关键路径上。

虚切通路由器的每个报文发送只回复一个信元，它的信元处理比 FBFC 所使用虫孔路由器简单，但是信元处理不在关键路径上。虚切通路由器可以将 VA 和 SA 合并成一个流水线阶段^[82]，但是这种合并可能会增加关键路径的延迟，同时由于 VA 可以为每个虚拟网络独立设置^[44]，这种合并可能会降低性能。虫孔交换路由器也可以优化流水线设计^[38, 41]。FBFC 获得性能提升的原因是其能更为有效地利用缓存资源，因此本章研究与路由器流水线设计基本无关。

气泡设计需要一些额外的连线资源来预防饿死。在 4×4 torus 网络中，*starve* 信号中的 *router ID* 字段需要 4 位；*starve* 信号完成一个 ring 上的传输需要 4 个时钟周期，因此，只需考虑一个 8 周期窗口内的 *starve* 信号排序，即 *time stamp* 字段需要 3 位。当配置 4 个虚拟网络时，LBS 和 FBFC-L 的饿死预防边带网络的带宽是 10 位；CBS 的边带网络带宽是 4 位，其中 *N2C* 和 *C2N* 信号各需要一位，另外两位用于编码虚拟网络号；由于 FBFC-C 需要处理两种类型的饿死，其边带网络带宽是 14 位。假设 128 位的切片位宽，这些额外的连线开销依次是 7.81%、3.125% 和 10.94%。由于连线资源并不是片上网络的限制因素，这些额外开销是可以接受的。尽管饿死预防需要增加一些额外的逻辑，但是由于饿死预防使用边带网络实现，这些逻辑不会影响关键路径延迟。

气泡设计要求虚通道深度需要满足一些最低要求：FBFC-C 要求虚通道深度至少要等于最长报文的切片数，该要求与已有的最好设计（CBS）相同。FBFC-L 的虚通道深度最低要求比 FBFC-C 多一个缓存单元。如下两个因素使得这些最低虚通道深度要求很容易得到满足：第一，在 cache 一致性片上网络中，最大报文切片数是由编码一个完整的 cache 行所决定的。丰富的片上连线资源增加了切片的宽度，在 128 位^[45, 49] 或 256 位^[33] 切片宽度下，编码一个 cache 行并不需要很多切片。第二，为了防止链路层流控成为系统瓶颈，虫孔路由器要求虚通道深度最好能覆盖信元往返延迟。

缓存是片上网络中消耗功耗和面积最严重的模块^[45, 49]。在合成流量模式下，FBFC 使用 CBS 的一半缓存资源能够获得与其相当的性能；同时，在 PARSEC 测试集中，FBFC 在使用 LBS 的一半缓存资源时，能够获得与其类似的性能。这带来了很大的好处，片上缓存数减半可以降低 34% 的功耗和 30% 的面积^[49]。

5.8 进一步讨论

切片气泡流控是已有报文气泡流控的重要扩展，除了解决了这些报文气泡设计的缺陷，并获得了较大的性能提升外，切片气泡流控还有许多其它优点。首先，已有的虚切通网络气泡设计要求虚通道深度必须是报文长度的整数倍，而基于虫孔交换提出的切片气泡流控没有这个要求，从而提供了更高的设计灵活性。第二，基于‘剩下一个空闲缓存单元’的思想，可以得到许多其它设计。例如，与动态报文分段（dynamic packet fragmentation）机制^[45] 相结合，可以给出如下设计：报文在下游虚通道只有一个空闲缓存单元时即开始注入，当某个需要注入的切片会占用关键切片气泡时，报文停止该切片的注入，并且将这个切片动态更新为头切片。这种设计允许虚通道深度少于报文长度，同时也解决了 FBFC-C 和 FBFC-L 的长报文注入局限性。但是该设计需要在目标节点将属于同一报文的多个切片进行重组^[45]。Wormhole cut-through 流控机制^[80] 允许不同报文在切片粒度上混合存储于同一条虚通道内，FBFC 可以与其结合得到类似于动态报文分段的设计。包括 FBFC 在内的气泡设计只使用了一条虚通道，因此它们面临头报文阻塞的问题。在多条虚通道（或虚拟网络）之间动态共享缓存^[49, 216] 可以减轻这种阻塞，同时也能缓解 FBFC-L 和 FBFC-C 的长报文注入局限性对性能的影响。将来的工作中会深入研究这些设计。

5.9 相关研究

当前在片外网络^[225]、工业界产品^[17, 152] 和片上网络^[30, 31, 153] 已经存在很多对 ring 和层次 ring 的研究和应用。Token ring、slotted ring 和 register insertion ring 是

三种主要的模式^[225]，它们避免死锁的原理与无缓存网络（bufferless network）或链路交换网络类似，即需要保证注入的报文在网络中不会阻塞^[31, 153]，因此，它们需要采用比较复杂的端到端的流控机制^[30]或中央控制单元^[17]，这些设计可能会面临可扩展性较差或吞吐率较低的问题^[30, 153, 226]。本章主要研究缓存网络，除实验比较的 dateline^[163]、LBS^[217] 和 CBS^[82] 外，还存在其它处理 torus 网络死锁的研究。在单周期路由器中，优先级仲裁能避免单切片报文网络的死锁^[153]。预防流控机制（prevention flow control）^[83] 综合使用了优先级仲裁和预防槽循环机制来避免虚切通网络死锁，但是该设计在传输混合长度报文时存在死锁。转向模型在虫孔交换网络中使用一条虚通道只能支持 2 维 torus 的非最短路由^[159]，同时其不能在 ring 中实现死锁避免。Shin 和 Kim 针对虫孔交换网络提出了一种死锁恢复策略，该策略只使用一条虚通道^[34]，它允许网络中先出现死锁，然后采用一些机制从死锁中进行恢复。由于切片气泡流控从设计上消除了死锁的出现，因此与他们的研究有很大不同。就我们所知，切片气泡流控是第一种能够在虫孔交换网络中只使用一条虚通道实现死锁避免，并支持最短路由的设计。

报文气泡流控广泛应用于商业产品^[155]、新型片上网络路由器^[42] 和自适应路由算法^[82, 222] 上。由于切片气泡流控的性能优于报文气泡流控，它也可以被应用于这些领域。由于某些 cache 一致性协议要求报文按序传输^[7, 163]，本章使用能提供按序报文传输的维序路由算法。Singh 等使用非最短路由实现 torus 网络的全局负载均衡^[59]，他们的死锁避免是基于 dateline 实现的，因此本章的研究与他们的研究互补。切片气泡流控利用了 cache 一致性网络传输的大部分是短报文这个观察，最近有很多其它研究也是基于这个观察，包括 whole packet forwarding^[150]、packet chaining^[47] 和 NoX 路由器^[48]。配置更多的虚拟网络可以使每个虚拟网络上传输相同长度的报文。例如，Alpha 21364^[210] 使用了 7 个虚拟网络，使得每个虚拟网络传输等长报文。但是更多虚拟网络会带来更高硬件开销，设计者一般希望使用较少的虚拟网络数。DASH^[218]、Origin 2000^[219] 和 Piranha 机器^[220] 都采用协议层死锁恢复机制来减少一个虚拟网络：它们使用了 2 个虚拟网络来实现 3 跳目录 cache 一致性协议，这些机器上的所有虚拟网络都传输混合长度报文。

5.10 本章小结

本章主要研究在 torus 网络保持无死锁的前提下提高缓存的利用率。本章分析了已有设计在这方面的局限，并进一步提出了切片气泡流控 torus 网络死锁避免理论。切片气泡流控通过在每个 ring 上维持一个空闲缓存单元实现死锁避免。切片气泡流控只需使用一条虚通道，因此其路由器频率比 dateline 高 30%。与之前的报文气泡设计不同，切片气泡流控无需将短报文视为长报文，因此提高了缓

存利用率。基于切片气泡流控理论，本章给出了两种实现：本地切片气泡策略（FBFC-L）和关键气泡策略（FBFC-C）。FBFC-L 的最少缓存需求比已有的最好设计多一个缓存单元，FBFC-C 的缓存需求与已有的最好设计相同。在 PARSEC 测试集中，FBFC 相对于 LBS 的平均性能提升是 13.0%，其最大性能提升是 22.7%。随着缓存数量的减少，FBFC 的性能优势更为明显。在合成流量模式下，FBFC 使用 CBS 的一半缓存资源时能够获得与其类似的性能。同时在 PARSEC 测试集下，FBFC 使用 LBS 的一半缓存资源时能够获得与其相当的性能。

第六章 高效支持 cache 一致性协议归约和多播通信的技术

聚合通信 (collective communication) 对许多体系结构和编程模式的性能和正确性具有关键影响。为了防止这些重要的聚合通信成为整个系统的瓶颈，必须要对它们提供硬件支持。在片上网络中，对多播 (multicast) 或一对多通信提供硬件支持已经取得了相当的网络吞吐率提升，同时也获得了一定的功耗下降。本章研究对归约 (reduction) 或多对一通信提供硬件支持。本章选取一个归约通信案例进行具体讨论：目录 cache 一致性协议中一条 cache 行被提升到修改 (modified) 状态时需要收集的 acknowledgement (ACK) 消息。

本章工作主要包括两个方面：1、提出了一种消息组合框架，其能有效支持 ACK 消息的归约操作；2、提出了一种均衡自适应多播路由算法 (Balanced, Adaptive Multicast, BAM)。所提出消息组合框架可以与多种多播路由算法一起使用。通过将传输中的 ACK 消息组合，该消息组合框架不仅能在低到中等负载下降低报文传输延迟，同时也能提升网络饱和吞吐率，这个框架结构只需要较低的额外硬件开销。同时，本章所提出的 BAM 多播路由算法能够均衡地利用缓存资源，实现一定的网络吞吐率提升。

6.1 引言

为有效挖掘众核结构的性能，片上网络必须高效支持一系列通信模式。聚合通信对应用程序的性能有至关重要的影响，因此，许多巨型机都实现了专用或优化的聚合和栅栏通信网络，这些巨型机包括 NYU Ultracomputer^[227]、CM-5^[228]、Cray T3D^[229]、Blue Gene/L^[230] 和天河 1A^[231] 等。类似的，众核平台也需要对聚合通信提供硬件支持，但是由于面积和功耗的限制，众核平台无法承受专用硬件聚合通信网络^[232]。本章研究如何在已有的片上网络中集成对聚合通信的支持。

许多并行程序和编程模式需要使用包括广播 (broadcast)、多播 (multicast) 或归约 (reduction) 操作在内的聚合通信。例如，目录 cache 一致性协议需要发送广播消息作废 (invalidate) 分布在多个节点上的 cache 行共享^[9]；令牌 cache 一致性协议需要使用多播通信收集可用令牌^[233]；栅栏同步也需要使用归约和多播通信^[234, 235]。这些聚合通信对众核平台的性能有很重要的影响，当不采用任何专门硬件支持多播通信时，即使 1% 的多播报文也会导致网络吞吐率急剧下降^[69]。因此，人们提出了大量多播路由算法以提升片上网络的性能^[65, 68–72, 74, 236]。

多播报文根据其携带消息的类型触发目标节点执行相应的操作，例如作废 cache 行^[9] 或统计令牌的数目^[233]。为了将这些操作的状态通知多播报文的源节

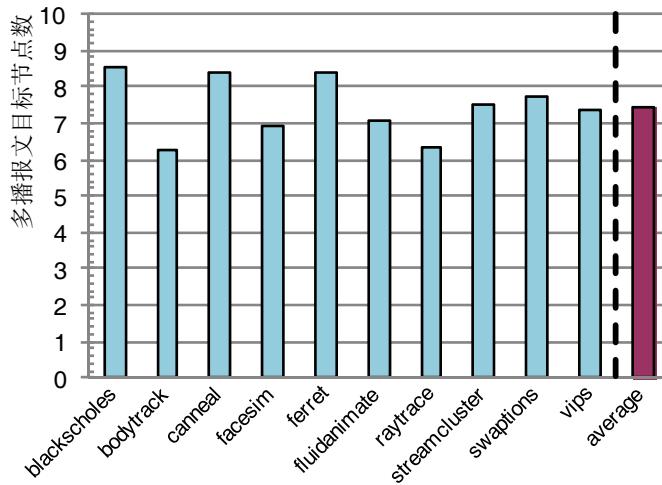


图 6.1 PARSEC 测试集多播报文的平均目标节点数目

点，每个目标节点都会回复一个消息，这种多对一的通信模式被称为归约通信^[237]。在 16 核的 CMP 平台上，本节测试了 PARSEC 应用程序^[201]中 cache 行作废消息的目标节点数。如图6.1所示，每个多播报文平均需要发送给 7.44 个目标节点，也就是说每个 cache 行作废消息平均会产生 7.44 个 ACK 消息。已有的片上网络多播路由设计^[65, 68–72, 74, 236]假设这些 ACK 消息通过多个单播报文传输到同一个目标节点上，这会给网络带来冗余的报文传输，同时也会造成瞬态网络热点。一种高效、可扩展的片上网络设计应该消除这些冗余的报文传输操作。

Cache 一致性的多播 - 归约事务通常是由处理核存储操作 (store) 引起的。在乱序执行计算核上，store 操作不处于关键路径上，但是它们会推迟后续对同一 cache 行的载入 (load) 操作。对于采用简单按序执行计算核的 CMP 平台，这些 store 操作位于关键路径上，它们对全系统的性能影响较大。只有当源节点接收到所有的回复消息时，多播 - 归约事务才算结束^[9, 73, 88, 218, 233, 238]。图6.2给出了在 4×4 mesh 网络中多播 - 归约事务的延迟，*Multicast* 表示最后一个多播目标节点接收到报文的延迟，*Transaction* 表示源节点接收到最后一个 ACK 的延迟，网络采用第6.5节介绍的 BAM+NonCom 路由，背景单播报文流量模式是 uniform random。从图中可以看出，归约报文传输占整个多播 - 归约事务 40% 的延迟，因此，需要设计高效的归约报文传输机制来提升多播 - 归约事务的性能。

Cache 一致性归约消息的一个重要特性是它们的格式比较简单，同时所携带的信息比较相似。例如，invalidation ACK 消息只携带节点的 ACK 状态信息，令牌统计回复消息只携带节点的可用令牌数目。因此，可以将这些归约消息在传输过程中可以组合，而不会丢失重要信息。消息组合可以减少冗余网络传输操作，从而优化性能。本章提出了一种低硬件开销的消息组合框架。为了简化讨论，本章使用目录 cache 一致性协议中的 invalidation ACK 消息作为一个研究案例，该框

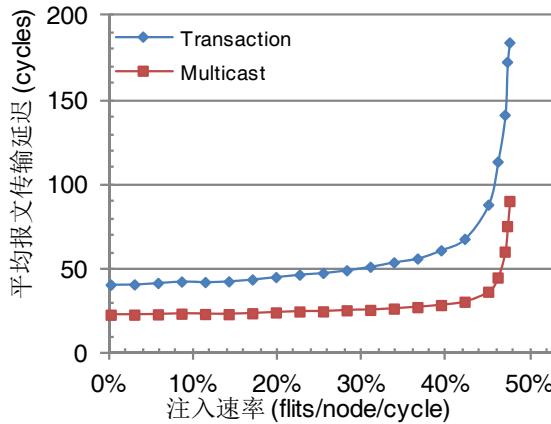


图 6.2 多播 - 归约事务延迟

架可以方便扩展到其它类型的协议上，比如令牌一致性协议^[233]。

本章所提出的消息组合框架可以与多种多播路由算法一起使用。多播报文的传输过程建立起一颗逻辑树，该消息组合框架要求 ACK 报文按照多播逻辑树的相反方向传输。路由器上设置一个消息组合表（Message Combination Table, MCT）记录多播事务期待接收的 ACK 数目和当前已经接收到的 ACK 数目。ACK 报文到达路由器时，会访问 MCT 表。如果路由器没有接收到所有期待的 ACK 报文，则会更新 MCT 表项，同时将输入的 ACK 报文丢包。如果路由器已接收到所有期待的 ACK 报文，则更新输入的 ACK 报文，并转发至逻辑树的下一节点。将 ACK 报文丢包降低了网络负载和功耗。

本章的设计目标是在网络中既存在单播报文，又存在多播报文时，提升整个网络的性能。最近，Wang 等提出了迭代划分多播路由算法（Recursive Partitioning Multicast, RPM）^[70] 采用两个虚拟网络避免多播路由死锁，造成了垂直方向和水平方向的缓存资源不均衡，影响了系统的性能。鉴于此，本章提出了均衡自适应多播路由（Balanced, Adaptive Multicast, BAM）算法。BAM 不需要采用两个虚拟网络避免多播路由死锁，从而均衡了不同维度上的缓存资源；BAM 的输出端口计算是基于多播报文的所有目标节点的位置，其能够高效地利用网络带宽。

6.2 归约消息组合框架

本节以一个多播 - 归约消息传输过程为例描述归约消息组合框架。在图6.3(a)中，9号节点注入一个目标节点是0、7和15的多播报文，这个多播报文的传输建立了一颗逻辑树^[239]，图中灰圈表现报文目标节点，白圈表示进行报文复制的节点，即逻辑树的分叉节点（fork node）。多播报文的目标节点需要向逻辑树的根节点9回复一个ACK报文。如图6.3(b)所示，当不采用消息组合框架时，每个ACK消息使用一个单播报文发送到9号节点，这些ACK报文的传输需

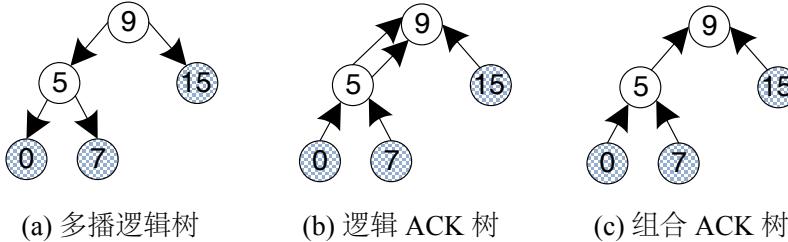


图 6.3 消息组合框架概述

要经过一些相同的网络链路，带来了一些冗余网络操作。图6.3(c)给出了采用消息组合的逻辑树，它与逻辑多播树的方向相反，节点 5 和节点 9 作为逻辑树分叉节点负责收集和转发 ACK 报文。消息组合框架的实现有两个关键点：1. 保证 ACK 报文沿着多播报文建立起的逻辑树反向传输；2. 保证分叉节点能知道每个多播-归约事务需要等待的 ACK 数目和已接收到的 ACK 数目。

