计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(4): 930−949, 2025

基三众核架构中基于同步哈密顿环的无死锁策略

李春峰[1] Karim Soliman[1] 计卫星[2] 石 峰[1]
[1](北京理工大学计算机学院 北京 100081)
[2](北京师范大学人工智能学院 北京 100875)
(lichunfeng@bit.edu.cn)

Deadlock-Free Strategies Based on Synchronized Hamiltonian Ring in Triplet-Based Many-Core Architecture

Li Chunfeng[1], Karim Soliman[1], Ji Weixing[2], and Shi Feng[1]
[1](*School of Computer Science & Technology*, *Beijing Institute of Technology*, *Beijing* 100081)
[2](*School of Artificial Intelligence*, *Beijing Normal University*, *Beijing* 100875)

Abstract Ensuring deadlock-free data transmission in the network-on-chip (NoC) is a prerequisite for providing reliable communication services for multi-processor system-on-chip (MPSoC), directly determining the availability of NoC and even MPSoC. Existing general-purpose deadlock-free strategies are oriented to arbitrary topologies, making it challenging to utilize the features and advantages of a specific topology. Moreover, these strategies may even increase network latency, power consumption, and hardware complexity. In addition, due to significant differences in the regular network between routing-level and protocol-level deadlocks, existing solutions struggle to simultaneously address both types of deadlock issues, affecting MPSoC reliability. We propose a deadlock-free strategy with synchronous Hamiltonian rings based on the inherent Hamiltonian characteristics of the triplet-based many-core architecture (TriBA). This method uses the topology's symmetric axes and Hamiltonian edge to allocate independent store-and-forward buffers for data transmission, preventing protocol-level deadlocks and improving data transfer speed. Additionally, we design a directional determination method for data transmission within the same buffer using cyclic linked-list technology. This method ensures data independence and synchronous forward transmission, eliminates routing-level deadlocks, and reduces data transfer latency. Based on optimizing redundant calculations in look-ahead routing algorithms, we propose a deadlock-free routing mechanism called Hamiltonian shortest path routing (HamSPR) based on a synchronous Hamiltonian ring. GEM5 simulation results show that, compared with existing solutions in the TriBA, HamSPR reduces average packet latency and power consumption in synthetic traffic patterns by 18.78%−65.40% and 6.94%−34.15%, respectively, while improving throughput by 8.00%−59.17%. In the PARSEC benchmark, HamSPR achieves maximum reductions of 16.51% in application runtime and 42.75% in average packet latency, respectively. Moreover, compared with 2D-Mesh, TriBA demonstrates an application performance improvement of 1%−10% in PARSEC benchmark.

Key words many-core processors; network-on-chip; triplet-based many-core architecture; Hamiltonian characteristics; routing algorithm; deadlock prevention

摘 要 确保片上网络(network-on-chip, NoC)中的数据传输无死锁,是NoC为多处理器片上系统(multi-

收稿日期: 2023-12-25; **修回日期**: 2024-08-12
基金项目: 国家自然科学基金项目(60973010)
This work was supported by the National Natural Science Foundation of China (60973010).
通信作者: 石峰(bitsf@bit.edu.cn)

processor system-on-chip, MPSoC）提供可靠通信服务的前提，决定了 NoC 甚至 MPSoC 的可用性. 现有的通用防死锁策略难以发挥出特定拓扑结构的自身特点和优势，甚至可能会增加网络延迟、功耗以及硬件复杂性. 另外，由于路由级和协议级死锁存在显著差异，现有无死锁方案较难同时解决这 2 类死锁问题，影响了 MPSoC 的可靠性. 利用基三众核架构（triplet-based many-core architecture, TriBA）中拓扑结构自身具有的哈密顿特性提出了基于同步哈密顿环的无死锁策略，该策略依据拓扑结构自身的对称轴和哈密顿边对数据传输进行分类，预防了协议级死锁并提高了数据传输速度；同时使用循环链表技术判断同一缓冲区内数据同步传输方向，消除了路由级死锁并降低了数据传输延迟. 在优化前瞻路由算法基础上，设计了基于同步哈密顿环的无死锁路由机制 HamSPR（Hamiltonian shortest path routing）. GEM5 仿真结果表明，与 TriBA 现有方法相比，HamSPR 在合成流量下的平均数据包延迟和功耗分别降低了 8.78%～65.40% 和 6.94%～34.15%，吞吐量提高了 8.00%～59.17%；在 PARSEC 测试集下的应用运行时间和平均数据包延迟分别最高实现了 16.51% 和 42.75% 的降低. 与 2D-Mesh 架构相比，TriBA 在 PARSEC 测试集下的应用性能实现了 1%～10% 的提升.

关键词 众核处理器；片上网络；基三众核架构；哈密顿特性；路由算法；死锁预防

中图法分类号 TP302

DOI: 10.7544/issn1000-1239.202331042 **CSTR:** 32373.14.issn1000-1239.202331042

随着科学计算、人工智能以及数据挖掘等领域的快速发展，并行处理大规模计算任务的场景需求日益增加，这对现代多处理器片上系统（multi-processor system-on-chip, MPSoC）中的通信架构提出了严峻挑战. 设计无死锁、高性能、可扩展和模块化的片上通信架构成为提升 MPSoC 性能的重要手段[1]. 基三众核架构（triplet-based many-core architecture, TriBA）是一种以 $WK_{d,h}$（d 代表度，h 代表网络的级数）递归结构[2]作为核间互连拓扑的新型片上网络架构[3]. 与当前主流 2D-Mesh 架构[4-5]相比，基于 $WK_{3,F}$（F 为层数）的 TriBA 具有更高的可扩展性[6]、独特的拓扑优势[3]和相似的 Tile 化布局布线方式[7]. 前期研究结果表明，TriBA 在通信负载、吞吐量、网络延迟以及功耗等方面与 2D-Mesh 相比具有明显优势[8-16]. 另外，TriBA 的片上内存网络相比于 2D-Mesh 架构具有更高的可扩展性[17]. 在相同的缓存命中率和网络规模下，其片上 Cache 加速比大约是 2D-Mesh 的 3 倍，并且该加速比随着网络规模的增加保持不变. 相比于扩展性有限的 2D-Mesh 架构[18]，TriBA 更具潜力应用在未来的 MPSoC 中.

由于片上网络（network-on-chip, NoC）中的物理资源有限，因此采用共享的方式以实现资源的最大化利用. 但当网络中存在循环资源依赖链而无法完成数据转发时，就会发生死锁. 随着网络的规模和复杂性逐渐增加，死锁发生的频率会越来越高[19]. 无死锁是 NoC 设计中最重要的问题，因为它是实现高效可靠的片上通信的前提，决定了 NoC 甚至 MPSoC 是否可用[19]. 最近相关工作表明[18-21]，无论是在 MPSoC 还是 Chiplet 中，死锁问题依然面临较为严峻的挑战. 通用的防死锁策略能够解决任意拓扑结构中出现的路由级或者协议级死锁，但难以发挥特定拓扑结构的特点和优势，甚至还会增加网络的延迟、功耗以及硬件复杂性，降低 MPSoC 的性能. 例如，针对任意循环网络的 Ring-Split 无死锁策略，通过采用非循环子网来绕过死锁，但这会带来较高的功耗以及硬件复杂性等问题[20]；针对任意不规则网络的死锁恢复（deadlock removal for arbitrary irregular networks, DRAIN）方法，由于会产生额外的控制存储开销，并不适合大规模的 MPSoC[21]；针对任意规则网络的基于链路权重的防死锁策略，当网络负载较重或某些链路频繁出现拥塞时，需重新优化链路权重以保持通信通畅，这不但会消耗额外的功耗，而且不正确的权重配置可能会导致死锁的发生[22]；针对任意规则或不规则网络的死锁检测恢复策略，需要实时检测并对陷入死锁的数据包及时地处理，这会导致 NoC 产生较高的硬件开销和逻辑复杂性[23]. 另外，由于路由级和协议级死锁发生原理存在显著差异，现有无死锁方案难以同时解决规则网络中的这 2 类死锁问题[19-21]，影响了 MPSoC 的可靠性.

哈密顿路径或者哈密顿环能够以线性或者循环的方式进行有效通信，这有益于提高网络的通信效率[24]. 另外，哈密顿性存在于大多数片上拓扑结构中（如 2D-Mesh, Torus），它是实现无死锁路由算法的基础之一[25]，也是设计无死锁策略最有效的方法之一[26].

因此, 本文针对现有无死锁方案无法较好地适用于 TriBA 架构而限制了架构的可用性和可靠性问题[11,27], 在对 TriBA 中的哈密顿特性等相关拓扑理论进行深入研究的基础上, 提出了基于同步哈密顿环的无死锁策略. 该策略是首次针对 TriBA 架构提出的能够同时解决路由级和协议级死锁, 并且在不产生额外开销(延迟、功耗)的情况下提升网络性能和可靠性的无死锁方法. 该方法对于其他具有哈密顿特性的众核架构(如中国科学院计算技术研究所提出的 Godson-T 众核架构[28])解决同类问题也具有重要的借鉴意义. 本文的主要贡献有 3 方面:

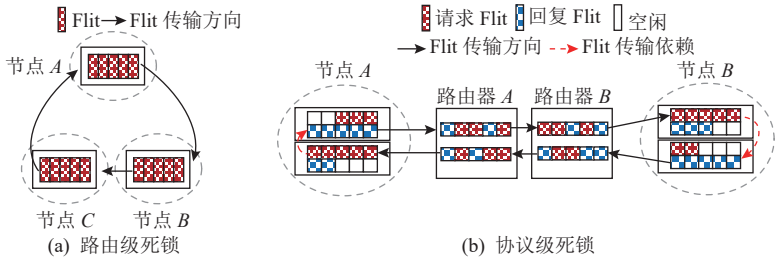
1) 设计了针对 TriBA 拓扑结构的对称轴和最大哈密顿边层次判断算法. 该算法能够在拓扑结构自身具有的对称轴基础上为不同类型的数据传输确定所经过的最大哈密顿边层次, 从而为其选择独立的存储转发缓冲区, 预防了不同数据传输之间的相互依赖和资源竞争, 消除了潜在的协议死锁并提高了数据传输速度.

2) 采用循环链表技术设计了同一哈密顿环内数据传输方向判断方法. 该方法能够将同一哈密顿环中的数据传输总体按照顺时针和逆时针方向进行划分, 结合全局时钟同步能够消除同一哈密顿环内不同方向的数据传输因竞争缓冲区而产生的路由级死锁, 降低了数据传输延迟.

3) 优化了现有的前瞻路由(look ahead shortest path routing, LA4T)算法中存在的冗余计算, 并在此基础上设计了基于同步哈密顿环的无死锁路由(Hamiltonian shortest path routing, HamSPR)机制. GEM5 仿真结果表明, 本文提出的方法优于 TriBA 现有的无死锁路由方案和采用 X-Y 和 West-first 无死锁路由的 2D-Mesh 架构, 能够进一步提高架构的通信性能和可用性.