在 $n \times n$ mesh 网络中，多播报文头部携带一个 $\log(n \times n)$ 位的 *pre_rep_router* 域，它记录多播报文上次进行复制的路由器编号，这个域的初始值等于源节点编号，在每次复制时更新。一个 3 位的 *ID* 字段用于区分同一源节点注入的多播报文，多播报文在注入时分配一个 *ID* 值。*src* 字段记录多播报文的源节点。

路由器上增加了一个消息组合表 (Message Combination Table, MCT)。当多播报文需要复制时，会从 MCT 中分配一个表项。该表项记录了多播报文上次进行复制操作的路由器编号和预期的 ACK 数目。这样就使得多播报文传输建立起的逻辑树的每个分支都有一个 MCT 表项指向逻辑树的上一个分叉节点。

每个多播目标节点会回复一个 ACK 报文，ACK 报文头部一个 $\log(n \times n)$ 位的 *cur_dest* 域记录了报文当前目标地址，其初始值等于触发该 ACK 的多播报文的 *pre_rep_router* 域。ACK 报文中还携带 *multicast_src* 和 *multicast_ID* 域，它们分别等于多播报文中的 *src* 和 *ID* 域。另外，还有 $\log(n \times n)$ 位的 *ACK_count* 域记录 ACK 报文携带的回复数目。

ACK 报文在到达当前目标节点时，会访问 MCT 表。如果还没有接收到所有预期的 ACK，路由器会将该 ACK 报文丢包，同时增加 MCT 表项中当前已接收的 ACK 数目。如果当前已接收到所有预期的 ACK，则设置该 ACK 报文的 *cur_dest* 域指向下一个多播复制路由器，该 ACK 报文将被发送到逻辑树的上一个分叉节点。这样就保证了 ACK 报文会沿着多播逻辑树的反方向传输。

6.2.1 消息组合表格式

图6.4给出消息组合表 (MCT) 的格式，*V* 域表示表项是否有效，*src*、*ID*、*pre_rep_router* 域从初始化该表项的多播报文头部获得。MCT 表是一个联想

| V | src | ID | pre_rep_router | incoming_port | expected_count | cur_ACK_count |
|-------|--------|--------|----------------|---------------|----------------|---------------|
| 1 bit | 4 bits | 3 bits | 4 bits | 3 bits | 4 bits | 4 bits |

图 6.4 消息组合表 (MCT 表) 格式

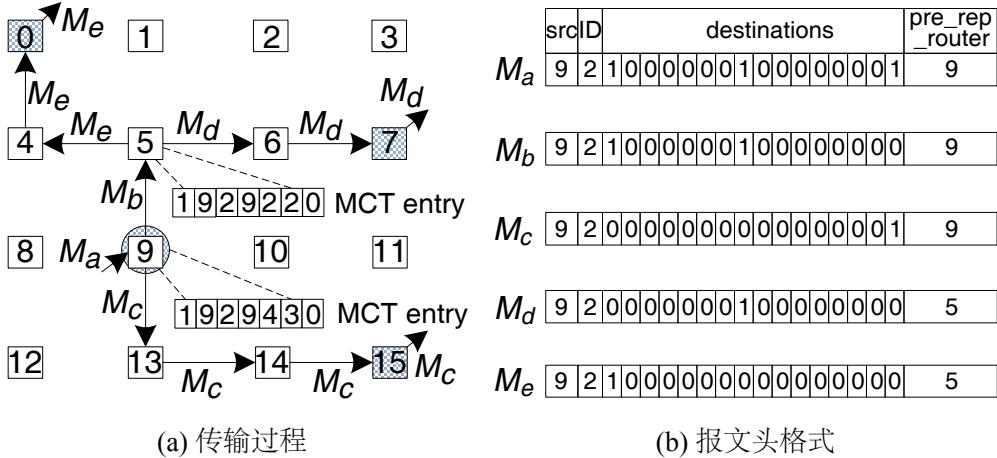


图 6.5 多播报文传输实例

存储器 (Content-Addressable Memory, CAM)，*src* 和 *ID* 域是 CAM 的访问标签，*incoming_port* 记录多播报文在本路由器的输入端口号，*expected_count* 域记录预期的 ACK 数目，它等于多播报文的目标节点数，*cur_ACK_count* 域记录当前已接收的 ACK 数目。MCT 表中记录预期的 ACK 数目，而非逻辑树的分支数目，这是为了处理 MCT 表项数不足的情况，详细分析见第 6.2.3 节。

6.2.2 消息组合实例

图 6.5(a)给出了图 6.3(a)的逻辑多播树在 4×4 mesh 网络中的传输过程，图 6.5(b)是这些多播报文的头格式。本章提出的消息组合框架与多播报文格式无关，但是为了论述方便，假设 *destinations* 域采用位编码方式^[240] 记录多播报文目标节点。 M_a 是注入的多播报文，其 *destinations* 域包含了三个目标节点。 M_a 在路由器 9 上复制成 M_b 和 M_c 两个报文，路由器 9 分配为 M_a 一个 MCT 表项，该表项 *src*、*ID*、*pre_rep_router* 的值分别从 M_a 的相应域中获得。MCT 表项的 *incoming_port* 值为 ‘4’，表示 M_a 是从本地端口输入到路由器 9 的；*expected_count* 值为 M_a 的目标节点数目，即为 ‘3’；*cur_ACK_count* 值为 ‘0’。报文 M_b 在路由器 5 上复制成 M_d 和 M_e 两个报文，同样的，路由器 5 为 M_b 分配一个 MCT 表项，该表项的 *expected_count* 值为 ‘2’。报文 M_d 和 M_e 中的 *pre_rep_router* 值为 ‘5’，表示上一次复制操作发生在路由器 5 上。

目标节点在接收到多播报文后会回复一个 ACK 报文。图 6.6(a)给出了 ACK

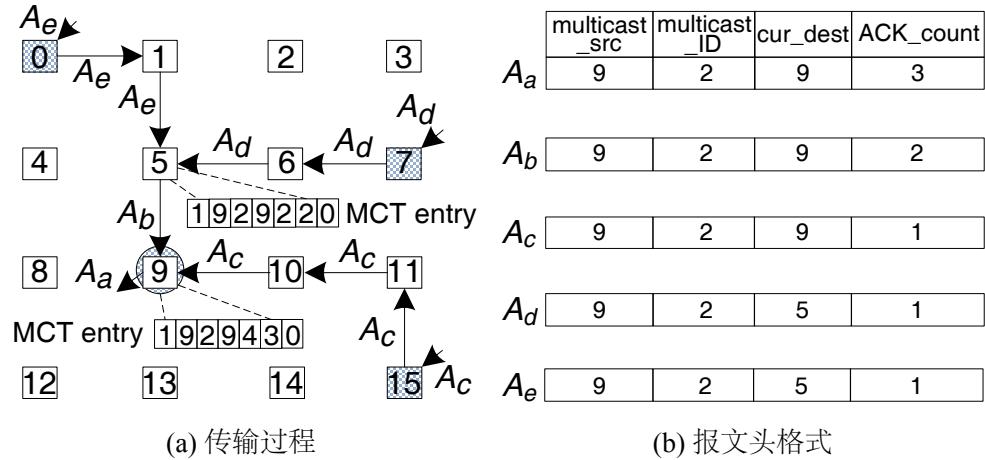


图 6.6 ACK 报文传输实例

报文的传输过程，图6.6(b)是这些 ACK 报文的头格式。 A_c 、 A_d 和 A_e 报文分别由多播报文 M_c 、 M_d 和 M_e 触发，这些 ACK 报文的 $multicast_src$ 、 $multicast_ID$ 和 cur_dest 域分别等于其触发多播报文的 src 、 ID 和 pre_rep_router 域。 A_c 、 A_d 和 A_e 中的 ACK_count 值都为 ‘1’，它们只携带了一个核的 ACK 信息。

ACK 报文中的 cur_dest 域定义了当前目标节点。如图6.6(a)所示， A_e 的当前目标节点是路由器 5。 A_e 通过一条与 M_e 不同的物理路径传输，消息组合框架只要求 ACK 报文按照多播逻辑树，而非物理路径传输，这增加了 ACK 报文的路由灵活性。同样，报文 A_c 的传输路径与 M_c 不同。 A_d 和 A_e 的 cur_dest 域都是路由器 5，它们在路由器 5 上进行组合。由于不同 ACK 报文的传输路径不同，多个 ACK 报文同时请求 MCT 表的概率很低 ($\leq 0.1\%$)，因此，消息组合框架使用一个小仲裁器对 MCT 表的多个同时访问请求进行排序。假设 A_d 比 A_e 先到达路由器 5， A_d 以 $multicast_src$ 和 $multicast_ID$ 为标签访问 MCT 表。由于 MCT 表项中 cur_ACK_count 和 A_d 报文中 ACK_count 的和是 ‘1’，而 $expected_count$ 值为 ‘2’， A_d 被丢包，同时 cur_ACK_count 值增 1。

A_e 报文到达路由器 5 时也会访问 MCT 表。此时路由器 5 已经接收到所有预期的 ACK，因此 A_e 会保持在网络中。同时， A_e 的 cur_dest 域被更新为 MCT 表项中的 pre_rep_router 域， A_e 的 cur_ACK 域被置为 ‘2’，此时 A_e 携带了处理器 0 和处理器 7 的 ACK 消息（图中 A_b ）。 A_b 使用 MCT 表项的 $incoming_port$ 作为这一跳的输出端口。为了避免再次进行路由计算，组合得到的 ACK 报文需按照多播报文的路径传输一跳，从而减少了一个时钟的延迟。此时，路由器 5 已经接收到所有预期的 ACK，相应 MCT 表项可以作废。最后，路由器 9 接收到 A_b 和 A_c ，并将它们组合成报文 A_a ，完成了一次多播 - 归约通信。

6.2.3 消息组合表项不足

第6.2.2节的例子假设多播报文复制时总是有空闲 MCT 表项。由于 MCT 表是有限的，有可能多播报文复制时没有空闲 MCT 表项，此时复制得到的报文不更新它们的 *pre_rep_router* 域。在图6.5中，如果路由器 5 在 M_b 复制时没有空闲 MCT 表项， M_d 和 M_e 会保持它们的 *pre_rep_router* 域为 ‘9’。当 ACK 报文 A_d 和 A_e 注入时，它们的 *cur_dest* 域被置为 ‘9’，这两个报文将在路由器 9 上组合，而不是图6.6中的路由器 5。在这种情况下，路由器 9 会从逻辑树同一分支上接收到两个 ACK 报文，这也是 MCT 表记录预期 ACK 数而非分支数的原因。按照这种设计，较小的 MCT 表只会影响性能，而不会带来正确性问题，第6.5.3节评估了 MCT 表大小对性能的影响。

6.3 均衡自适应多播路由算法

本节描述均衡自适应多播路由算法。为有效利用网络带宽，多播路由算法应该基于所有目标节点的位置计算输出端口^[70]。Wang 等提出的迭代划分多播路由 (Recursive Partitioning Multicast, RPM) ^[70] 能高效利用网络带宽。RPM 首先基于报文位置将网络划分成多个区域，之后采用了一系列优先级规则计算每个区域的输出端口。这些优先级规则尽可能地将多播报文沿着同一条路径传输，之后进行复制操作并将复制得到的报文发送到相互独立的目标节点集合。

尽管 RPM 能有效利用网络带宽，但是不同维度缓存资源的不均衡影响了它的性能。具体而言，为避免多播路由死锁，RPM 使用了两个虚拟网络：VN0 传输向上的报文，VN1 传输向下的报文。这种死锁避免方式将水平方向缓存划分给两个虚拟网络使用，而垂直方向缓存只给一个虚拟网络使用^[70, 237]，造成了缓存资源相对更为受限的水平方向成为性能瓶颈。为不同维度配置不同缓存数可以解决这个问题，但是其导致不同端口的控制逻辑的规模和复杂性不同，增加了设计难度^[33]。此外，路由器关键路径是由分配器在不同端口上的最大仲裁器所决定，这种异构结构可能会降低路由器的频率。因此，本章使用同构路由器结构。

基于如上观察，本章提出了均衡自适应多播路由算法 (Balanced, Adaptive Multicast, BAM)。与 RPM 采用两个虚拟网络实现死锁避免不同，BAM 的死锁避免是基于 Duato 的单播死锁避免理论^[162]。Cache 一致性协议的多播报文一般传输控制消息，片上大量的连线资源可以将这些控制消息编码成单切片报文^[69]，从而使得每个多播分支的路由相互独立。这些相互独立的分支可以被视为多个不相关的单播报文，因此，可以将 Duato 的单播死锁避免理论应用到多播路上。Duato 理论将虚通道划分成适应性虚通道和逃逸虚通道。二维 mesh 网络最短路由最多

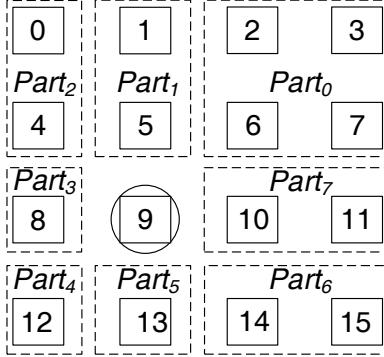


图 6.7 网络区域划分示意图

只有两个候选端口，处于适应性虚通道中的报文可以被任意发送到这两个端口上，这个属性使得 BAM 能基于多播所有目标节点的计算输出端口。值得一提的是，BAM 与基于 Duato 理论的单播路由算法是兼容的。

与 RPM 类似，BAM 首先基于报文当前位置进行网络划分，图6.7给出了将网络划分成 8 个区域的示意图。位于 $Part_1$ 、 $Part_3$ 、 $Part_5$ 和 $Part_7$ 中的目标节点只有一个候选输出端口，因此，如果多播报文有处于 $Part_1$ 、 $Part_3$ 、 $Part_5$ 或 $Part_7$ 中的目标节点，则对应的北输出端口、西输出端口、南输出端口或东输出端口一定会被使用。这些输出端口被称为该多播报文的‘必须’输出端口。位于 $Part_0$ 、 $Part_2$ 、 $Part_4$ 和 $Part_6$ 中的目标节点有两个候选输出端口，为了有效利用网络带宽，BAM 基于启发式策略为处于 $Part_0$ 、 $Part_2$ 、 $Part_4$ 和 $Part_6$ 中的目标节点选择输出端口：(1). 如果这些区域的两个候选输出端口中有且只有一个该多播报文的‘必须’输出端口，则使用该‘必须’输出端口发送报文；(2). 如果这些区域的两个候选输出端口都是该多播报文的‘必须’输出端口，或者两个都不是‘必须’输出端口，则自适应地选择具有较低拥塞程度的输出端口。这种设计最大程度地重用了‘必须’输出端口，从而能高效地使用链路带宽。

图6.8给出了 BAM 算法的计算逻辑， P_i 表示多播报文是否有处于 $Part_i$ 中的目标节点。以区域 $Part_0$ 的计算逻辑为例： N_{P_0} 和 E_{P_0} 分别表示路由器使用北端口或东端口将报文发送到 $Part_0$ ， N_{ne} 或 E_{ne} 表示在东北象限的两个输出端口中，北端口或东端口相对而言具有更低的拥塞，这些信号值由路由计算模块提供。

$Escape_n$ 、 $Escape_w$ 、 $Escape_s$ 和 $Escape_e$ 分别表示是否可以使用北端口、西端口、南端口和东端口的逃逸虚通道。如果路由器需要使用北端口将多播报文发送至位于 $Part_0$ 或 $Part_2$ 中的目标节点，该报文将来要进行一个违反维序路由的转向，因此不允许使用北端口的逃逸虚通道。南端口逃逸虚通道的使用规则与北端口类似，东端口和西端口的逃逸虚通道任何时候都可以使用。一旦多播报文进入逃逸虚通道，在后续路由将使用逃逸虚通道按照维序路由进行报文复制^[69]，此

$$\begin{aligned}
 N_{p_0} &= P_1 \cdot \overline{P}_7 + P_1 \cdot P_7 \cdot N_{ne} + \overline{P}_1 \cdot \overline{P}_7 \cdot N_{ne} & N &= P_1 + N_{p_0} + N_{p_2} \\
 E_{p_0} &= \overline{P}_1 \cdot P_7 + P_1 \cdot P_7 \cdot E_{ne} + \overline{P}_1 \cdot \overline{P}_7 \cdot E_{ne} & W &= P_3 + W_{p_2} + W_{p_4} \\
 N_{p_2} &= P_1 \cdot \overline{P}_3 + P_1 \cdot P_3 \cdot N_{nw} + \overline{P}_1 \cdot \overline{P}_3 \cdot N_{nw} & E &= P_7 + E_{p_0} + E_{p_6} \\
 W_{p_2} &= \overline{P}_1 \cdot P_3 + P_1 \cdot P_3 \cdot W_{nw} + \overline{P}_1 \cdot \overline{P}_3 \cdot W_{nw} & S &= P_5 + S_{p_4} + S_{p_6} \\
 S_{p_4} &= \overline{P}_5 \cdot \overline{P}_3 + P_5 \cdot P_3 \cdot S_{sw} + \overline{P}_5 \cdot \overline{P}_3 \cdot S_{sw} & Escape_n &= \overline{N}_{p_0} \cdot \overline{N}_{p_2} \cdot P_1 \\
 W_{p_4} &= P_5 \cdot \overline{P}_3 + P_5 \cdot P_3 \cdot W_{sw} + \overline{P}_5 \cdot \overline{P}_3 \cdot W_{sw} & Escape_w &= W \\
 S_{p_6} &= \overline{P}_5 \cdot \overline{P}_7 + P_5 \cdot P_7 \cdot S_{se} + \overline{P}_5 \cdot \overline{P}_7 \cdot S_{se} & Escape_s &= \overline{S}_{p_4} \cdot \overline{S}_{p_6} \cdot P_5 \\
 W_{p_6} &= \overline{P}_5 \cdot P_7 + P_5 \cdot P_7 \cdot E_{se} + \overline{P}_5 \cdot \overline{P}_7 \cdot E_{se} & Escape_e &= E
 \end{aligned}$$

图 6.8 BAM 多播路由算法逻辑

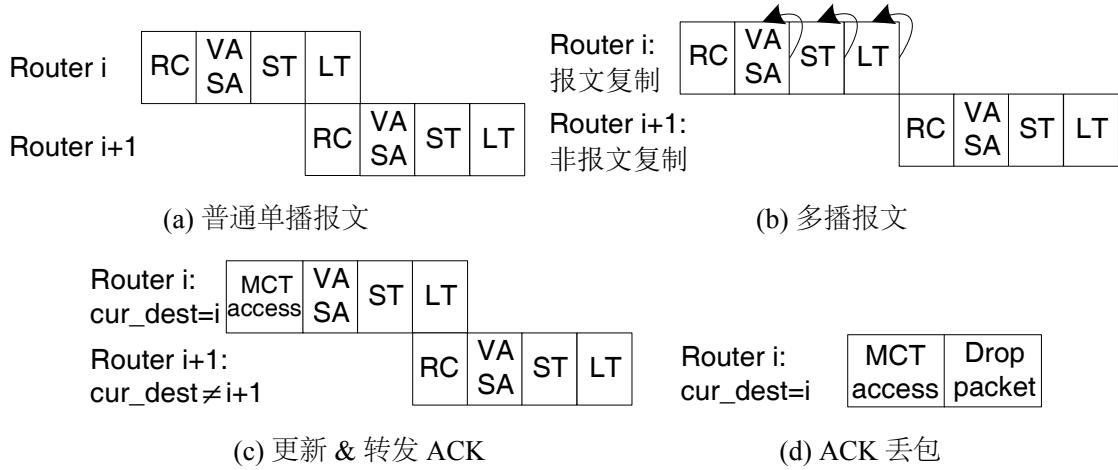


图 6.9 路由器流水线

时，逃逸虚通道之间没有死锁。由于任何位于适应性虚通道中的报文都有机会进入逃逸虚通道，BAM 是没有死锁的。与 RPM 不同，BAM 不需要设置两个虚拟网络，其在垂直方向和水平方向之间的缓存资源较为均衡，有利于提升性能。此外，BAM 基于所有目标节点的位置计算输出端口，能有效利用网络带宽。

6.4 路由器流水线和微结构

本章基准路由器是一个猜测交叉开关分配虚通道路由器^[7, 38, 163]。路由器使用超前信号发送方式^[41, 175]在切片传输的前一时钟将单播路由相关信息发送到下一个节点，因此，路由计算段（RC）与链路传输段（LT）并行执行。路由器使用预选择策略选择输出端口，其基于前一时钟的网络状态，预选择每个象限下一时钟的优先输出端口^[60, 63, 148]。单播报文的流水线如图6.9(a)所示。

由于使用超前信号发送多播报文相关信息需要太多的连线，多播报文采用三级流水线结构，如图6.9(b)所示。如果多播报文需要发送到多个输出端口，报文采

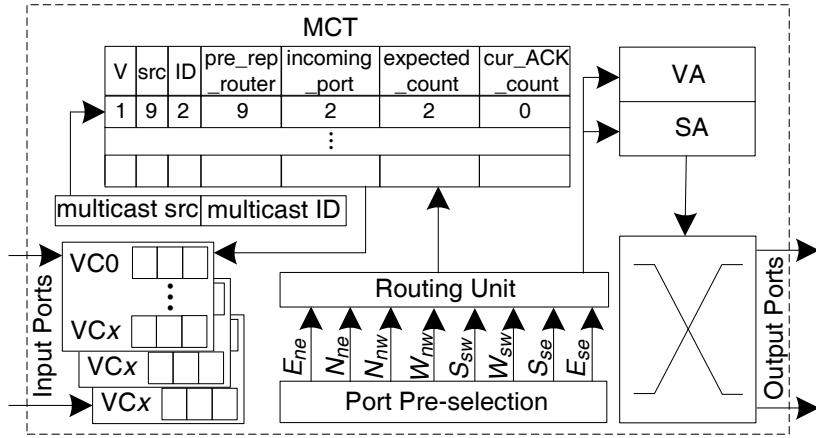


图 6.10 路由器微结构

用异步复制策略进行复制，异步复制策略可以消除多播分支间的耦合。虚通道分配器（VA）和交叉开关分配器（SA）对多播报文的处理与单播报文类似，唯一的区别是：只有在所请求的多个输出端口都得到满足时，多播报文才能从输入虚通道移除^[70, 74]。

ACK 报文的流水线与普通单播报文的流水线不同。当 ACK 报文到达时，路由器会检查报文的 *cur_dest* 域，如果 *cur_dest* 域不等于本路由器编号，对该 ACK 报文的处理与普通单播报文相同（图6.9(a)）；如果 *cur_dest* 域等于本路由器编号，则 ACK 报文会访问 MCT 表。如第6.5.3节所述，MCT 表的访问可以在一个流水段内完成，不会增加关键路径延迟。图6.9(c)和图6.9(d)给出了 ACK 报文的流水线。

图6.10描述了路由器微结构。如果多播报文需要发送到多个输出端口上，则路由器在 MCT 表中为该多播报文分配一个表项，这个操作与 VA/SA 操作重叠，不会影响路由器关键路径。Port Pre-selection 模块为每个象限预选择优先输出端口^[60, 63, 148]，单播和多播路由计算都会使用该模块的 8 个输出信号。

6.5 实验评估

本节评估了所提出的消息组合框架与 RPM 和 BAM 结合使用时的性能。实验评估中既使用了合成流量模式，又使用了真实应用程序。实验修改时钟精确模拟器 Booksim^[163]，以模拟第6.4节中描述的路由器流水线和微结构。合成流量模式评估使用两个虚拟网络以避免协议层死锁^[163]，多播报文和 ACK 报文分别传输在这两个虚拟网络上。RPM 需要进一步将多播虚拟网络划分成两个子虚拟网络，分别用于传输向上和向下的报文。BAM 不需要进一步划分多播虚拟网络。普通单播报文在注入时随机选择虚拟网络，但是一旦注入后，报文只能固定在一个虚拟网络内传输。单播报文采用自适应路由算法。BAM 的多播虚拟网络、BAM 的 ACK 虚拟网络和 RPM 的 ACK 虚拟网络上的自适应路由算法是基于 Duato 理论^[162] 设

表 6.1 实验配置参数

| 属性 | 基准值 | 变量 |
|-----------------|---|-------------------------------|
| 网络拓扑 | 4×4 mesh | 8×8 mesh, 16×16 mesh |
| 虚通道 (VC) 配置 | 4 slots/VC, 8 VCs/port | 4 VCs/port, 6 VCs/port |
| 报文长度 | 单播: 1 个切片 (50%), 5 个切片 (50%) ACK: 1 个切片; 多播: 1 个切片 | - |
| 单播流量模式 | uniform random, transpose, bit rotation, hot spot | - |
| 多播报文比例 | 10% | 5%, 15%, 20% |
| 多播报文 目标节点数 | 2 - 10 (均匀分布) | 2 - 4, 4 - 14, 10 - 14, 15 |
| ACK 回复延迟 | 1-4 时钟周期内均匀分布 | - |
| MCT 表项 | 64 | 0, 1, 4, 16 |
| Warmup 时钟, 测量时钟 | 10000 & 100000 cycles | - |

计的，它们配置了一条逃逸虚通道。由于 RPM 的多播虚拟网络被划分成两个子虚拟网络，其自适应路由算法不需要配置逃逸虚通道。自适应路由算法使用局部自适应选择策略：当存在两个候选输出端口时，使用具有较多空闲缓存的端口。

多播报文和 ACK 报文只包含 1 个切片。普通单播报文的长度呈现双峰分布，其中 50% 的报文包含 5 个切片，另外 50% 的报文包含 1 个切片。实验使用了多种合成流量模式^[163]，包括 uniform random、transpose、bit rotation 和 hotspot。实验对注入多播报文比例进行了控制；多播报文的目标节点数目和位置均匀随机分布。节点回复 ACK 报文的延迟在 1 到 4 个时钟内均匀随机分布。基准实验配置的 MCT 表具有 64 个表项，第6.5.3节评估了 MCT 表大小对性能的影响，表6.1给出了基准实验配置参数和用于敏感性分析的参数值。

实验使用两个模拟器评估全系统的性能：FeS2^[185] 模拟 x86 体系结构，Booksim 模拟 FeS2 产生的报文在片上网络层的传输。实验模拟了一个 16 核的 CMP 平台，其拓扑结构是 4×4 mesh 网络，并评估了 PARSEC 测试集^[201] 在 16 线程运行时的性能。实验假设处理器频率是网络频率的 5 倍。Cache 行包括 64 字节，网络切片大小是 16 字节，长短报文分别包含 5 个和 1 个切片。实验使用了一种基于目录的分布式 MOESI 协议，该协议需要 4 个虚拟网络以防止协议层死锁，Cache 行作废报文在 VN0 上传输，ACK 报文在 VN2 上传输。每个虚拟网络的配置，包括虚通道数目、虚通道深度和 MCT 表大小，与表6.1中的基准配置相同。PARSEC 应用程序采用 *simsmall* 输入集，全系统性能指标是程序运行时间，表6.2给出了全系统模拟的配置参数。

表 6.2 全系统模拟参数

| 参数名 | 参数值 |
|------------------|----------------------------|
| 核数 | 16 |
| 一级 cache (数据或指令) | private, 4-way, 32KB each |
| 二级 cache | private, 8-way, 512KB each |
| Cache 一致性协议 | 分布式目录 MOESI 协议 |
| 网络拓扑 | 4×4 Mesh, 4 个虚拟网络 (VN) |

6.5.1 性能

实验评测了四种不同设计的性能：RPM 不使用消息组合框架 (RPM+NonCom)、RPM 使用消息组合框架 (RPM+Com)、BAM 不使用消息组合框架 (BAM+NonCom) 和 BAM 使用消息组合框架 (BAM+Com)。

6.5.1.1 网络整体性能

图6.11给出了网络整体性能。将 RPM+Com 和 BAM+Com 分别与 RPM+NonCom 和 BAM+NonCom 进行比较，RPM+Com 和 BAM+Com 都取得了性能提升，它们不仅降低了报文平均延迟，也提升了饱和吞吐率。分析发现，消息组合将 ACK 报文的平均链路传输数从 4.7 降低到 2.5，减少了 ACK 报文 45% 的网络操作。

丢包机制减少了报文平均跳数，因此，在低负载和高负载下都能降低网络平均延迟。此外，报文组合减轻了 *ejection* 端口的拥塞，也有利于降低网络延迟。在低到中等负载下（本章定义低到中等网络负载是指注入率从 0 到网络平均延迟等于零负载延迟两倍时的注入率），RPM+Com 的延迟比 RPM+NonCom 低 10%~20%（平均 14.1%），在更高网络负载下，RPM+Com 获得的延迟下降更为显著。BAM+Com 相对于 BAM+NonCom 也获得了类似的延迟下降。将 ACK 报文丢包降低网络负载，从而提高饱和吞吐率。在这 4 种流量模式下，消息组合框架给 RPM+Com 带来了 8.5%~11.2%（平均 9.6%）的饱和吞吐率提升。BAM+Com 相对于 BAM+NonCom 也获得了类似的饱和吞吐率提升。

BAM 在不同维度间维持了缓存资源的均衡，有利于获得更高的饱和吞吐率。将 BAM+NonCom 和 BAM+Com 分别与 RPM+NonCom 和 RPM+Com 进行比较，BAM+NonCom 和 BAM+Com 都支持更高的饱和吞吐率。尽管 BAM+NonCom 在 transpose、bit rotation 和 hotspot 模式的低负载延迟高于 RPM+Com，BAM+NonCom 的饱和吞吐率优于 RPM+Com。在这 4 种模式下，BAM+Com 相对于 RPM+NonCom 的饱和吞吐率提升分别是 14.2%、27.6%、26.4% 和 25.1%（平均 23.4%），ACK 丢包和均衡缓存配置带来了这些性能提升。从图6.11可以看出，BAM+NonCom 和 RPM+NonCom 之间的趋势与 BAM+Com 和 RPM+Com 之间的趋势相似，因此，

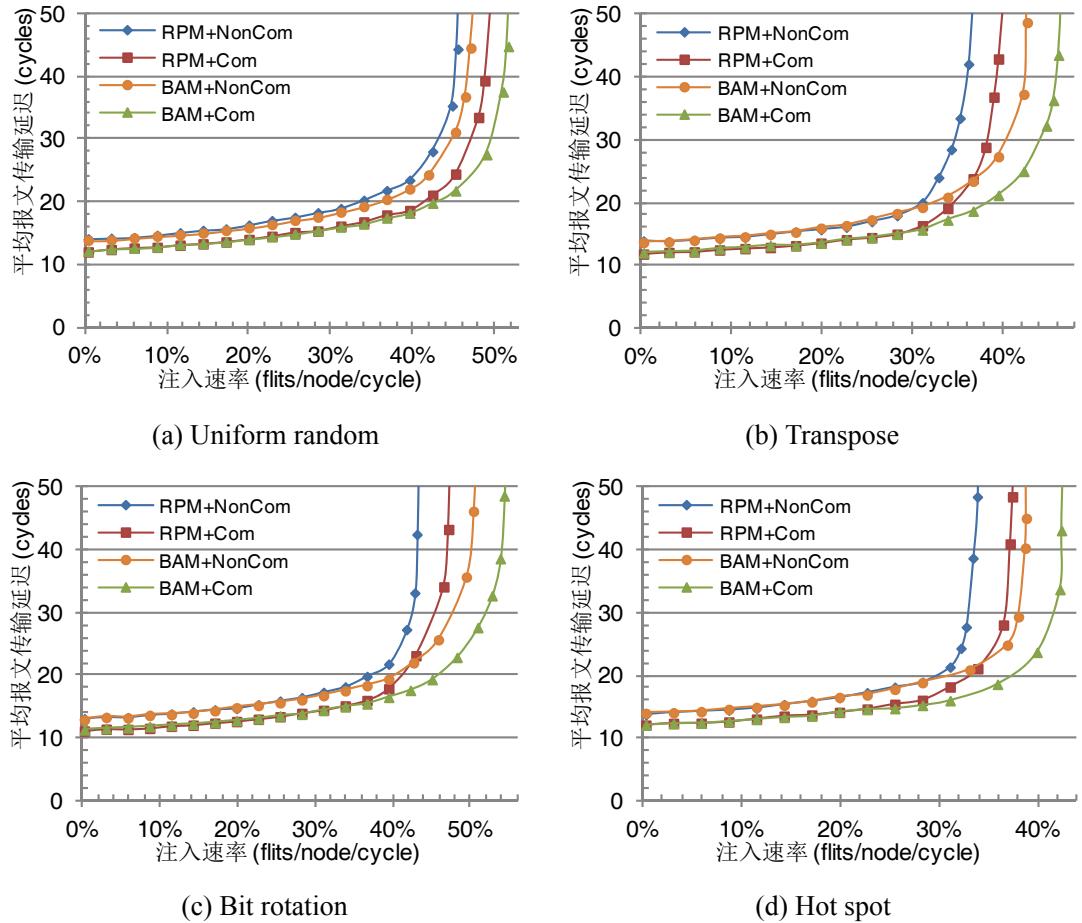


图 6.11 网络总体性能

后续章节不再讨论 BAM+NonCom 的性能。

6.5.1.2 多播 - 归约事务延迟

为深入分析消息组合对多播 - 归约事务的影响，图6.12测试了多播 - 归约事务的传输延迟，该图包含 5 组注入率，其中最后一组注入率高于 RPM+NonCom 的饱和吞吐率。将 ACK 报文丢包降低了网络拥塞，加速了多播 - 归约事务的传输，在所有的注入率下，RPM+Com 和 BAM+Com 的多播 - 归约事务延迟都低于 RPM+NonCom。多播报文需要复制成多个报文，其传输延迟取决于面临最大拥塞的复制报文，多播报文对网络负载下降的敏感性不如 ACK 报文，使得 ACK 报文传输速度的提升高于多播报文传输速度的提升。在低到中等负载（小于 30%）的 uniform random 模式下，BAM+Com 平均多播延迟比 RPM+NonCom 低 9.5%，而 ACK 传输延迟则低 17.6%。多播报文和 ACK 报文传输的加速给多播 - 归约事务延迟带来了 13.4% 的平均下降。报文组合能减少多播源节点需要等待的 ACK 报文数目，单个 ACK 报文遇到的网络拥塞低于多个不同的 ACK 报文遇到的拥塞。因此，网络负载越高，消息组合对事务传输的加速越明显。在高负载的