1 背景及相关工作

路由级死锁[29]和协议级死锁[19]是当前 NoC 中主要存在的 2 类死锁问题. 当多个数据包保留在网络缓冲区并等待其他数据包占用的缓冲区变得可用时, 就会发生路由级死锁, 如图 1(a)所示. 通过对数据传输施加一些规则或者限制能够消除该类死锁问题, 如在 2D-Mesh 架构中, 通过设置转弯限制以避免网络中出现循环依赖的维序路由[22]; 采用基于不同链路权重的查表路由[30]; 限制非最小路由的 BoomGate 方法[21]等. 当不同消息类别(例如请求、响应)的数据包被迫相互等待时, 就会发生协议级死锁, 如图 1(b)所示. 通过采用虚拟网络、死锁恢复等方案能够解决这类死锁问题, 如在虚拟化虚拟队列(virtualizing virtual queues, VVQ)[19]方法中通过虚拟化虚拟通道以创建 VVQ 缓冲区架构从而避免发生协议死锁, 以及在 DRAIN[29]方法中通过定期排出(即强制移动)网络中可能涉及循环依赖的数据包以从潜在的死锁中恢复. 尽管协议级死锁发生的频率相比于路由级死锁相对较低[19], 但同时解决网络中潜在的路由级和协议级死锁, 对于提高网络的可靠性至关重要[30]. 由于路由级死锁和协议级死锁存在显著差异, 目前已有的死锁解决方案大多只针对规则网络中的其中一类死锁问题, 并且一些解决方案可能会带来额外的延迟、功耗和硬件复杂性, 显著影响了架构的性能和通用性[20-21,29-30]. 例如, 在一些基于 2D-Mesh 的多核处理器中, 用户可能需要执行特定于应用程序的优化[31-32]或是需要构建环形网络[33-34]以加速计算或消除死锁.



(a) 路由级死锁 (b) 协议级死锁

Fig. 1 Illustration of routing-level and protocol-level deadlocks generation in NoC

图 1 NoC 中路由级和协议级死锁产生示意图

表 1 汇总了 TriBA 现有的路由解决方案, 其中只有 Look-up Table, DM4T, LA4T 解决了潜在的路由级死锁或协议级死锁, 其余算法均未考虑潜在的死锁问题, 其主要原因有 2 点: 1)现有的通用防死锁方案一方面可能会产生额外的开销或带来较高的硬件复杂性, 影响架构的性能和可用性; 另一方面现有方案大都只能解决路由级死锁或协议级死锁, 影响了架构的可靠性. 2)TriBA 中存在众多环形路径, 并且每

Table 1 Routing Solutions Schemes Under TriBA Architecture

表 1 TriBA 架构下的路由解决方案

算法	对比方法	延迟（↓）	吞吐量（↑）	功耗（↓）	路由级死锁	协议级死锁
Look-up Table[11]	2D-Mesh	√	√	×	√	×
DDRA[35]					×	×
Min-DDRA[36]	DDRA			√	×	×
DM4T[27]	DDRA	√	√	×	√	×
SPORT[37]	DDRA 和 Min-DDRA	√			×	×
SPR4T[3]	SPORT		√	√	×	×
LA4T[38]	DDRA 和 SPORT	√	√	√	×	√
HamSPR（本文）	SPR4T 和 LA4T	√	√	√	√	√

注：↓表示降低；↑表示提升；√表示满足；×表示不满足.

一条路径在路由过程中都可能成为关键路径，不同环形路径之间的相互交叉以及关键路径的多变导致难以开发出高效的无死锁策略. 因此，研究能够同时解决路由级死锁和协议级死锁，又不影响架构的性能和可靠性的无死锁策略，成为提高 TriBA 架构的通信性能和可用性的关键. 石峰等人[39]提出了一种基于尖端子图分类传输数据的防死锁方法，但该方法仅针对架构中出现的协议级死锁，并且未探讨相关路由算法及其性能. 该方法是针对 TriBA 拓扑特性而设计的，也是本文主要对比的方法之一.

如图 2 所示，本文根据基三拓扑结构中具有的哈密顿特性提出了一种基于同步哈密顿环的无死锁策略，该策略同时采用了分类数据传输（如 1 级哈密顿环、2 级哈密顿环）和同步数据传输（路由器全局时钟同步）的方案，并结合同类数据传输方向划分和局部类气泡（空槽）流控方法，能够同时消除网络中潜在的路由级死锁和协议级死锁，提升架构的通信性能和可靠性. 由于该策略是根据拓扑结构自身具有的哈密顿特性而设计的，因此不影响架构的通用性.

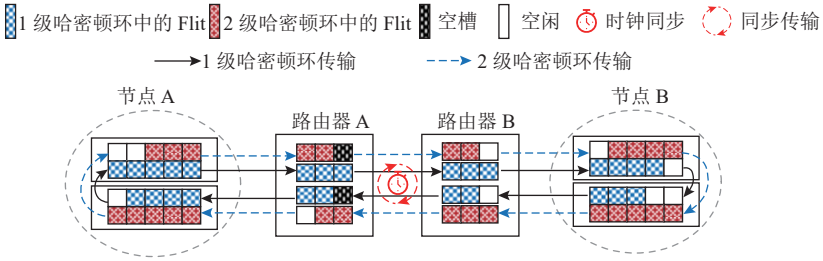


Fig. 2 Illustration of deadlock-free strategy based on synchronized Hamiltonian rings