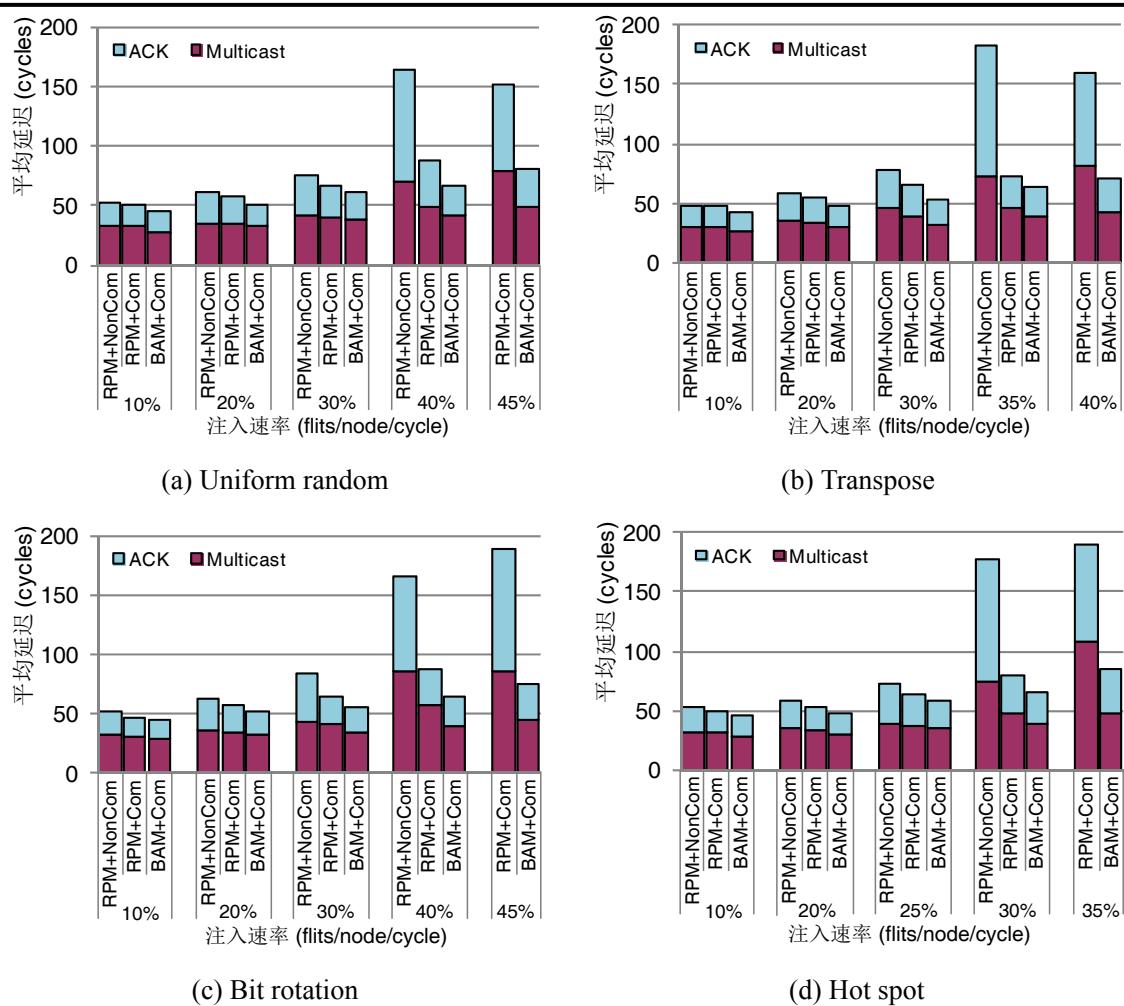


图 6.12 多播 - 归约事务传输延迟

uniform random 模式下，RPM+Com 和 BAM+Com 相对于 RPM+NonCom 分别降低了 46.2% 和 57.6% 的多播 - 归约事务延迟。换句话说，在高负载情况时，消息组合框架可以将多播 - 归约事务加速一倍。随着网络负载的增加，BAM+Com 的事务传输延迟明显低于 RPM+Com 的事务传输延迟。

6.5.1.3 PARSEC 程序性能

图6.13给出了各设计在 PARSEC 程序下相对于 RPM+NonCom 的性能加速比。消息组合框架主要优化了全系统模拟中 1 个虚拟网络（ACK 虚拟网络）的性能，但是聚合通信一般处于应用程序的关键路径上，同时消息组合降低了交叉开关的冲突，从而能提升其它 3 个虚拟网络的性能，在这 10 个应用程序中，RPM+Com 相对于 RPM+NonCom 的性能提升为 5.3%~8.3%。消息组合的有效性取决于多播报文的目标节点数目，Bodytrack、facesim 和 raytrace 的多播报文目标节点数较低，因此，消息组合在这 3 个程序上获得的性能提升低于其它程序。

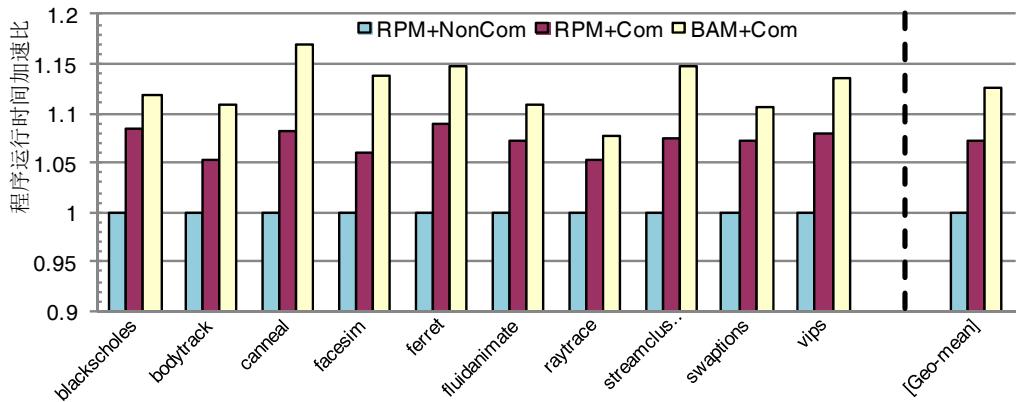


图 6.13 PARSEC 测试集结果

BAM+Com 所使用的均衡缓存配置支持更高的网络吞吐率，它在网络负载较重或具有较多猝发通信的程序中能获得性能提升。均衡缓存配置给 blackscholes、fluidanimate、raytrace 和 swaptions 带来的性能提升 (BAM+Com vs. RPM+Com) 为 2.3%~3.7%，这些程序的网络负载较低，限制了均衡缓存配置获得的性能提升。然而，由于 bodytrack、canneal、facesim、ferret、streamcluster 和 vips 的网络负载相对较高，同时也具有一些猝发通信，BAM+Com 相对于 RPM+Com 在这些程序上的性能提升为 5.5%~8.6%。在这 10 个程序中，BAM+Com 相对于 RPM+NonCom 的平均性能提升是 12.7%，最大性能提升是 canneal 中的 16.8%。

6.5.2 BAM 和 RPM 多播虚拟网络性能

本节研究 RPM 多播虚拟网络的不均衡缓存配置对单播路由和多播路由的影响。由于 RPM 和 BAM 的 ACK 虚拟网络相同，本节实验只评估了它们的多播虚拟网络，该虚拟网络包含 4 条虚通道。RPM 需要进一步划分两个子虚拟网络，每个子虚拟网络的水平方向包含 2 条虚通道，垂直方向包含 4 条虚通道。

6.5.2.1 单播性能

图 6.14 给出了只传输单播报文时的性能。RPM 的多播虚拟网络测试了三种路由算法，分别是 XY、YX 和一种局部自适应路由算法 (Adaptive)（图 6.14 中的 *RPM's VN - Adaptive* 曲线）。XY 路由能均衡分布 uniform random 通信，因此，其在该模式的算法中性能最好。由于自适应地选择输出端口能减轻非均衡缓存配置对性能的负面影响，Adaptive 在其它三种模式中获得了最高的性能。因此，本章基准配置中单播报文使用自适应路由算法。

尽管 Adaptive 的性能优于 XY 和 YX，它依然受限于非均衡缓存配置，RPM 每个子虚拟网络的水平方向的缓存资源只有垂直方向的一半，导致水平方向成为

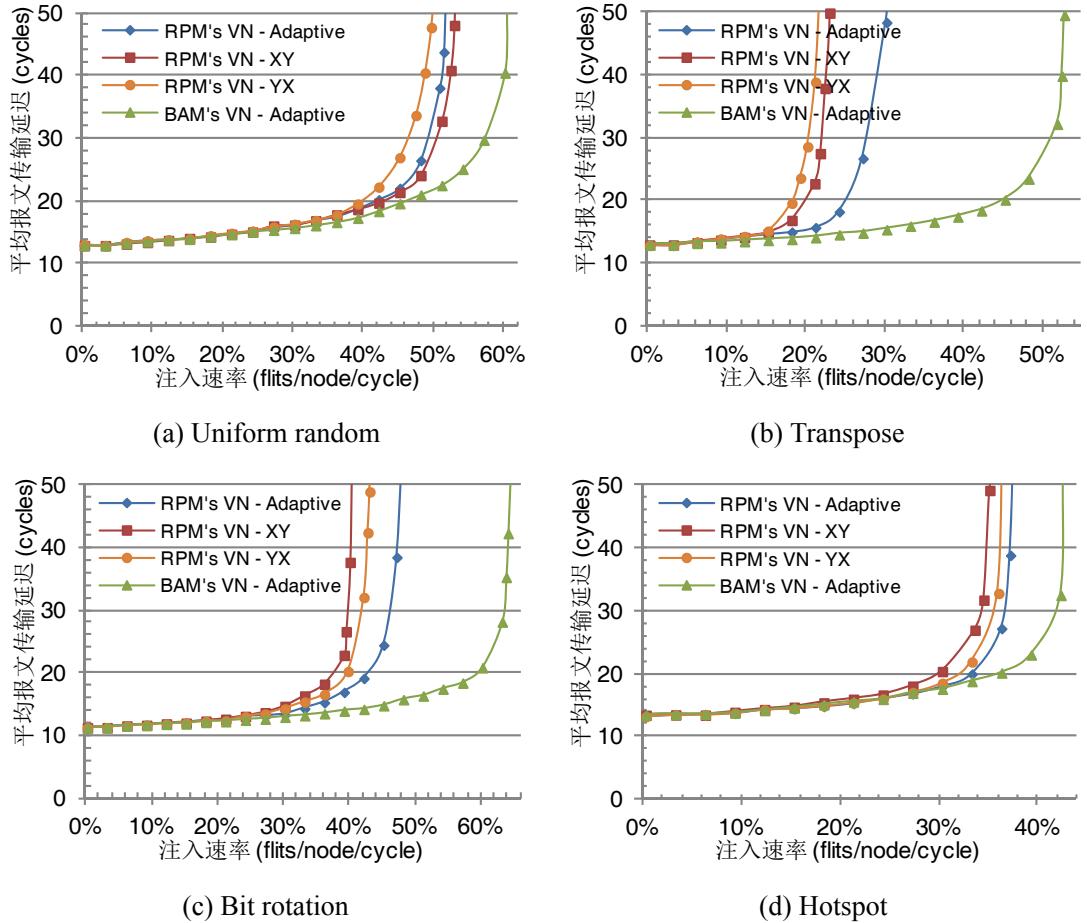


图 6.14 单播通信在 RPM 和 BAM 多播虚拟网络中的性能

性能瓶颈。然而，在 BAM 的多播虚拟网络中，水平方向和垂直方向的缓存资源是均衡的。BAM 虚拟网络中的自适应路由算法相比于 RPM 虚拟网络中的自适应路由算法获得了很大的性能提升，最大的饱和吞吐率提升是在 transpose 模式下的 73.2%，在这 4 种流量模式下，*BAM's VN - Adaptive* 相比于 *RPM's VN - Adaptive* 的平均饱和吞吐率提升是 35.3%。

6.5.2.2 多播性能

图6.15给出了只传输多播报文时的性能，图中的 *Adaptive* 是一种自适应的多播路由算法，它为每个目标节点自适应地选择输出端口，不考虑重用‘必须’输出端口。这种设计影响了带宽的有效利用，导致 *Adaptive* 的饱和吞吐率比 BAM 低 8.7%。多播报文只有在请求的所有输出端口都得到满足时才能从虚通道移除，因此，相对拥塞的水平方向对多播报文的影响高于对单播报文的影响。图6.15中 BAM 的饱和吞吐率比 RPM 高 47.1%，而图6.14中的单播报文的平均提升是 35.3%。RPM、Adaptive 和 BAM 的多播报文平均跳数分别是 8.6、8.8 和 8.4，这也说明了 BAM 的启发式端口选择策略能高效利用网络带宽。

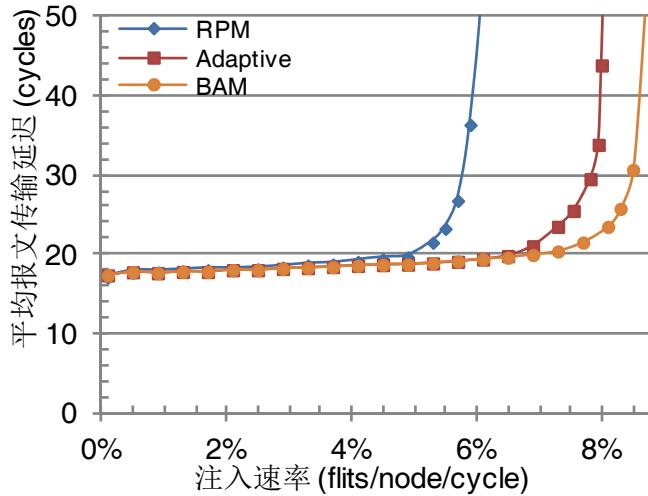


图 6.15 100% 多播报文比例下的性能

6.5.3 消息组合表大小

第6.2.3节指出 MCT 表的大小不会影响正确性，但是会影响性能。MCT 表的项数过少会阻止消息组合，增加 ACK 报文的跳数。为了得到合适的 MCT 表规模，本节配置一个无限 MCT 表，并记录不同负载情况下的使用表项数，实验是在基准网络配置中（表6.1）使用 uniform random 模式进行的。图6.16(a)给出了网络所有节点的最大使用 MCT 表项数和平均使用表项数。在低到中等负载下（注入率小于 39%），最大使用表项数低于 10，平均使用表项数低于 1.5；即使当网络饱和时（52% 的注入率），最大使用表项数是 49，平均使用表项数是 10.15。这些结果表明较少的 MCT 表项数能支持较高的性能。

图6.16(b)给出了不同规模 MCT 表支持的性能。0-entry 曲线表示不进行消息组合，即所有的 ACK 报文在注入时都将目标地址设置为多播报文的源地址。MCT 表只有一个表项时，ACK 组合能降低 10% 的平均网络延迟，更多的表项能进一步降低网络延迟，尤其是在高负载情况下。当 MCT 表项数从 1 个上升到 64 个时，饱和吞吐率提升从 3.3% 增长到 12.1%。

本节在 32 nm 工艺下使用 Cacti^[241] 评估 MCT 表的功耗、面积和延迟，如表6.3所示。在 1 GHZ 频率下，访问 64 表项的 MCT 表可以在一个时钟内完成。64 表项的 MCT 表在几乎所有负载情况下都能获得较大性能提升。例如，在全系统模拟中，PARSEC 测试集的最大使用表项数少于 25。在面积受限的环境下，使用更少的表项也能获得一定的延迟降低和吞吐率提升。

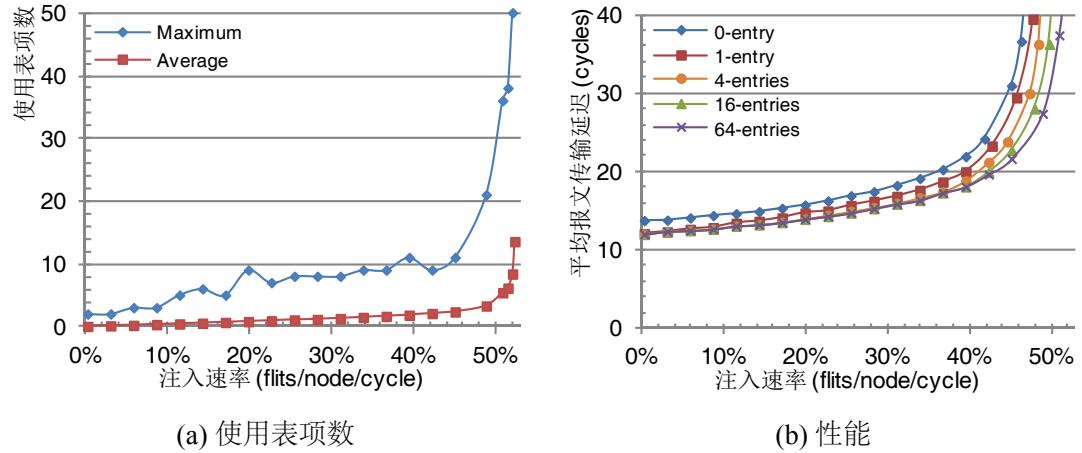


图 6.16 消息组合表 (MCT) 大小评估

表 6.3 MCT 表硬件开销

| 表项数 | 面积 (mm^2) | 能耗 (nJ) | 延迟 (ns) | 字节数 |
|-----|---------------|---------|---------|-----|
| 16 | 0.0011 | 0.0008 | 0.138 | 48 |
| 32 | 0.0017 | 0.0013 | 0.146 | 96 |
| 64 | 0.0031 | 0.0026 | 0.153 | 192 |

6.5.4 敏感性分析

为了进一步理解设计的性能和可扩展性，本节开展了一系列敏感性分析实验，敏感性分析主要针对虚通道数目、多播报文比例、多播报文目标节点数目和网络规模四个方面。图6.17给出了四种合成流量模式下的平均性能提升，在条块分组中，前两个长条表示网络饱和吞吐率的提升，后两个长条表示网络延迟在低到中等负载时的下降。

6.5.4.1 虚通道数目

图6.17(a)给出了物理链路配置 8 条、6 条或 4 条虚通道时的性能，从图中可以观察到一个有趣的现象：当配置较少虚通道时，由消息组合框架带来的性能提升更为明显 (RPM+Com vs. RPM+NonCom)，同时由均衡缓存配置带来的性能提升有所下降 (BAM+Com vs. RPM+Com)。这个现象可以从两方面进行解释：第一，虚通道数越少，缓存资源越珍稀，消息组合能增加这种珍稀资源的可重用性。例如，当配置 4 条虚通道时，RPM+Com 的饱和吞吐率比 RPM+NonCom 高 14.8%。随着虚通道数的增加，缓存资源的珍稀性逐渐下降。但是即使配置 8 条虚通道时，消息组合框架依然能获得 9.6% 的饱和吞吐率提升。

第二，BAM+Com 使用逃逸虚通道避免死锁。水平方向逃逸虚通道任何时候都可以使用，而只有当输出端口符合维序路由时，才能使用垂直方向逃逸通

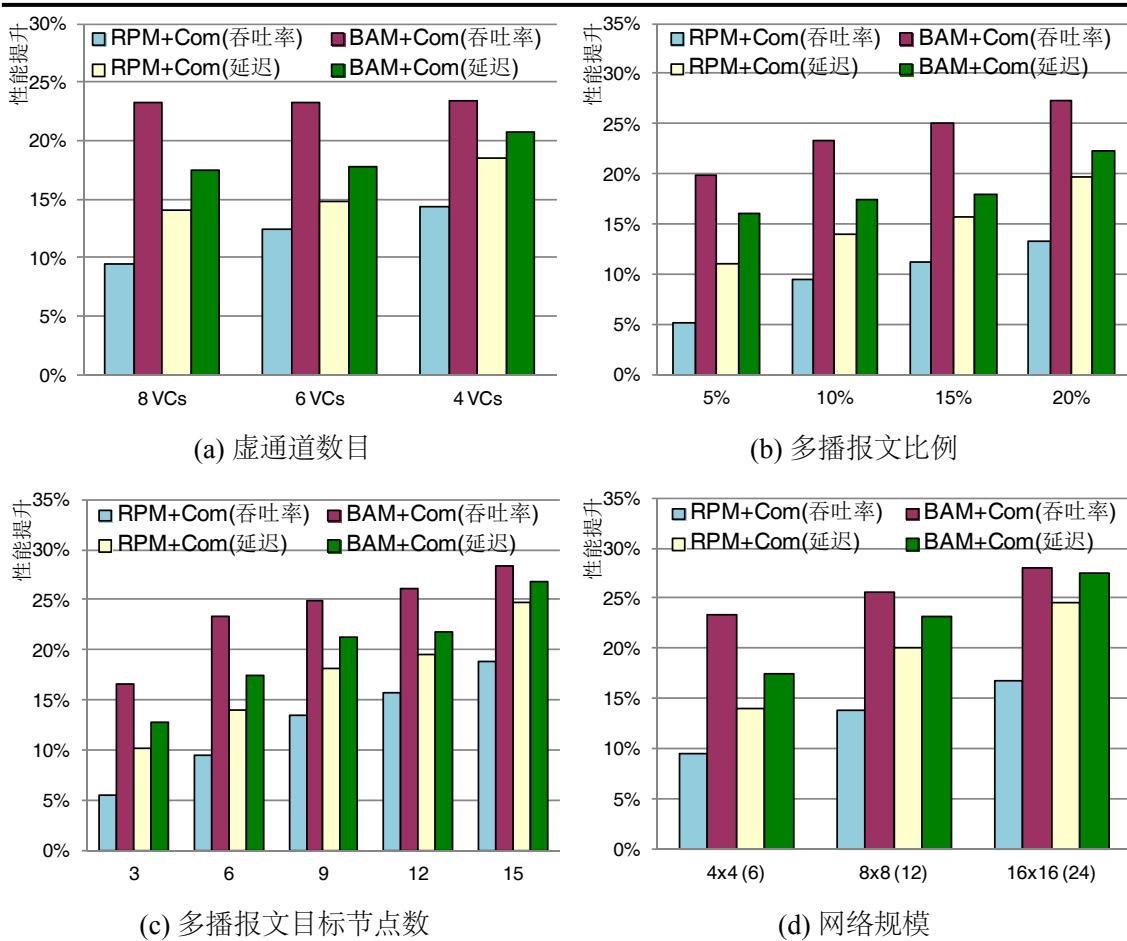


图 6.17 RPM+Com 和 BAM+Com 相对于 RPM+NonCom 的性能提升

道。因此，不同维度的逃逸虚通道使用存在不均衡，这种不均衡程度随着虚通道数的减少而上升。然而，RPM+Com 的不均衡缓存配置更为严重，RPM+Com 多播虚拟网络垂直方向缓存数量是水平方向的两倍。因此，即使只配置 4 条虚通道，BAM+Com 相对于 RPM+Com 依然获得了 9.0% 的饱和吞吐率提升。

当配置更少虚通道时，RPM+Com 相对于 RPM+NonCom 的延迟下降更明显，该延迟下降在 4 条虚通道时是 18.5%。BAM+Com 的均衡缓存配置使得其相对于 RPM+NonCom 的延迟下降更为显著。BAM+Com 和 RPM+Com 之间的延迟下降差异没有它们的饱和吞吐率提升差异明显，这是因为自适应路由只有当负载较大时才能降低延迟。

6.5.4.2 多播报文比例

图6.17(b)给出了不同多播报文比例下的性能。消息组合框架带来的饱和吞吐率提升在更高多播报文比例时更为明显，均衡缓存配置带来的饱和吞吐率提升在不同多播报文比例下基本保持不变。多播报文比例的提高带来的更多的 ACK 报文数目。因此，消息组合机制具有更大的可能性组合 ACK 报文，从而减低网络

负载。当多播报文比例是 20% 时，BAM+Com 获得了 27.2% 的饱和吞吐率提升。多播报文比例的上升也能带来更为明显的延迟下降。本实验配置的虚通道数保持不变（8 条），因此，BAM+Com 与 RPM+Com 之间的性能差异基本保持不变。

6.5.4.3 多播报文的平均目标节点数

图6.17(c)给出了多播报文平均目标节点数为 3、6、9、12 和 15 时的性能。这个参数对性能的影响与多播报文比例对性能的影响类似。随着多播报文平均目标节点数的增加，每个多播报文产生的 ACK 报文数也会增加。因此，消息组合框架能够将更多 ACK 报文组合，从而带来更大的性能提升。当平均目标节点数目从 3 个变化到 15 个时，BAM+Com 获得的饱和吞吐率提升从 17% 增长到 28%，RPM+Com 获得的延迟降低从 10% 增长到 25%，BAM+Com 获得的延迟降低从 13% 增长到 27%。

6.5.4.4 网络规模

图6.17(d)给出了 4×4 mesh、 8×8 mesh 和 16×16 mesh 网络下的性能。由于 8×8 和 16×16 mesh 网络具有更多的节点，它们的多播报文平均目标节点数分别为 12 和 24 个。报文在更大规模网络中需要传输更多的跳数，因此，报文组合能够减少更多的网络操作数，从而获得更大的性能提升。在 16×16 mesh 中，消息组合框架获得了 16.8% 的饱和吞吐率提升。类似的，消息组合框架在规模越大的网络中获得的延迟下降越明显。在 16×16 mesh 中，RPM+Com 和 BAM+Com 分别获得了 25% 和 27% 的延迟下降。BAM+Com 的均衡缓存配置带来的性能提升在不同规模网络中基本保持不变。上述敏感性分析实验使用了 64 表项的 MCT 表。有人可能会认为更多的多播报文目标节点数、更高的多播报文比例或者更大的网络规模会需要使用更大的 MCT 表。然而，实验表明，随着这些参数的增加，网络饱和吞吐率会下降，使得网络同时支持的多播 - 归约事务数也会下降。因此，64 表项的 MCT 表在上述不同网络配置下都获得了较高的性能。

6.6 功耗和能量延迟积分析

本节使用一个片上网络功耗模型^[205] 计算链路、输入缓存和路由器逻辑的功耗，输入缓存功耗和链路功耗中包括静态功耗。本节将 MCT 表访问功耗集成到这个模型上，网络各模块的活跃指数通过 Booksim 模拟器获得。假设链路带宽是 128 位，频率是 1 GHz，实验使用 32nm 工艺。图6.18(a)和图6.18(b)分别给出了单播报文是 transpose 流量模式时的功耗和能量延迟积结果。该图将功耗分成三部分：MCT 表访问功耗、网络静态功耗和网络动态功耗。

MCT 表访问功耗只占整个网络功耗的一小部分。这有两方面的原因：1、

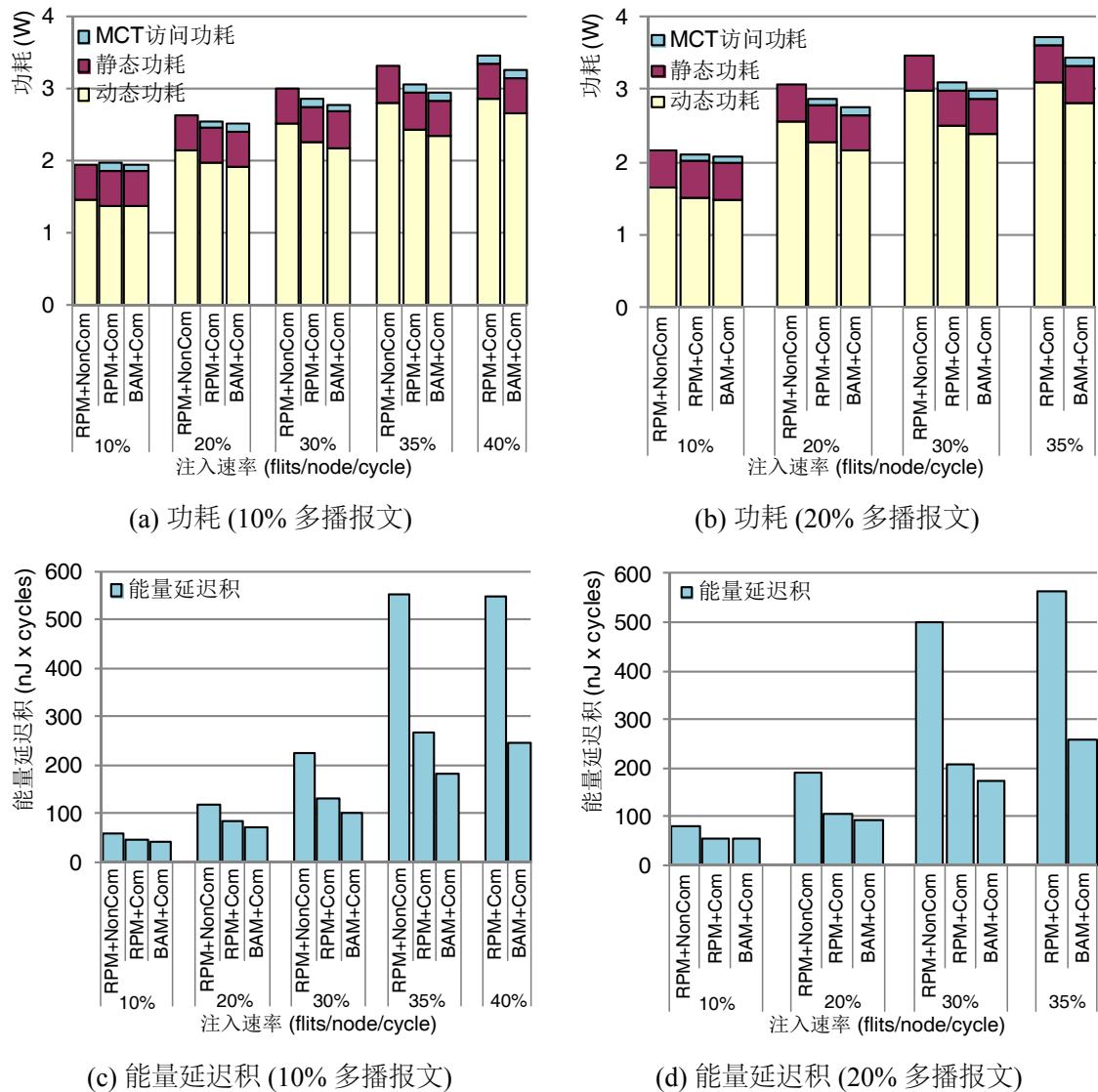


图 6.18 功耗和能量延迟积结果

MCT 表规模较小。每个 MCT 表项是 3 个字节，一个 64 表项的 MCT 表包含 192 字节，这只是切片缓存大小的 7.5%。2、MCT 表访问的频率很低。即使在网络饱和时，也只有 7.2% 的时钟周期会访问 MCT 表；当网络没有饱和时，MCT 表的访问频率更低。与 RPM+NonCom 相比，RPM+Com 在所有的注入率下都获得了功耗的降低，这是由 ACK 报文丢包带来的。BAM+Com 的均衡缓存资源配置能进一步降低功耗。随着注入率的上升，RPM+Com 和 BAM+Com 所获得的功耗下降更为明显。在 10% 多播报文比例下，RPM+Com 在 35% 注入率时相对于 RPM+NonCom 获得了 7.6% 的功耗下降。此时，BAM+Com 的功耗下降是 10.8%。消息组合机制在更高的多播报文比例下能够降低更多的功耗。

如图6.18(c)和图6.18(d)所示，本章设计的能量优势在能量延迟积^[242]上更为

明显，这是因为 ACK 报文丢包不仅能减少网络操作数，同时也会降低网络延迟。在 10% 多播报文比例下，RPM+Com 和 BAM+Com 在低到中等负载下获得的能量延迟积下降为 20%~40%。在高负载时，它们获得了 60%~75% 的能量延迟积下降。多播报文比例的上升能带来更大的能量延迟积下降。

6.7 相关研究

本节讨论消息组合和片上网络多播路由的相关研究。

6.7.1 消息组合

栅栏同步是一种重要的聚合通信，这种通信模式先进行归约操作，然后再进行广播操作。Panda 提出了 gathering 和 broadcasting worm 设计^[234] 支持栅栏同步通信。包括 NYU Ultracomputer^[227]、CM-5^[228]、Cray T3D^[229]、Blue Gene/L^[230] 和天河 1A^[231] 在内的许多巨型机都采用了专用或优化的网络实现栅栏消息的组合。Oh 等观察到在众核结构上实现专用的栅栏通信网络会比较昂贵，因此，他们提出采用片上传输线技术高效支持多个同时的栅栏操作^[232]。本章主要研究 cache 一致性协议中的聚合通信，这些聚合通信先执行多播操作，再执行 ACK 报文的归约操作。Bolotin 等提出对 ACK 报文进行组合可能会给性能带来好处^[238]，但是他们没有给出一个详细的设计或评估。Krishna 等提出了一种高效的设计来支持 cache 一致性协议中的聚合通信^[73]。本章所设计的消息组合框架与他们的设计有很大不同：本章设计使用联想存储器（MCT 表格）记录 ACK 报文的到达信息，而 Krishna 等则将先到达的 ACK 报文保存在虚通道内以等待之后到达的 ACK 报文，并进行报文组合^[73]。

6.7.2 片上网络多播路由算法

近几年，人们对片上网络多播路由算法进行了大量的研究。Lu 等使用基于路径的多播路由^[68]：多播报文发送前需要先发送路径建立请求和并等待路径建立回复消息。这种设计的延迟较大，基于树结构的多播路由能降低延迟开销。Enright Jerger 等提出的 VCTM 多播路由算法引入了虚拟多播树的概念^[69]；bLBDR 设计使用小区域内部的广播来实现多播操作^[65]；RPM 算法主要研究提高网络带宽的有效使用率^[70]；基于 bLBDR 的区域隔离机制，Wang 等将 RPM 算法扩展到非规则区域^[71]；MRR 是一种基于 rotary router^[42] 的多播路由算法^[72]。Whirl 能够有效支持广播和密集多播操作^[73]，其死锁避免原理和本章设计的 BAM 相同：两者都采用了 Duato 的单播路由死锁避免理论^[162]。BAM 和 RPM 都采用位编码方式编码多播目标节点，这种编码不能扩展到更大规模的网络上。Wang 等提出了一种

压缩技术能增加这种编码的可扩展性^[243]；此外，类似于 cache 目录中采用的粗粒度位向量设计^[244] 也可以增加这种编码的可扩展性。

6.8 本章小结

本章主要研究了对 cache 一致性协议中归约和多播通信的硬件支持设计。高效的片上网络设计应该能够智能处理网络中的通信，为提高性能和降低功耗，必须要消除不必要的或冗余的网络消息。本章所提出的消息组合框架正好实现了这一点。该框架将多播报文所产生的多个 ACK 报文进行组合，组合操作需要在路由器上增加一个 64 表项的联想存储器。此外，本章还提出了均衡自适应多播路由算法，该算法能使不同维度上的缓存资源保持均衡，带来了一定的性能提升。实验结果表明，消息组合框架不仅在低到中等负载情况下降低了 14.1% 的平均网络延迟，同时也提高了 9.6% 的饱和吞吐率。当虚通道数更少、网络规模更大、多播报文比例更高或多播报文目标节点数更多时，消息组合框架带来的性能提升更明显。此外，均衡自适应多播路由算法所使用的均衡缓存配置能进一步带来 13.8% 的饱和吞吐率提升。本章所提出的消息组合框架可以进行扩展以支持令牌一致性协议或其它并行结构上的归约通信。

第七章 结束语

本章对全文进行总结，并展望未来的研究工作。

7.1 工作总结

协同设计 cache 一致性协议和片上网络是一个重要的研究方向。本文基于众核平台 cache 一致性协议的层次结构、通信特征和通信模式对片上网络进行了优化设计。本文观察到 cache 一致性协议的四个特征：1. 层次协议结构；2. 通信中包含大量短报文；3. 需要同时传输长报文和短报文；4. 需要使用归约和多播等聚合通信。基于这四方面的特征，本文优化设计了片上网络的路由算法和流控机制。具体而言，本文贡献包括如下四方面：

(1). 提出了面向负载整合工作模式的路由算法

层次 cache 一致性协议结构和应用程序有限的并行度导致多应用程序将同时运行在一个众核计算平台上，这种负载整合工作模式对路由算法设计提出了新的要求。本研究首先对已有设计的局限性进行了分析，这些设计包括局部自适应路由算法和全局自适应路由算法。局部自适应路由算法对网络状态的短视性限制了其性能，而全局自适应路由算法的性能则受限于干扰信息，在负载整合模式下，这些干扰信息耦合了多个同时运行的应用程序。