图 2 基于同步哈密顿环的无死锁策略示意图

2 基三众核架构

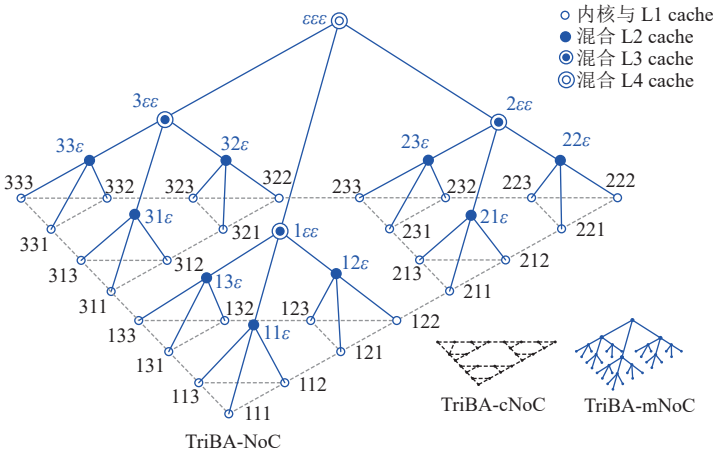
TriBA 的拓扑结构是递归拓扑 $WK_{d,h}$[2] 的一个特例 $WK_{3,F}$（度为 3，层数为 F）. 参照图论的命名风格，将 TriBA 拓扑命名为 TC^F（F 为层数），它可以通过递归的方式构成[3]. TriBA 的片上存储网络是一种三叉树网络，其将所有分布式 cache 模块连接到一起形成一个多级片上 cache 结构（层次化分组内共享）. 通过将 TriBA 与其内存网络融合在一起能够形成一种新的众核架构 TriBA-NoC，如图 3 所示，其中 TriBA 作为通信子网络 TriBA-cNoC（TriBA communication NoC），三叉树作为内存子网络 TriBA-mNoC（TriBA memory NoC）. 另外，虽然 TriBA-NoC 具有 3D 风格的特点，但却能够以瓦片（Tile）的方式集成在 2D 的硅平面上，如图 4 所示，并采用与 Chiplet 相同的设计理念，在实现晶圆面积的最佳使用效率的同时能够实现相同单元模块的重用[7]. 由于本文主要研究对象为 TriBA，因此对于 TriBA-mNoC 不做过多探讨.

2.1 拓扑结构的定义

使用 TC^F 表示 TriBA 的拓扑结构，结合图论方法对 TC^F 进行以下定义：

定义 1. 图 $TC^F=\{V(TC^F),E(TC^F)\}$，其中顶点集 $V(TC^F)=\{x_F\cdots x_2x_1|x_f\in\mathbb{T}\}$，边集 $E(TC^F)=\{\bar{x}_{F\cdots f+1}\cdot a\,b^{f-1}\leftrightarrow\bar{x}_{F\cdots f+1}b\,a^{f-1}\,|\,a,b\in\mathbb{T};\ a<b;f\in\mathbb{N};\ f=1,2,\cdots,F;\ \bar{x}_{F\cdots f+1}\in\mathbb{T}^{F-f}\}$.

说明：$\mathbb{T}=\{1,2,3\}$ 为 TriBA 的三文字集，用于命名 TC^F 的顶点，其二进制编码形式为 $\mathbb{T}=\{01,10,11\}$（下同）. 由若干 $\mathbb{T}$ 中文字符组成的序列称为文字串，$x_F\cdots x_2x_1$ 是长度为 F 的文字串，其中文字符 $x_i\in\mathbb{T}$，$1\leqslant i\leqslant F$，$x'_Fx'_F\cdots x'_1x'_1$ 为其二进制表示形式，x'_F 表示二进制数第 2 位，x'_F 表示二进制数第 1 位，如 111 的二进制表示为 010101. x^f 表示 $\mathbb{T}$ 中 f 个相同文字 x 构成的文字串，用 $\bar{x}$ 表示内容和长度已知文字串，如 $\bar{x}_{m\cdots n}$ 表



注：ε 表示占位符，可由不同的数字取代.

Fig. 3 Illustration of TriBA-NoC topology structure

图 3 TriBA-NoC 的拓扑结构示意图

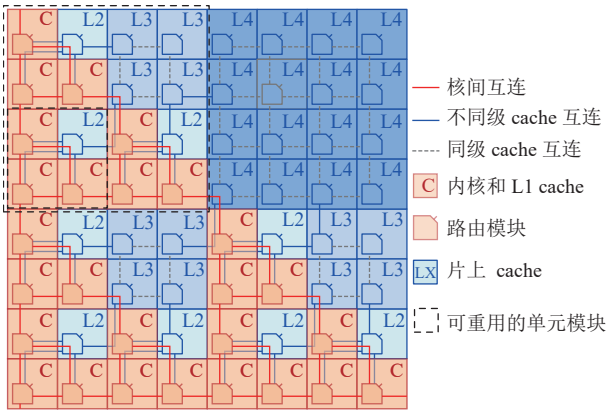


Fig. 4 Illustration of layout and wiring for TriBA-NoC[7]

图 4 TriBA-NoC 的布局布线示意图[7]

示 $x_m\cdots x_n$. $\mathbb{T}^F$ 为三文字集的笛卡儿幂集，表示为 $\{x_F\cdots x_2x_1|x_i\in\mathbb{T},i=1,2,\cdots,F\}$，即所有可能的长度为 F 的文字串的集合. b^{f-1} 表示为 $f-1$ 个 b 的文字串 ($b\in\mathbb{T}$). $x_F\cdots x_2x_1\leftrightarrow y_F\cdots y_2y_1$ 形式表示顶点 $x_F\cdots x_2x_1$ 和 $y_F\cdots y_2y_1$ 之间的双向边. $\mathbb{N}$ 是自然数集. F 是给定的自然数.