本研究针对这两方面的问题，提出了基于目标的自适应路由算法，该路由算法将目标节点的位置信息集成到输出端口选择过程中。通过使用一种低开销的拥塞信息传播网络，基于目标的自适应路由算法一方面提供了较高的适应性，另一方面也为应用程序提供了动态隔离性。应用程序之间的动态隔离性在网络中同时运行多个应用程序时至关重要，它是保证应用程序性能可预测性的关键。实验结果表明，基于目标的自适应路由算法在许多网络配置下都获得了较好的性能，该路由算法的额外连线资源需求比较低，同时其降低了能量延迟积。

(2). 提出了一种面向 cache 一致性通信的完全自适应路由算法

由于芯片面积、功耗和频率的限制，片上网络只能配置少量的虚通道，虚通道数目受限环境给完全自适应路由算法设计提出了新的挑战。基于已有的虫孔网络死锁避免机制设计的完全自适应路由算法需要采用保守虚通道分配策略，该策略会给具有大量短报文的 cache 一致性片上网络带来较大的性能损失。因此，本研究提出了面向虫孔交换网络完全自适应路由算法的全报文发送虚通道分配策略。该策略允许多个报文同时存在于一条虚通道内部，从而在虚通道受限的环境中带来了很大的性能提升。全报文发送策略的提出是基于 cache 一致性片上网络

传输的报文大部分长度较短这个观察。本研究证明如果路由算法在采用保守虚通道分配策略时没有死锁，则其采用全报文发送策略同样也不会出现死锁。因此，全报文发送策略对之前的死锁避免理论进行了扩展。

基于全报文发送策略，本研究进一步给出了一种完全自适应路由算法设计，该设计能够在较低硬件开销情况下提供较高的路由灵活性。与保守虚通道分配策略相比，全报文发送策略在合成流量模式下平均获得了 88.9% 的饱和吞吐率提升。在网络负载较重的 PARSEC 应用程序中，全报文发送策略最高获得了 37.8% 的全系统性能提升。与保守策略相比，全报文发送策略能够在使用一半的缓存资源或者虚通道数目时获得类似的性能。

(3). 提出了面向 torus 片上网络的切片气泡流控机制

Cache 一致性片上网络需要同时传输长报文和短报文，已有 torus 网络死锁避免不能高效处理这种混合长度报文的传输。在 torus 网络保持无死锁的前提下提高缓存的利用率是一个重要的研究问题。本研究分析了已有设计在这方面的局限，并进一步提出了切片气泡流控 torus 网络死锁避免理论。切片气泡流控通过在每个 ring 上维持一个空闲缓存单元实现死锁避免，切片气泡流控只需使用一条虚通道，因此其路由器频率比 dateline 高 30%。与之前的报文气泡机制不同，切片气泡流控不需要将短报文视为长报文，因此提高了缓存利用率。

基于这个理论，本研究给出了两种实现：本地切片气泡策略（FBFC-L）和关键气泡策略（FBFC-C）。FBFC-L 的最少缓存需求比已有的最好设计多一个缓存单元，FBFC-C 的需求与已有的最好设计相同。在 PARSEC 测试集中，FBFC 相对于 LBS 的平均性能提升是 13.0%，其最大性能提升是 22.7%。随着缓存数量的减少，FBFC 的性能优势更为明显。在合成流量模式下，FBFC 使用 CBS 的一半缓存资源时能够获得与其类似的性能；同时在 PARSEC 测试集下，FBFC 使用 LBS 的一半缓存资源时能够获得与其相当的性能。

(4). 提出了一种高效支持 cache 一致性协议归约和多播通信的技术

Cache 一致性协议需要使用包括归约和多播通信在内的聚合通信，为了防止这些重要的聚合通信成为系统性能的瓶颈，必须要对它们提供硬件支持。高效的片上网络设计应该能够采用智能方式处理网络中的通信，为了提高性能和降低功耗，它们必须要消除归约通信中不必要的或冗余的网络消息。本研究所提出的消息组合框架正好实现了这一点，这个组合框架将一个多播报文所产生的归约多个 ACK 报文在传输过程中进行组合。本研究在每个路由器上增加了一个低开销的 64 表项的联想存储器以实现报文的组合操作。此外，本研究还提出了均衡自适应多播路由算法，其能够使不同维度上的缓存资源保持均衡，因此带来了一定的性能提升。

实验结果表明，所提出的消息组合框架不仅在低到中等负载情况下降低了 14.1% 的平均网络延迟，同时也提高了 9.6% 的饱和吞吐率。当网络中配置更少的虚通道数、使用更大规模的网络、更高的多播报文比例或更高的多播报文目标节点数时，消息组合框架带来的性能提升更明显。均衡自适应多播路由算法的均衡缓存配置能够额外带来 13.8% 的饱和吞吐率提升。所提出的消息组合框架可以方便扩展以支持诸如令牌一致性协议或其它并行结构上的归约通信。

7.2 研究展望

本文紧紧围绕“面向 cache 一致性通信优化片上网络设计”这一目标，基于 cache 一致性协议的层次结构、通信报文长度分布和通信模式需求，对片上网络的路由算法和流控机制进行了优化设计。所提出的优化设计不仅取得了一定的系统性能提升和硬件开销下降，同时也对已有完全自适应路由算法死锁避免理论和 torus 网络死锁避免理论进行了扩展。因此，本文既具备一定的工程价值，又具备一定的理论意义。今后将在以下几个方面进一步开展研究：

(1). 基于目标的网络源节流技术设计

本文针对负载整合模式提出了一种基于报文目标的自适应路由算法输出端口选择策略，该设计的思想可以被扩展到源节流技术设计上。在自适应路由算法中，当注入率高于网络支持的饱和吞吐率时，网络性能会出现下降，此时需要采用源节流技术以维持网络的性能。之前存在的研究采用了类似于区域拥塞感知路由算法的设计思想，它们基于全网络的拥塞状态做出每个源节点是否需要采用节流技术的决策。这种设计在负载整合模式下遇到与区域拥塞感知路由算法类似的问题，即决策所考虑的网络状态中包含冗余信息，这些冗余信息会耦合多应用程序的性能。因此可以采用一种基于报文目标的源节流技术消除这些冗余信息，从而更好地支持负载整合工作模式。

(2). 消息传递众核平台路由算法和流控机制的优化

本文针对 cache 一致性众核平台的结构和通信特征对片上网络的路由算法和流控机制进行了优化，消息传递机制也是一种重要的众核平台编程模型。在这种消息传递众核结构下，片上网络传输的通信由所采用的消息传递机制决定。消息传递机制与 cache 一致性机制产生的通信特征和所需求的通信模式有较大区别。因此，一种高效的支持消息传递机制的片上网络设计需要仔细考察该机制产生的通信特征，并基于这些特征对片上网络设计进行优化。本文所采用的一些方法学，包括统计报文的长度分布情况、每个虚拟网络传输的报文长度以及观察聚合通信对性能的影响都可以被应用到研究消息传递机制的通信特征上。与本文类似，基于消息传递平台通信特征设计的片上网络也应该能取得一定的系统性能提

升和硬件开销下降。

(3). 基于切片气泡流控的完全自适应路由算法设计

本文采用所提出的切片气泡流控机制设计了 torus 网络上的确定性路由算法，该设计需要使用一些额外的连线资源处理饿死现象。一种消除这些额外饿死处理连线资源的设计是采用完全自适应路由算法设计。在完全自适应路由算法设计中，逃逸虚通道上采用所提出的切片气泡流控机制，而普通虚通道则没有报文注入限制，因此，不会出现饿死现象。同时，所提出的切片气泡流控机制可以通过控制每个 ring 上的关键气泡数目支持源节流技术，从而在注入率高于饱和吞吐率时维持完全自适应路由算法的性能。基于切片气泡流控的完全自适应路由算法可以综合使用本文第4章中所提出的全报文发送策略允许一条虚通道同时容纳多个报文，从而进一步提高虚通道的利用率。

(4). 栅栏同步聚合通信的支持技术

本文针对 cache 一致性协议中的多播和归约通信提供了硬件支持，所提出的消息组合框架动态将传输中的 ACK 报文进行组合，从而降低了网络负载，取得了报文传输延迟、饱和吞吐率和网络功耗的性能提升。类似的，对栅栏同步中的归约通信进行消息组合也能取得这些性能提升。栅栏同步中的归约通信与 cache 一致性协议中的归约通信的一个区别是：cache 一致性协议的归约通信是在多播通信之后进行的，因此，可以利用多播通信已经建立起的逻辑树完成归约消息的组合。但是栅栏同步是先进行归约通信，之后进行多播通信，因此，需要考虑如何保证这些归约消息能够经过网络中的一些共同节点，从而给予它们进行消息组合的机会。

致 谢

博士论文写到致谢，心中感慨万千。回首往事，保持一颗感恩的心是不断前进的动力。

首先，向我的指导老师王志英教授致以崇高的敬意和深深的感激。从 2005 年进入师门以来，王老师不仅对我的科研、学习和生活非常关心，而且总是给学生创造机会，这些机会的宝贵性足以影响我的一生。王老师高屋建瓴地为我选定研究方向，并且在我对研究方向感到迷惑困顿时给予我鼓励。每次我向王老师汇报一个小的研究想法的时候，老师您总是鼓励我瞄准前沿，从大处着眼、小处着手。当我对写作感到无从下手时，老师总是在自己丰富经验和渊博知识的基础上，对文章结构给予我最重要的指导。每次论文被拒我感到沮丧时，老师您总是鼓励我坚持，继续修改，继续投稿。每次当取得一点小小的成就时，老师您总是非常高兴。老师对学生的生活给予了无微不至的关怀，尤其我身在海外期间感受更为明显，能在您门下攻读博士学位是我最大的幸运，师恩点滴，学生铭记在心。

衷心感谢戴葵老师在本科和硕士期间对我的指导，从戴老师身上初次学习到科研探索的过程，同时老师也教给了我很多为人处世和思考解决问题的方法。衷心感谢师门的赖明澈师兄和黄立波师兄，你们在我研究生期间给予我大量的指导，并分享了自己大量宝贵的经验。两位师兄一直是我学习的榜样。

非常感谢师门沈立老师，老师您从本科开始一直对我的科研工作非常关心，感谢您在国外期间也时常抽出时间对我进行一些指导。非常感谢师门陆弘毅老师给予我的帮助。非常感谢肖依老师对我科研工作进展情况的关心。非常感谢张春元老师对本文结构提出的意见，与您的讨论开阔了本文写作的思路。

感谢师门任江春老师、刘芳老师、王蕾老师、龚锐师兄、郭建军师兄、吕雅帅师兄、陈微师姐、石伟师兄、陈芳园师姐、马俊师兄、伍江江师兄、赖鑫师兄、何蕾师姐、朱耀凯师兄等。感谢师门甘新标、王友瑞、刘聪、任洪广、梅松竹、程勇、苏博、朱琪、林正毅、李家文、张开、陈向、任珊珊、周洁、徐帆、郑重、陈项灏等同学。感谢大家一起营造的师门良好的科研和生活环境。感谢师门师兄、同学和师弟们对我的信任，让我担任师门足球队压力最大的位置：守门员。

非常感谢赵丹、姚路、曹维和甘新标同学在我离开长沙期间无数次地帮助我。感谢同在多伦多学习的戴泽福师兄、陈迅师兄、黎渊、冷洪泽、肖灿文老师、丁文霞老师、杜湘渝老师、彭元喜老师、李勇老师、卢芳云老师、谢毓香老师、吴学忠老师、乔晋葳等，我们一起在国外营造了一个家一样的环境。感谢学院刘

越老师、姜新文老师、唐玉华老师、赵文涛老师在我求学期间对我关心和帮助。

感谢朝夕相处的赵丹、姚路、曹维、潘志辉、刘权、韦中伟、梅松竹、纪长、刘聪、任洪广、韩彪、刘晓东、王友瑞、甘新标、高潭、方旭东、刘闻坚、王涛、陈建军、罗准辰、张湘莉兰、张硕等同学。感谢一队 03 级、五队 07 级和七队 09 级的同学，我们一起经历了科大的生活，并留下了许多美好的回忆。

特别感谢我在多伦多大学访学期间的合作教授 Natalie Enright Jerger，您刻苦钻研的工作态度、勇攀高峰的积极精神、严谨务实的专业作风为我树立了良好的楷模，也让我受益匪浅。感谢 Enright Jerger 教授课题组的冯恺、代文博、Robert Hesse、Parisa Khadem Hamedani、Steven Gurinkel、Mario Badr、Latch Dimitrov、Sam Vafaei 多次阅读我的文章初稿，并且给予宝贵的修改意见。感谢多伦多大学的陈谨师姐对我的照顾，您的帮助让我很快适应了多伦多的生活。感谢 David Han 和 Tim Liu 同学多次在我遇到挫折时给予一些宝贵的安慰。感谢多伦多大学的 Paul Chow 教授、Andreas Moshovos 教授、Jianwen Zhu 教授和 Greg Steffan 教授对我的照顾和帮助。

感谢 Stanford 大学 CVA 小组的 Daniel Becker 和 Nan Jiang 对我的帮助，尤其是 Daniel Becker 耐心细致地讲解您的开源 RTL router 的结构，并对我的文章初稿给出详细而宝贵的建议。感谢 KAIST 的 John Kim 教授对我请教问题的耐心回复。感谢南加州大学的 L zhong Chen 和 Timothy Pinkston 教授与我分享一些尚未发表的设计细节。感谢华中科技大学的邹雪城老师、郑朝霞老师、吴丹师姐和陈攀同学在华中科技大学期间对我的帮助。感谢中科院计算所的付斌章同学、韩银和老师、鄢贵海老师、包云岗老师，在开会时与你们的讨论开阔了我的研究思路。

感谢学院、学员大队、学员队各级领导对我的教育、关心和帮助，你们的辛勤工作为我们创造了良好的学习和生活环境。

非常感谢永州四中的盘明媚老师，您的帮助给了我一次追梦的机会。感谢杨晔同学陪我一起走过的一段岁月。

感谢我的母亲父亲，你们含辛茹苦将我抚养成人，并教会了我很多人生道理，再多的文字也无法表达对你们的感激。多年成长经历中有太多的人需要感谢，纸短言长，我再次对一直关心帮助支持鼓励我的人说声谢谢！

最后，向在百忙之中能够抽出时间对我的论文进行评审并提出宝贵意见的各位专家和教授致以诚挚的谢意！

参考文献

- [1] Dally W, Towles B. Route packets, not wires: on-chip interconnection networks [C]. In DAC 2001.
- [2] Jantsch H A, et al. Network on chip: An architecture for billion transistor era [C]. In NorChip 2000.
- [3] Benini L, De Micheli G. Powering networks on chips: energy-efficient and reliable interconnect design for SoCs [C]. In Proceedings of the 14th international symposium on Systems synthesis.
- [4] Culler D, Singh J, Gupta A. Parallel computer architecture: a hardware/software approach [M]. Morgan Kaufmann, 1999.
- [5] Martin M M K, Hill M D, Sorin D J. Why on-chip cache coherence is here to stay [J]. Commun. ACM. 2012, 55 (7): 78–89.
- [6] Sorin D, Hill M, Wood D. A Primer on Memory Consistency and Cache Coherence [M]. 1st ed. Morgan & Claypool Publishers, 2011.
- [7] Enright Jerger N, Peh L. On-Chip Networks [M]. 1st ed. Morgan & Claypool Publishers, 2009.
- [8] Agarwal N, Peh L, Jha N. In-network snoop ordering (INSO): Snoopy coherence on unordered interconnects [C]. In HPCA 2009.
- [9] Hennessy J L, Patterson D A. Computer Architecture: A Quantitative Approach [M]. 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.
- [10] Moore G. Moore's law [J]. The New Hacker's Dictionary. 1965, 121.
- [11] Wittenbrink C, Kilgariff E, Prabhu A. Fermi GF100 GPU architecture [J]. Micro, IEEE. 2011, 31 (2): 50–59.
- [12] ITRS. International Technology Roadmap for Semiconductors, 2009 Edition. <http://www.itrs.net>. 2009.
- [13] Patterson D, Hennessy J. Computer organization and design: the hardware/software interface [M]. Morgan Kaufmann, 2009.
- [14] Hinton G, Sager D, Upton M, et al. The microarchitecture of the Pentium® 4 processor [C]. In Intel Technology Journal. 2001.
- [15] Burger D, Goodman J. Billion-transistor architectures: There and back again [J]. Computer. 2004, 37 (3): 22–28.