在 TriBA-NoC 中，内核之间通过 TriBA-cNoC 子网进行互连，片上 cache 之间通过 TriBA-mNoC 子网互连，内核与 cache 间则通过 TriBA-mNoC 实现互连. 例如在图 3 中，节点(或内核)111, 112, 113 与节点 11ε(L2 cache)相连，节点(L2 cache)11ε，(L2 cache)12ε，(L2 cache)13ε与节点 1$\varepsilon\varepsilon$(L3 cache)相连，剩余不同节点间的连接关系以此类推. TriBA-NoC 布局布线示意图如图 4 所示.

2.2 TC^F 路由子图

2.2.1 焦点路由图 FRG^f

当进行核间数据传输时，跨越整个 TC^F 图的情况并不多，更多的传输是在它的子图中进行. 例如在图 5(a)中，当需要计算顶点 $\bar{s}$ 到顶点 $\bar{t}$ 的最短路径(即距离)时，相关计算只需要聚焦到 TC^3 子图中即可，而不需要在整个 TC^F 上进行计算. 因此，将路由计算发生在同时覆盖路由起点和路由终点的最小子图称为焦点路由图 FRG^f(focus routing graph, FRG)，其中 f($f<F$)表示该图的规模(层数)，有时 f 也可能被一个超节点名所取代.

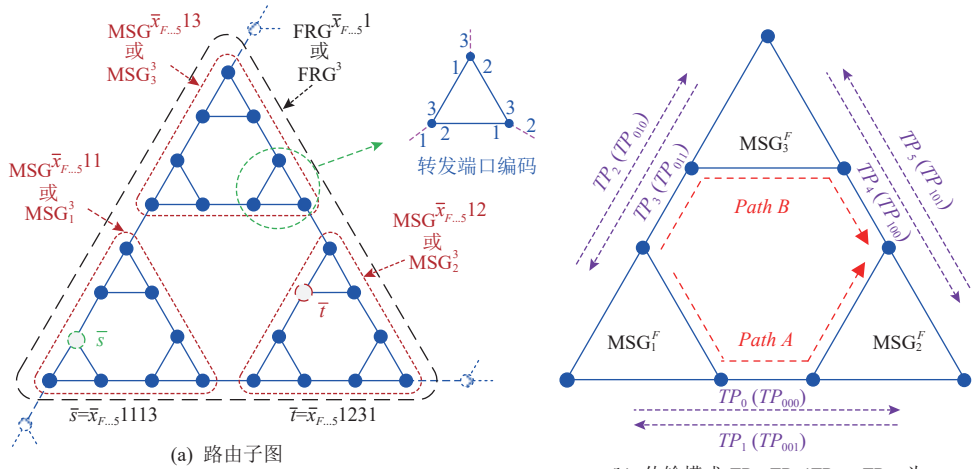
2.2.2 最大焦点路由子图 MSG_m^f

MSG_m^f(maximum subgraph of FRG, MSG)在 FRG^f 的基础上进行定义，旨在路由计算过程中能够准确定位路由起点和路由终点的具体位置，并准确给出最短路径，如图 5(a)中的 MSG_1^3, MSG_3^3, MSG_2^3.MSG_m^f 中的 f 与 FRG^f 相同，m 表示 FRG^f 的尖端顶点 v^f 所在的焦点路由子图，其中 $m,v\in\mathbb{T}$，v 表示尖端节点名，如图 3 中 TC^3 的 111(1^3)，222(2^3)，333(3^3)节点. 注意：MSG_m^f 是 TC^{f-1} 类子图，FRG^f 是 TC^f 类子图，MSG_m^f 中 f 可能像 FRG^f 那样被超节点名取代. 另外，在 MSG_m^f 的基础上针对 TC^F 拓扑特性定义了数据传输的不同模式和路径，如图 5(b)所示.

2.3 哈密顿特性

在图论中，哈密顿路径是指在无向图中，经过每个顶点 1 次且只有 1 次的路径. 若是在哈密顿路径中起点和终点是同一个节点，则这条哈密顿路径就构成了一个哈密顿环[40]. TC^F 的哈密顿性是指其拥有哈密顿路径(Hamiltonian path)和哈密顿环(Hamiltonian cycle)，下文简称H-路径和H-环.

定义 2. 在 TC^F 中的任意两尖端节点间存在一条路径，若它的起始节点为其中一个尖端节点，终点为

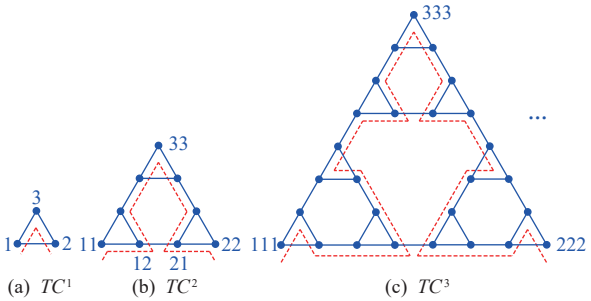


(a) 路由子图　(b) 传输模式 TP_0~TP_5 (TP_{000}~TP_{101} 为二进制表示) 及路径 *Path A* 和 *Path B*

Fig. 5　Illustration of the different routing subgraphs in TC^F, as well as the transmission modes and paths

图 5　TC^F 中不同的路由子图及传输模式和路径示意图

另一个尖端节点，并且经过TC^F中的每个顶点 1 次且仅 1 次，则称这条路径为H-路径，如图 6 所示.