-
- [16] Sankaralingam K, Nagarajan R, Liu H, et al. Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture [C]. In Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on. 2003: 422–433.
 - [17] Kahle J A, et al. Introduction to the Cell multiprocessor [J]. IBM J. Res. Dev. 2005, 49 (4.5): 589 –604.
 - [18] Taylor M, Psota J, Saraf A, et al. Evaluation of the Raw microprocessor: An exposed-wire-delay architecture for ILP and streams [C]. In ISCA 2004.
 - [19] Hammond L, Hubbert B, Siu M, et al. The stanford hydra cmp [J]. Micro, IEEE. 2000, 20 (2): 71–84.
 - [20] Vangal S, et al. An 80-tile 1.28 TFLOPS network-on-chip in 65nm CMOS [C]. In ISSCC 2007.
 - [21] del Cuvillo J, Zhu W, Hu Z, et al. Toward a software infrastructure for the cyclops-64 cellular architecture [C]. In HPCS 2006.
 - [22] Wentzlaff D, et al. On-Chip Interconnection Architecture of the Tile Processor [J]. Micro, IEEE. 2007, 27 (5): 15 –31.
 - [23] Sterling T. Multicore: the New Moore’s Law [C]. In Invited Presentation to ICS 2007.
 - [24] Lu Z, Jantsch A. Trends of terascale computing Chips in the next ten years [C]. In ASICON 2009.
 - [25] Guerrier P, Greiner A. A generic architecture for on-chip packet-switched interconnections [C]. In DATE 2000.
 - [26] Adriahtenaina A, et al. SPIN: A Scalable, Packet Switched, On-Chip Micro-Network [C]. In DATE 2003.
 - [27] Balfour J, Dally W. Design tradeoffs for tiled CMP on-chip networks [C]. In ICS 2006.
 - [28] Kim J, Balfour J, Dally W. Flattened Butterfly Topology for On-Chip Networks [C]. In MICRO 2007.
 - [29] Grot B, et al. Express Cube Topologies for on-Chip Interconnects [C]. In HPCA 2009.
 - [30] Bourduas S, Zilic Z. Modeling and evaluation of ring-based interconnects for Network-on-Chip [J]. J. Syst. Archit. 2011, 57 (1): 39–60.
 - [31] Fallin C, et al. A High-Performance Hierarchical Ring On-Chip Interconnect with Low-Cost Routers [C]. In SAFARI Technical Report. 2011.
-

-
- [32] Mirza-Aghatabar M, et al. An Empirical Investigation of Mesh and Torus NoC Topologies Under Different Routing Algorithms and Traffic Models [C]. In DSD 2007.
 - [33] Mishra A K, Vijaykrishnan N, Das C R. A case for heterogeneous on-chip interconnects for CMPs [C]. In ISCA 2011.
 - [34] Shin M, Kim J. Leveraging torus topology with deadlock recovery for cost-efficient on-chip network [C]. In ICCD 2011.
 - [35] Manevich R, Walter I, Cidon I, et al. Best of both worlds: A bus enhanced NoC (BENoC) [C]. In NOCS 2009.
 - [36] Das R, et al. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs [C]. In HPCA 2009.
 - [37] Zhao H, et al. A hybrid NoC design for cache coherence optimization for chip multiprocessors [C]. In DAC 2012.
 - [38] Peh L-S, Dally W. A delay model and speculative architecture for pipelined routers [C]. In HPCA 2001.
 - [39] Mullins R, West A, Moore S. Low-latency virtual-channel routers for on-chip networks [C]. In ISCA 2004.
 - [40] Matsutani H, et al. Prediction router: Yet another low latency on-chip router architecture [C]. In HPCA 2009.
 - [41] Kumar A, et al. A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS [C]. In ICCD 2007.
 - [42] Abad P, et al. Rotary router: an efficient architecture for CMP interconnection networks [C]. In ISCA 2007.
 - [43] Kim J. Low-cost router microarchitecture for on-chip networks [C]. In MICRO 2009.
 - [44] Becker D U, Dally W J. Allocator implementations for network-on-chip routers [C]. In SC 2009.
 - [45] Fallin C, et al. CHIPPER: A low-complexity bufferless deflection router [C]. In HPCA 2011.
 - [46] Dimitrakopoulos G, Galanopoulos K. Switch allocator for bufferless network-on-chip routers [C]. In INA-OCMC 2011.
 - [47] Michelogiannakis G, Jiang N, Becker D, et al. Packet chaining: efficient single-cycle allocation for on-chip networks [C]. In MICRO 2011.
-

-
- [48] Hayenga M, Lipasti M. The NoX router [C]. In MICRO 2011.
 - [49] Nicopoulos C, et al. ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers [C]. In MICRO 2006.
 - [50] Xu Y, et al. Simple virtual channel allocation for high throughput and high frequency on-chip routers [C]. In HPCA 2010.
 - [51] Ramanujam R S, et al. Design of a High-Throughput Distributed Shared-Buffer NoC Router [C]. In NOCS 2010.
 - [52] Ahmadiania A, Shahrabi A. A Highly Adaptive and Efficient Router Architecture for Network-on-Chip [J]. Comput. J. 2011, 54 (8): 1295–1307.
 - [53] Becker D, et al. Adaptive Backpressure: Efficient Buffer Management for On-Chip Networks [C]. In ICCD 2012.
 - [54] Michelogiannakis G, Balfour J, Dally W. Elastic-buffer flow control for on-chip networks [C]. In HPCA 2009.
 - [55] Kodi A, Sarathy A, Louri A. iDEAL: Inter-router dual-function energy and area-efficient links for network-on-chip (NoC) architectures [C]. In ISCA 2008.
 - [56] Kim G, Kim J, Yoo S. FlexiBuffer: Reducing leakage power in on-chip network routers [C]. In DAC 2011.
 - [57] Hu J, Marculescu R. DyAD - smart routing for networks-on-chip [C]. In DAC 2004.
 - [58] Seo D, et al. Near-optimal worst-case throughput routing for two-dimensional mesh networks [C]. In ISCA 2005.
 - [59] Singh A, et al. GOAL: a load-balanced adaptive routing algorithm for torus networks [C]. In ISCA 2003.
 - [60] Kim J, et al. A low latency router supporting adaptivity for on-chip interconnects [C]. In DAC 2005.
 - [61] Li M, Zeng Q-A, Jone W-B. DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip [C]. In DAC 2006.
 - [62] Ascic G, et al. Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip [J]. Computers, IEEE Transactions on. 2008, 57 (6): 809 –820.
 - [63] Gratz P, Grot B, Keckler S. Regional congestion awareness for load balance in networks-on-chip [C]. In HPCA 2008.
-

-
- [64] Ramanujam R S, Lin B. Destination-based adaptive routing on 2D mesh networks [C]. In ANCS 2010.
 - [65] Rodrigo S, et al. Efficient unicast and multicast support for CMPs [C]. In MICRO 2008.
 - [66] Zhang Z, Greiner A, Taktak S. A reconfigurable routing algorithm for a fault-tolerant 2D-Mesh Network-on-Chip [C]. In DAC 2008.
 - [67] Kinsy M A, et al. Application-aware deadlock-free oblivious routing [C]. In ISCA 2009.
 - [68] Lu Z, Yin B, Jantsch A. Connection-oriented multicasting in wormhole-switched networks on chip [C]. In IEEE Symp. on Emerging VLSI Tech. and Architectures.
 - [69] Enright Jerger N, Peh L-S, Lipasti M. Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support [C]. In ISCA 2008.
 - [70] Wang L, et al. Recursive Partitioning Multicast: A bandwidth-efficient routing for Networks-on-Chip [C]. In NOCS 2009.
 - [71] Wang X, et al. On an efficient NoC multicasting scheme in support of multiple applications running on irregular sub-networks [J]. Microprocess. Microsyst. 2011, 35: 119–129.
 - [72] Abad P, Puente V, Gregorio J-A. MRR: Enabling fully adaptive multicast routing for CMP interconnection networks [C]. In HPCA 2009.
 - [73] Krishna T, Peh L-S, Beckmann B M, et al. Towards the Ideal On-chip Fabric for 1-to-Many and Many-to-1 Communication [C]. In MICRO 2011.
 - [74] Kang Y H, Sondeen J, Draper J. Multicast routing with dynamic packet fragmentation [C]. In GLSVLSI 2009.
 - [75] Peh L, Dally W. Flit-reservation flow control [C]. In HPCA 2000.
 - [76] Kumar A, et al. Express Virtual Channels: Towards the Ideal Interconnection Fabric [C]. In ISCA 2007.
 - [77] Kumar A, Peh L-S, Jha N K. Token flow control [C]. In MICRO 2008.
 - [78] Lu Z, Liu M, Jantsch A. Layered switching for networks on chip [C]. In DAC 2007.
 - [79] Enright Jerger N, Peh L, Lipasti M. Circuit-switched coherence [C]. In NOCS 2008.
 - [80] Samman F, et al. Wormhole cut-through switching: Flit-level messages interleaving for virtual-channelless network-on-chip [J]. Microprocess. Microsyst. 2011, 35 (3): 343–358.
-

-
- [81] Concer N, Petracca M, Carloni L P. Distributed flit-buffer flow control for networks-on-chip [C]. In CODES+ISSS 2008.
 - [82] Chen L, et al. Critical Bubble Scheme: An Efficient Implementation of Globally Aware Network Flow Control [C]. In IPDPS 2011.
 - [83] Joshi A, Mutyam M. Prevention flow-control for low latency torus networks-on-chip [C]. In NOCS 2011.
 - [84] Cheng L, et al. Interconnect-aware coherence protocols for chip multiprocessors [C].
 - [85] Muralimanohar N, Balasubramonian R. Interconnect design considerations for large NUCA caches [C].
 - [86] Eisley N, Peh L, Shang L. In-network cache coherence [C]. In MICRO 2006.
 - [87] Agarwal N, Peh L, Jha N. In-network coherence filtering: snoopy coherence without broadcasts [C]. In MICRO 2009.
 - [88] Enright Jerger N. SigNet: Network-on-chip filtering for coarse vector directories [C]. In DATE 2010.
 - [89] Hu J, Marculescu R. Energy- and performance-aware mapping for regular NoC architectures [J]. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on. 2005, 24 (4): 551 – 562.
 - [90] Chou C-L, Ogras U, Marculescu R. Energy- and Performance-Aware Incremental Mapping for Networks on Chip With Multiple Voltage Levels [J]. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on. 2008, 27 (10): 1866 –1879.
 - [91] Chou C-L, Marculescu R. Run-Time Task Allocation Considering User Behavior in Embedded Multiprocessor Networks-on-Chip [J]. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on. 2010, 29 (1): 78 –91.
 - [92] Lei T, Kumar S. A two-step genetic algorithm for mapping task graphs to a network on chip architecture [C]. In DSD 2003.
 - [93] Grot B, Hestness J, Keckler S, et al. Kilo-NOC: a heterogeneous network-on-chip architecture for scalability and service guarantees [C]. In ISCA 2011.
 - [94] Lee J, Ng M, Asanovic K. Globally-synchronized frames for guaranteed quality-of-service in on-chip networks [C]. In ISCA 2008.
 - [95] Grot B, Keckler S, Mutlu O. Preemptive virtual clock: a flexible, efficient, and cost-effective QOS scheme for networks-on-chip [C]. In MICRO 2009.
-

- [96] Ouyang J, Xie Y. Loft: A high performance network-on-chip providing quality-of-service support [C]. In MICRO 2010.
- [97] Qian Y, Lu Z, Dou W. Analysis of worst-case delay bounds for on-chip packet-switching networks [J]. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on. 2010, 29 (5): 802–815.
- [98] Qian Y, Lu Z, Dou W. Applying network calculus for performance analysis of self-similar traffic in on-chip networks [C]. In CODES+ISSS 2009.
- [99] Carloni L, Pande P, Xie Y. Networks-on-chip in emerging interconnect paradigms: Advantages and challenges [C]. In NOCS 2009.
- [100] Jose A, Patounakis G, Shepard K. Pulsed current-mode signaling for nearly speed-of-light intrachip communication [J]. Solid-State Circuits, IEEE Journal of. 2006, 41 (4): 772–780.
- [101] Kim B, Stojanovic V. Equalized interconnects for on-chip networks: Modeling and optimization framework [C]. In ICCAD 2007.
- [102] Kirman N, Kirman M, Dokania R, et al. Leveraging optical technology in future bus-based chip multiprocessors [C]. In MICRO 2006.
- [103] Shacham A, Bergman K, Carloni L. The case for low-power photonic networks on chip [C]. In DAC 2007.
- [104] Chang M, et al. CMP network-on-chip overlaid with multi-band RF-interconnect [C]. In HPCA 2008.
- [105] Vandeveld B, et al. Thermo-mechanics of 3D-wafer level and 3D stacked IC packaging technologies [C]. In EuroSimE 2008.
- [106] Mishra A, Dong X, Sun G, et al. Architecting on-chip interconnects for stacked 3D STT-RAM caches in CMPs [C]. In HPCA 2011.
- [107] Park D, Eachempati S, Das R, et al. MIRA: A multi-layered on-chip interconnect router architecture [C]. In ISCA 2008.
- [108] Hopkins D, et al. Circuit techniques to enable 430 GB/s/mm² proximity communication [C]. In ISSCC 2007.
- [109] Gratz P, et al. On-Chip Interconnection Networks of the TRIPS Chip [J]. Micro, IEEE. 2007, 27 (5): 41 –50.
- [110] Liang J, Swaminathan S, Tessier R. aSOC: A scalable, single-chip communications architecture [C]. In PACT 2000.

- [111] Millberg M, et al. The Nostrum backbone-a communication protocol stack for networks on chip [C]. In VLSI Design 2004.
- [112] Dall’Osso M, et al. Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs [C]. In ICCD 2003.
- [113] Bolotin E, et al. QNoC: QoS architecture and design process for network on chip [J]. Journal of Systems Architecture. 2004, 50 (2): 105–128.
- [114] Bjerregaard T, Sparso J. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip [C]. In DATE 2005.
- [115] Park S, et al. Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45nm SOI [C]. In DAC 2012.
- [116] Steenhof F, et al. Networks on chips for high-end consumer-electronics TV system architectures [C]. In DATE 2006.
- [117] Coppola M, et al. Design of Cost-Efficient Interconnect Processing Units: Spider-gon STNoC [M]. CRC, 2008.
- [118] Howard J, et al. A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS [C]. In ISSCC 2010.
- [119] Agarwal A, et al. Tile processor: Embedded multicore for networking and multimedia [C]. In Hot Chips 2007.
- [120] Seventh ACM/IEEE International Symposium on Networks-on-Chip. <http://nocsymposium.org/>. 2013.
- [121] 林世俊, 张凡, 金德鹏, 等. 分布式同步的 GALS 片上网络及其接口设计 [J]. 清华大学学报: 自然科学版. 2008, 48 (1): 32–35.
- [122] 马立伟, 孙义和. 片上网络拓朴优化: 在离散平面上布局与布线 [J]. 电子学报. 2007, 35 (5): 906–911.
- [123] 王宏伟, 陆俊林, 佟冬, 等. 层次化片上网络结构的簇生成算法 [J]. 电子学报. 2007, 35 (5): 916–920.
- [124] Yu Z, You K, Xiao R, et al. An 800MHz 320mW 16-core processor with message-passing and shared-memory inter-core communication mechanisms [C]. In ISSCC 2012.
- [125] 朱晓静, 胡伟武, 马可, 等. Xmesh: 一个 mesh-like 片上网络拓扑结构 [J]. 软件学报. 2007, 18 (9): 2194–2204.
- [126] 朱晓静. 片上网络的结构设计与性能分析 [D]. 合肥: 中国科学技术大学, 2008.

- [127] 张磊, 李华伟, 李晓维. 用于片上网络的容错通信算法 [J]. 计算机辅助设计与图形学学报. 2007, 19 (4): 508–514.
- [128] 黄琨, 马可, 曾洪博, 等. 一种分片式多核处理器的用户级模拟器 [J]. 软件学报. 2008, 19 (4): 1069–1080.
- [129] 付方发, 张庆利, 王进祥, 等. 支持多种流量分布的片上网络性能评估技术研究 [J]. 哈尔滨工业大学学报. 2007, 39 (5): 830–834.
- [130] 李磊. 片上网络 NoC 的通信研究 [D]. 杭州: 浙江大学, 2007.
- [131] 武畅. 片上网络体系结构和关键通信技术研究 [D]. 西安: 电子科技大学, 2008.
- [132] 常政威. 网络化 MPSoC 高能效设计技术研究 [D]. 西安: 电子科技大学, 2009.
- [133] 荆元利. 基于片上网络的系统芯片研究 [D]. 西安: 西北工业大学, 2005.
- [134] 肖翔, 董渭清, 文敏华. 网环步进码片上网络自适应路由算法设计 [J]. 西安交通大学学报. 2009, 43 (012): 70–74.
- [135] 唐杉. 基于片上网络互联的 SoC 调试技术研究 [D]. 北京: 北京邮电大学, 2008.
- [136] 赵宏智. 2D Mesh 片上网络中交换机服务性能影响的研究及其拓扑改进 [J]. 电子学报. 2009, 37 (2): 294–298.
- [137] 杨盛光, 李丽, 高明伦, 等. 面向能耗和延时的 NoC 映射方法 [J]. 电子学报. 2008, 36 (5): 937–942.
- [138] 段新明. 面向 NoC 的无死锁路由算法的研究 [D]. 天津: 南开大学, 2007.
- [139] 陶海洋. 片上网络低能耗和低延迟研究 [D]. 长沙: 湖南大学, 2009.
- [140] 朱兵. 基于片上网络的通信路由方法研究 [D]. 合肥: 合肥工业大学, 2009.
- [141] 刘祥远. 多核 SoC 片上网络关键技术研究 [D]. 长沙: 国防科学技术大学, 2007.
- [142] 钱悦. 片上网络演算模型及性能分析 [D]. 长沙: 国防科学技术大学, 2010.
- [143] Li Z, Zhu C, Shang L, et al. Transaction-aware network-on-chip resource reservation [J]. Computer Architecture Letters. 2008, 7 (2): 53–56.
- [144] Fu B, et al. An abacus turn model for time/space-efficient reconfigurable routing [C]. In ISCA 2011.

- [145] Lai M, Wang Z, Gao L, et al. A dynamically-allocated virtual channel architecture with congestion awareness for on-chip routers [C]. In DAC 2008.
- [146] Shi W, Xu W, Ren H, et al. A novel shared-buffer router for network-on-chip based on Hierarchical Bit-line Buffer [C]. In ICCD 2011.
- [147] 国家自然科学基金委员会. <http://isis.nsfc.gov.cn/>.
- [148] Ma S, Enright Jerger N, Wang Z. DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip [C]. In ISCA 2011.
- [149] Ma S, Enright Jerger N, Wang Z, et al. Holistic Routing Algorithm Design to Support Workload Consolidation in NoCs [J]. Computers, IEEE Transactions on. 2012, 99 (PrePrints).
- [150] Ma S, Enright Jerger N, Wang Z. Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip [C]. In HPCA. 2012.
- [151] Ma S, Enright Jerger N, Wang Z. Supporting Efficient Collective Communication in NoCs [C]. In HPCA. 2012.
- [152] Damaraju S, et al. A 22nm IA Multi-CPU and GPU System-on-Chip [C]. In ISSCC 2012.
- [153] Kim J, Kim H. Router microarchitecture and scalability of ring topology in on-chip networks [C]. In NoCArc 2009.
- [154] Scott S L, et al. The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus [C]. In Hot Interconnects 1996.
- [155] Adiga N R, et al. Blue Gene/L torus interconnection network [J]. IBM J. Res. Dev. 2005, 49 (2.3): 265 –276.
- [156] Kumar P, et al. Exploring concentration and channel slicing in on-chip network router [C]. In NOCS 2009.
- [157] Michelogiannakis G, Pnevmatikatos D, Katevenis M. Approaching ideal NoC latency with pre-configured routes [C]. In NOCS 2007.
- [158] Valiant L, Brebner G. Universal schemes for parallel communication [C]. In STOC 1981.
- [159] Glass C, Ni L. The Turn Model for Adaptive Routing [C]. In ISCA 1992.
- [160] Chiu G-M. The odd-even turn model for adaptive routing [J]. Parallel and Distributed Systems, IEEE Transactions on. 2000, 11 (7): 729 –738.
- [161] Dally W, Seitz C. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks [J]. IEEE Trans. Comput. 1987.

- [162] Duato J. A new theory of deadlock-free adaptive routing in wormhole networks [J]. Parallel and Distributed Systems, IEEE Transactions on. 1993, 4 (12): 1320–1331.
- [163] Dally W, Towles B. Principles and Practices of Interconnection Networks [M]. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.
- [164] Kermani P, Kleinrock L. Virtual cut-through: a new computer communication switching technique [J]. Computer Networks. 1979, 3: 267–286.
- [165] Dally W J, Seitz C L. The Torus Routing Chip [J]. Distributed Computing. 1986, 1: 187–196.
- [166] Ogras U Y, Hu J, Marculescu R. Key research problems in NoC design: a holistic perspective [C]. In CODES+ISSS 2005.
- [167] Marculescu R, et al. Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives [J]. Trans. Comp.-Aided Des. Integ. Cir. Sys. 2009, 28: 3–21.
- [168] Dally W. Virtual-channel flow control [J]. Parallel and Distributed Systems, IEEE Transactions on. 1992, 3 (2): 194–205.
- [169] Moscibroda T, Mutlu O. A case for bufferless routing in on-chip networks [C]. In ISCA. 2009.
- [170] Hayenga M, Enright Jerger N, Lipasti M. SCARAB: A single cycle adaptive routing and bufferless network [C]. In MICRO 2009.
- [171] Shin K, Daniel S. Analysis and implementation of hybrid switching [C]. In ISCA 1995.
- [172] Stunkel C B, et al. The SP2 high-performance switch [J]. IBM Syst. J. 1995, 34: 185–204.
- [173] Galles M. Spider: a high-speed network interconnect [J]. Micro, IEEE. 1997, 17 (1): 34–39.
- [174] Hirata Y, et al. A variable-pipeline on-chip router optimized to traffic pattern [C]. In NoCArc 2010.
- [175] Gratz P, Sankaralingam K, Hanson H, et al. Implementation and Evaluation of a Dynamically Routed Processor Operand Network [C]. In NOCS 2007.
- [176] Choi B, et al. Denovo: Rethinking hardware for disciplined parallelism [C]. In HotPar 2010.
- [177] Kelm J, et al. Cohesion: An adaptive hybrid memory model for accelerators [J]. Micro, IEEE. 2011, 31 (1): 42–55.

-
- [178] Shah M, et al. UltraSPARC T2: A highly-treaded, power-efficient, SPARC SOC [C]. In ASSCC 2007.
- [179] Butler M, et al. Bulldozer: An approach to multithreaded compute performance [J]. Micro, IEEE. 2011, 31 (2): 6–15.
- [180] Nickolls J, Dally W. The GPU computing era [J]. Micro, IEEE. 2010, 30 (2): 56–69.
- [181] Hansson A, Goossens K, Rădulescu A. Avoiding message-dependent deadlock in network-based systems on chip [J]. VLSI design. 2007, 2007.
- [182] Song Y, Pinkston T. A progressive approach to handling message-dependent deadlock in parallel computer systems [J]. Parallel and Distributed Systems, IEEE Transactions on. 2003, 14 (3): 259–275.
- [183] Gharachorloo K, et al. Architecture and design of AlphaServer GS320 [C]. In AS-PLOS 2000.
- [184] Marty M R, Hill M D. Virtual hierarchies to support server consolidation [C]. In ISCA 2007.
- [185] Neelakantam N, et al. FeS2: A Full-system Execution-driven Simulator for x86 [C]. In Poster presented at ASPLOS 2008.
- [186] Magnusson P S, et al. Simics: A Full System Simulation Platform [J]. Computer. 2002, 35: 50–58.
- [187] Yourst M. PTLsim: A cycle accurate full system x86-64 microarchitectural simulator [C]. In ISPASS 2007.
- [188] Martin M M K, et al. Multifacet’s general execution-driven multiprocessor simulator (GEMS) toolset [J]. SIGARCH Comput. Archit. News. 2005, 33: 92–99.
- [189] Hoskote Y, et al. A 5-GHz Mesh Interconnect for a Teraflops Processor [J]. Micro, IEEE. 2007, 27 (5): 51 –61.
- [190] Ilitzky D, et al. Architecture of the Scalable Communications Core’s Network on Chip [J]. Micro, IEEE. 2007, 27 (5): 62 –74.
- [191] Bell S, et al. TILE64 - Processor: A 64-Core SoC with Mesh Interconnect [C]. In ISSCC 2008.
- [192] Zhuravlev S, Blagodurov S, Fedorova A. Addressing shared resource contention in multicore processors via scheduling [C]. In ASPLOS 2010.
- [193] Mutlu O, Moscibroda T. Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems [C]. In ISCA 2008.

- [194] Schwiebert L, Bell R. Performance tuning of adaptive wormhole routing through selection function choice [J]. *J. Parallel Distrib. Comput.* 2002, 62: 1121–1141.
- [195] Feng W-C, Shin K G. Impact of selection functions on routing algorithm performance in multicomputer networks [C]. In ICS 1997.
- [196] Martínez J C, et al. On the Influence of the Selection Function on the Performance of Networks of Workstations [C]. In ISHPC 2000.
- [197] Dally W J, Aoki H. Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels [J]. *Parallel and Distributed Systems, IEEE Transactions on.* 1993, 4: 466–475.
- [198] Dally W, Seitz C. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks [J]. *Computers, IEEE Transactions on.* 1987, C-36 (5): 547 –553.
- [199] Duato J. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks [J]. *Parallel and Distributed Systems, IEEE Transactions on.* 1995, 6 (10): 1055 –1067.
- [200] Duato J. A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks [J]. *Parallel and Distributed Systems, IEEE Transactions on.* 1996, 7 (8): 841 –854.
- [201] Bienia C, et al. The PARSEC benchmark suite: characterization and architectural implications [C]. In PACT 2008.
- [202] Boyd-Wickizer S, et al. An analysis of Linux scalability to many cores [C]. In OSDI 2010.
- [203] Das R, et al. Performance and power optimization through data compression in Network-on-Chip architectures [C]. In HPCA 2008.
- [204] Sanchez D, Michelogiannakis G, Kozyrakis C. An analysis of on-chip interconnection networks for large-scale chip multiprocessors [J]. *ACM Trans. Archit. Code Optim.* 2010, 7 (1): 4:1–4:28.
- [205] Michelogiannakis G, et al. Evaluating Bufferless Flow Control for On-chip Networks [C]. In NOCS 2010.
- [206] Fleury E, Fraigniaud P. A General Theory for Deadlock Avoidance in Wormhole-Routed Networks [J]. *IEEE Trans. Parallel Distrib. Syst.* 1998, 9: 626–638.
- [207] Lin X, McKinley P, Ni L. The message flow model for routing in wormhole-routed networks [J]. *Parallel and Distributed Systems, IEEE Transactions on.* 1995, 6 (7): 755 –760.

- [208] Schwiebert L, Jayasimha D N. A necessary and sufficient condition for deadlock-free wormhole routing [J]. *J. Parallel Distrib. Comput.* 1996, 32: 103–117.
- [209] Verbeek F, Schmaltz J. On Necessary and Sufficient Conditions for Deadlock-Free Routing in Wormhole Networks [J]. *Parallel and Distributed Systems, IEEE Transactions on*. 2011, 22 (12): 2022 –2032.
- [210] Mukherjee S, et al. The Alpha 21364 network architecture [C]. In *Hot Interconnects 2001*.
- [211] Verbeek F, Schmaltz J. A Comment on “A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks” [J]. *Parallel and Distributed Systems, IEEE Transactions on*. 2011, 22 (10): 1775 –1776.
- [212] Anjan K, Pinkston T. An efficient, fully adaptive deadlock recovery scheme: DISHA [C]. In *ISCA 1995*.
- [213] Duato J, Pinkston T. A general theory for deadlock-free adaptive routing using a mixed set of resources [J]. *Parallel and Distributed Systems, IEEE Transactions on*. 2001, 12 (12): 1219 –1235.
- [214] Vaidya A, Sivasubramaniam A, Das C. Impact of virtual channels and adaptive routing on application performance [J]. *Parallel and Distributed Systems, IEEE Transactions on*. 2001, 12 (2): 223 –237.
- [215] Jin Y, Yum K H, Kim E J. Adaptive data compression for high-performance low-power on-chip networks [C]. In *MICRO 2008*.
- [216] Tamir Y, Frazier G. High-performance multiqueue buffers for VLSI communication switches [C]. In *ISCA 1988*.
- [217] Carrion C, et al. A flow control mechanism to avoid message deadlock in k-ary n-cube networks [C]. In *HiPC 1997*.
- [218] Lenoski D, et al. The directory-based cache coherence protocol for the DASH multiprocessor [C]. In *ISCA 1990*.
- [219] Laudon J, Lenoski D. The SGI Origin: a ccNUMA highly scalable server [C]. In *ISCA 1997*.
- [220] Barroso L A, , et al. Piranha: a scalable architecture based on single-chip multiprocessing [C]. In *ISCA 2000*.
- [221] Conway P, Hughes B. The AMD Opteron Northbridge Architecture [J]. *Micro, IEEE*. 2007, 27 (2): 10 –21.

- [222] Puente V, et al. The adaptive bubble router [J]. *J. Parallel Distrib. Comput.* 2001, 61 (9): 1180–1208.
- [223] Chen L, Pinkston T. Personal communication. 2012.
- [224] Duato J, et al. A comparison of router architectures for virtual cut-through and wormhole switching in a NOW environment [J]. *J. Parallel Distrib. Comput.* 2001, 61 (2): 224–253.
- [225] Bhuyan L, et al. Approximate Analysis of Single and Multiple Ring Networks [J]. *IEEE Trans. Comput.* 1989, 38: 1027–1040.
- [226] Ainsworth T, Pinkston T. On Characterizing Performance of the Cell Broadband Engine Element Interconnect Bus [C]. In NOCS 2007.
- [227] Gottlieb A, et al. The NYU Ultracomputer - designing a MIMD, shared-memory parallel machine [C]. In ISCA 1982.
- [228] Leiserson C, et al. The Network Architecture of the Connection Machine CM-5 [C]. In *Journal of Parallel and Distributed Computing*. 1992: 272–285.
- [229] Cray Research Inc. CRAY T3D System Architecture Overview [C]. 1993.
- [230] R A N, et al. An Overview of the BlueGene/L Supercomputer [C]. In SC 2002.
- [231] Yang X, Liao X, Lu K, et al. The TianHe-1A Supercomputer: Its Hardware and Software [J]. *J. Comput. Sci. Technol.* 2011, 26 (3): 344–351.
- [232] Oh J, Prvulovic M, Zajic A. TLSync: support for multiple fast barriers using on-chip transmission lines [C]. In ISCA 2011.
- [233] Martin M, Hill M, Wood D. Token Coherence: decoupling performance and correctness [C]. In ISCA 2003.
- [234] Panda D. Fast barrier synchronization in wormhole k-ary n-cube networks with multideestination worms [C]. In HPCA 1995.
- [235] Xu H, McKinley P, Ni L. Efficient implementation of barrier synchronization in wormhole-routed hypercube multicomputers [C]. In ICDCS 1992.
- [236] Samman F, Hollstein T, Glesner M. New Theory for Deadlock-Free Multicast Routing in Wormhole-Switched Virtual-Channelless Networks-on-Chip [J]. *Parallel and Distributed Systems, IEEE Transactions on*. 2011, 22 (4): 544 –557.
- [237] Duato J, Yalamanchili S, Ni L. *Interconnection Networks: An Engineering Approach* [M]. 1st ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 1997.
- [238] Bolotin E, et al. The Power of Priority: NoC Based Distributed Cache Coherency [C]. In NOCS 2007.

- [239] Enright Jerger N D, Peh L-S, Lipasti M H. Virtual tree coherence: Leveraging regions and in-network multicast trees for scalable cache coherence [C]. In MICRO 2008.
- [240] Chiang C-M, Ni L M. Multi-address Encoding for Multicast [C]. In 1st International Workshop on Parallel Computer Routing and Communication 1994.
- [241] Muralimanohar N, Balasubramonian R, Jouppi N. CACTI 6.0: A tool to model large caches, HPL-2009-85 [R]. 2009.
- [242] Gonzalez R, Horowitz M. Energy dissipation in general purpose microprocessors [J]. Solid-State Circuits, IEEE Journal of. 1996, 31 (9): 1277 –1284.
- [243] Wang L, et al. Efficient lookahead routing and header compression for multicasting in networks-on-chip [C]. In ANCS 2010.
- [244] Gupta A, Weber W-D, Mowry T. Reducing Memory and Traffic Requirements for Scalable Directory-Based Cache Coherence Schemes [C]. In ICPP 1990.

作者在学期间取得的学术成果

发表的学术论文

- [1] **Sheng Ma**, Natalie Enright Jerger, Zhiying Wang, Mingche Lai, Libo Huang. Holistic Routing Algorithm Design to Support Workload Consolidation in NoCs. In *IEEE Transactions on Computers*, August 2012. (体系结构领域旗舰期刊, 已刊出, SCI 检索, DOI 号: 10.1109/TC.2012.201)
- [2] **Sheng Ma**, Natalie Enright Jerger, Zhiying Wang. Supporting Efficient Collective Communication in NoCs. In *International Symposium on High Performance Computer Architecture (HPCA)*, 2012. (体系结构领域顶级会议, EI 检索: 20121814986351)
- [3] **Sheng Ma**, Natalie Enright Jerger, Zhiying Wang. Whole Packet Forwarding: Efficient Design of Fully Adaptive Routing Algorithms for Networks-on-Chip. In *International Symposium on High Performance Computer Architecture (HPCA)*, 2012. (体系结构领域顶级会议, EI 检索: 20121814986375)
- [4] **Sheng Ma**, Natalie Enright Jerger, Zhiying Wang. DBAR: An Efficient Routing Algorithm to Support Multiple Concurrent Applications in Networks-on-Chip. In *International Symposium on Computer Architecture (ISCA)*, 2011. (体系结构领域顶级会议, EI 检索: 20113714321818)
- [5] **Sheng Ma**, Libo Huang, Zhiying Wang, Mingche Lai. The Design of Multiple-Precision Floating-Point Multiplier with SIMD Support. In *International Conference on Electronics, Circuits, and Systems (ICECS)*, 2009. (EI 检索: 20101712893859)
- [6] **Sheng Ma**, Libo Huang, Mingche Lai, Zhiying Wang. A Comparative Study of Subword Parallel Adders for Multimedia Applications. In *International Conference on ASIC (ASICON)*, 2009. (EI 检索: 20101112773194)
- [7] **Sheng Ma**, Libo Huang, Zhiying Wang, Kui Dai. Implementation of OpenVG Path and Paint Algorithms on SDTA with Optimization. In *International Conference on Networking, Architecture, and Storage (NAS)*, 2009. (EI 检索: 20094712474424)
- [8] 马胜, 黄立波, 王志英, 戴葵, 刘聪. 子字并行加法器的设计与实现. *计算机工程与应用*, 2009.

- [9] 马胜, 戴葵, 黄立波, 王志英. OpenVG 算法在 SDTA 结构上的优化实现. 计算机工程, 2009.
- [10] Libo Huang, **Sheng Ma**, Li Shen, Zhiying Wang, Nong Xiao. Low-Cost Binary128 Floating-Point FMA Unit Design with SIMD Support. In *IEEE Transactions on Computers*, 61(5): 745-751, 2012. (体系结构领域旗舰期刊, SCI 检索: 912LR)
- [11] Mingche Lai, Lei Gao, **Sheng Ma**, Nong Xiao, Zhiying Wang. A Practical Low-latency Router Architecture with Wing Channel for On-chip Network. In *Microprocessors and Microsystems - Embedded Hardware Design*, 35(2): 98-109, 2011. (权威期刊, SCI 检索: 739PI)
- [12] Libo Huang, Li Shen, Zhiying Wang, Wei Shi, Nong Xiao, **Sheng Ma**. SIF: Overcoming the Limitations of SIMD Devices via Implicit Permutation. In *International Symposium on High Performance Computer Architecture (HPCA)*, 2010. (体系结构领域顶级会议, EI 检索: 20102112951179)
- [13] Libo Huang, Li Shen, **Sheng Ma**, Nong Xiao, Zhiying Wang. DM-SIMD: A New SIMD Predication Mechanism for Exploiting Superword Level Parallelism. In *International Conference on ASIC (ASICON)*, 2009.

授权专利

- [1] 赖明澈, 高蕾, 王志英, 肖依, 陆弘毅, 马胜, 任珊珊. 一种基于快速通道技术的单周期片上路由器 (授权号: CN102394809A, 授权时间: 2012.03.28).
- [2] 赖明澈, 高蕾, 王志英, 陆弘毅, 肖依, 沈立, 任珊珊, 马胜. 一种多线程边界网关协议并行处理方法 (授权号: CN102185751A, 授权时间: 2011.09.14).