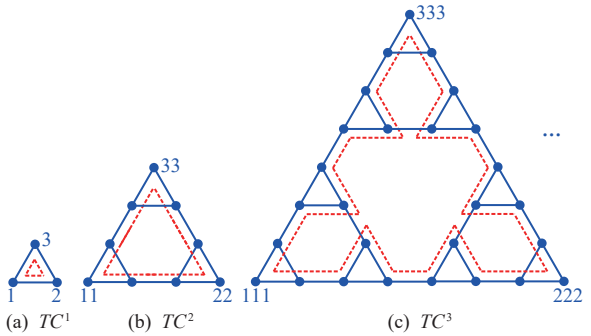
(a) TC^1　(b) TC^2　(c) TC^3

Fig. 6　H-path between apex 1^F and 2^F (taking TC^3 for example)

图 6　TC^F 尖端1^F和2^F之间的 H-路径（以TC^3为例）

定义 3. 在TC^F中，若 H-路径的起点和终点为同一个顶点，并且该路径穿越每个顶点 1 次且仅 1 次，则将这个闭合 H-路径称为 H-环，如图 7 所示.

TC^F的哈密顿性有益于解决架构中的流式计算或者防死锁问题，并且TC^F的每个子图都具有哈密顿环这一特征，因此该特性不受网络规模的影响.



(a) TC^1　(b) TC^2　(c) TC^3

Fig. 7　H-cycle between apex of TC^F (taking TC^3 for example)

图 7　TC^F 尖端之间的哈密顿环（以TC^3为例）

3　基于同步哈密顿环的无死锁策略原理

本节根据TC^F自身具有的哈密顿特性，首先介绍TC^F中哈密顿最大边及其环流的构造方法；其次阐述TC^F中的对称点及对称轴判断方法，并设计相应的判断算法；再次介绍根据对称轴为TC^F中的数据传输选择不同层次的哈密顿边并划分不同类型的缓冲区方法，以及设计相应的算法；最后，介绍使用链表技术划分同一哈密顿环内的数据传输方向判断方法.

3.1　哈密顿环流的构造

在TC^F哈密顿环的基础上，以圆环的风格将TC^F的哈密顿环路重新绘制，如图 8 所示，并对环路进一步的分解. 从图 8 中的哈密顿环可以发现，在运行时刻虽然TC^F的边上随机遍布着沿边传输的各个方向的数据，但在宏观上呈现了一种沿着哈密顿环的流动，将其称为虚拟哈密顿环流(virtual Hamiltonian ring, H-ring)，简称环流. 为了进一步理解哈密顿环流，定义了 TC^F中的哈密顿边和非哈密顿边.

定义 4. 按照图 8 中哈密顿圆环的画法，规定在圆环中哈密顿边位于圆环的最外侧，内侧向心方向依次出现链接 2 个顶点、链接 2 个TC^1、链接 2 个TC^2等的非哈密顿边，这些边的层次依次称为 2, 3, 4, …，哈密顿边是第 1 层边.

如图 8 所示，那些非哈密顿边链接的节点为哈密顿环上的“捷径”(如图 8 中的虚线和点划线). 在最短路径路由机制的约束下(数据传输始终选择最短的路径)，数据在哈密顿环和捷径间进行跳跃. 如果

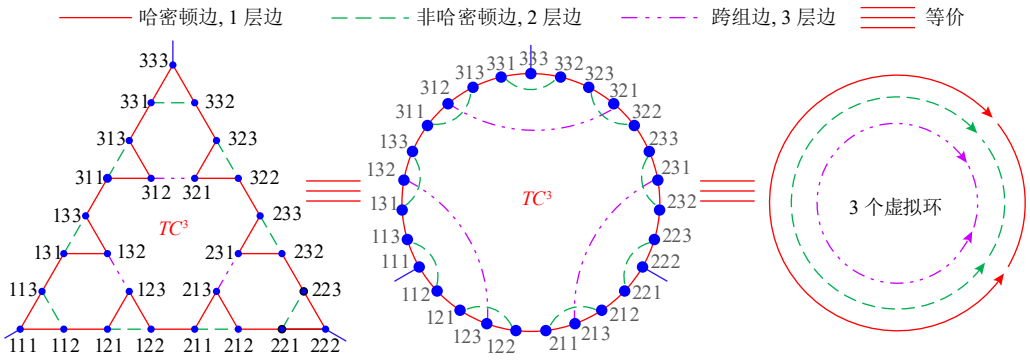


Fig. 8 Virtual Hamiltonian ring flow in TC^3

图 8 TC^3中的虚拟哈密顿环流

认为“捷径”上的数据在哈密顿路径中的传输使用了自己的独立通道, 那么在TC^F上的数据传输“逻辑上”形成了沿着F个独立逻辑通道的传输. 如在图 8 中将TC^3逻辑上划分为 3 个环, 每个环的级别取决于最大哈密顿边的层次.

接下来, 将进一步探讨如何确定数据传输所经过的最大边层次, 从而正确地为数据传输分配独立的缓冲区.

3.2 对称轴/对称点判别

要想使通信数据“逻辑上”在自己独立的通道中传输, 技术上必然分时复用相同的物理通道, 即通过独享各自数据类型对应的存储转发缓冲区来实现. 而对通信数据所使用缓冲区类型的判断, 需在路由计算的同时, 根据对称轴判断哈密顿路径结构和走向情况. 为此给出下述对称轴相关的定义、定理和算法.

定义 5. 在子图$TC^f \subseteq TC^F$平面图示中, 2 尖端p^f和q^f间的哈密顿路径(TC^F的哈密顿路径的一部分)中点(或称对称点)为尖端k^f, 该顶点或其所在对称轴记为$\bar{x}_{F\cdots f+1}A_k^f$, 如图 9 所示, 其中$\bar{x}_{F\cdots f+1}$是$TC^f$名(长度为$F-f$的文字串), $p,q,k \in \mathbb{T}$且$p \neq q \neq k$.

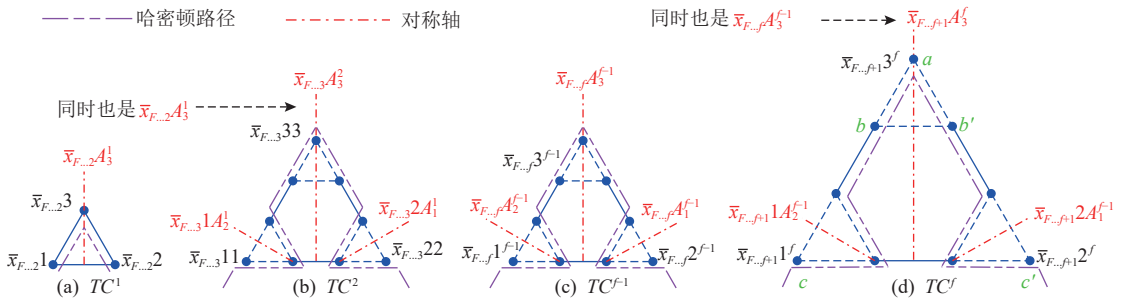


Fig. 9 H-path betwee 1^f and 2^f in apex TC^f and its associated symmetry axis

图 9 TC^f尖端1^f和2^f之间的H-路径及其相关对称轴

定理 1. 任意一个$TC^f \subseteq TC^F$尖端节点都是另外 2 个尖端节点间哈密顿路径的中点, 即该尖端沿该哈密顿路径到另外 2 个尖端节点的距离相同.

证明. 对TC^f规模f应用数学归纳法, 以尖端3^f到尖端1^f和2^f的哈密顿距离为例证明定理 1, 其他情况可类似证明.

1)基础步骤: $f=1$时, 即TC^1尖端节点 3 到另外 2 尖端 1 和 2 的哈密顿距离均为 1.

2)归纳步骤: 当$f>1$时, TC^f由 3 个TC^{f-1}及 3 个互连单边构成, 如图 9 所示. 根据归纳步骤, 尖端3^{f-1}到所在TC^{f-1}另外 2 个尖端的哈密顿距离相等, 即 2 个距离$\overline{ab}$和$\overline{ab'}$相等. 而根据TC^F中任何 2 个尖端顶点间 H-路径的长度为3^F-1, 距离$\overline{bc}$和$\overline{b'c'}$均为3^{f-1}, 于是定理 1 得证. 证毕.

定理 2. 对称轴为$\bar{x}_{F\cdots f+1}A_k^f$的图$TC^f \subseteq TC^F$由对称轴分别为$\bar{x}_{F\cdots f+1}pA_q^{f-1}$, $\bar{x}_{F\cdots f+1}kA_k^{f-1}$, $\bar{x}_{F\cdots f+1}qA_p^{f-1}$的$TC^{f-1}$子图及互连单边构成, 其中$\bar{x}_{F\cdots f+1}$是$TC^f$名, $p,q,k \in \mathbb{T}$且$p \neq q \neq k$.

证明. 对TC^f的f应用数学归纳法, 这里以它的对称轴是$\bar{x}_{F\cdots f+1}A_3^f$为例证明, 其他情况可类似证明.

1)基础步骤: $f=1$时(如图 9 中的TC^1), 如果 3 顶点的TC^1的名字为$\bar{x}_{F-1}$, 则它的对称轴为$\bar{x}_{F-1}A_3^1$, 即对称点$\bar{x}_{F-1}3$, 至于定理 2 中的子图则为单个顶点. 而当$f=2$时(图 9 中的TC^2), 如它的对称轴是$\bar{x}_{F-3}A_3^2$,

此时，其 3 个 TC^1 子图的对称轴按定理 2 所述分别为 $\bar{x}_{F\cdots3}1A_2^1$，$\bar{x}_{F\cdots3}3A_3^1$，$\bar{x}_{F\cdots3}2A_1^1$，对称点分别为 $\bar{x}_{F\cdots3}12$，$\bar{x}_{F\cdots3}33$，$\bar{x}_{F\cdots3}21$，与定理 1 所述情况相同.

2)归纳步骤：$f>2$ 时，TC^f 由 3 个 TC^{f-1} 及 3 个互连单边构成，如果 TC^f 的对称点为 $\bar{x}_{F\cdots f+1}A_3^f$，显然考虑到 3 个 TC^{f-1} 的哈密顿路径片段的端点情况，位于 TC^f 左下角的 TC^{f-1} 的对称点必然是 $\bar{x}_{F\cdots f+1}12^{f-1}$，对称轴为 $\bar{x}_{F\cdots f+1}1A_2^{f-1}$；位于正上方的 TC^{f-1} 的对称点是 $\bar{x}_{F\cdots f+1}33^{f-1}$，对称轴为 $\bar{x}_{F\cdots f+1}3A_3^{f-1}$；位于右下方的 TC^{f-1} 的对称点必然是 $\bar{x}_{F\cdots f+1}21^{f-1}$，对称轴为 $\bar{x}_{F\cdots f+1}2A_1^{f-1}$；故定理 2 得证. 证毕.

定理 2 中图 TC^f 的对称轴与其构成子图对称轴的关系是，如果图的对称轴是 $\bar{x}_{F\cdots f+1}A_k^f$，则层码 $x_f=k$ 构成子图的对称轴 $\bar{x}_{F\cdots f+1}kA_k^{f-1}$ 与此轴重合；层码 $x_f=p$ 构成的子图对称轴 $\bar{x}_{F\cdots f+1}A_q^{f-1}$ 与 $\bar{x}_{F\cdots f+1}A_q^f$ 重合；层码 $x_f=q$ 构成的子图对称轴 $\bar{x}_{F\cdots f+1}jA_p^f$ 与 $\bar{x}_{F\cdots f+1}A_p^f$ 重合；其 $p,q,k\in\mathbb{T}$ 且 $p\neq q\neq k$，如图 9 所示. 例如，$213A_2^4$ 是名称为 213 的 TC^4 子图，其对称轴/顶点是该 TC^4 的尖端 2^4 或顶点 2132222. 再例如，TC^4 对称轴 $\bar{x}_{F\cdots5}A_1^4$ 可以展开一层后，下一个层次 3 个 TC^3 子图的对称点分别为 $\bar{x}_{F\cdots5}2A_3^3$，$\bar{x}_{F\cdots5}1A_1^3$，$\bar{x}_{F\cdots5}3A_2^3$；每个对称轴所在子图均可根据定理 2 继续深入展开，例如 $\bar{x}_{F\cdots5}3A_2^3$ 所在 TC^3 子图继续展开后的 TC^2 子图对称轴分别为 $\bar{x}_{F\cdots5}31A_3^2$，$\bar{x}_{F\cdots5}32A_2^2$，$\bar{x}_{F\cdots5}33A_1^2$.

根据定理 1 和定理 2，设计了 FRG^f 的对称轴判断算法 DetAxiSym(determine the axis of symmetry)，具体如下：

算法 1. FRG^f 的对称轴判断算法 DetAxiSym.

输入：源节点 $s_F\cdots s_2s_1$，目标节点 $t_F\cdots t_2t_1$，当前通信模式 TP；

输出：FRG^f 的对称轴 $\bar{x}_{F\cdots f+1}A_x^f$(变量 *Axis* 标识).

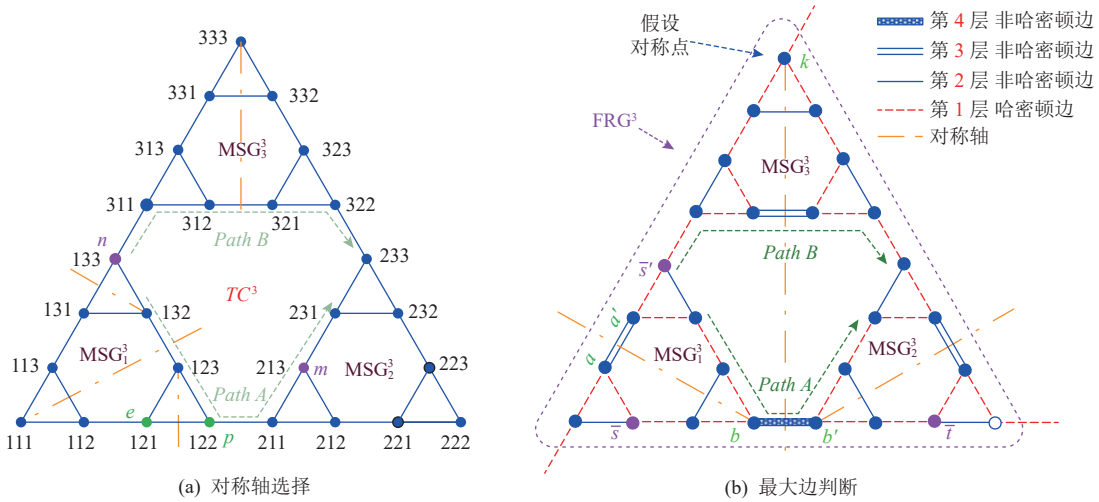
① begin
② if ($s_F\neq t_F$) then
③ if(TP_0或TP_1)then
④ $Axis\leftarrow 3A_3^{F-1}$;
⑤ else if (TP_2或TP_3)then
⑥ $Axis\leftarrow 2A_2^{F-1}$;
⑦ else $Axis\leftarrow 1A_1^{F-1}$; /*此时通信模式为$TP_4$或$TP_5$*/
⑧ end if
⑨ return *Axis*;
⑩ end if
⑪ $Axis\leftarrow s_FA_{s_F}^{F-1}$; /*$\mathrm{FRG}^f\subset\mathrm{FRG}^F$ 时的初始对称轴*/
⑫ for($i=F-1$; $i\geqslant f+1$; $i--$)do
⑬ if($s_i\neq s_{i-1}$)then
⑭ $s\leftarrow\mathbb{T}(s_i,s_{i-1})$; /*选择$\mathbb{T}$中不等于$s_i$和$s_{i-1}$的字符*/
⑮ $Axis\leftarrow\bar{x}_{F\cdots i+1}\times s_iA_s^{i-1}$; /*原 *Axis* 的值为$\bar{x}_{F\cdots i+1}A_x^i$*/
⑯ else $Axis\leftarrow\bar{x}_{F\cdots i+1}\times s_{i-1}A_{s_{i-1}}^{i-1}$;
⑰ end if
⑱ end for
⑲ return *Axis*;
⑳ end

以 TC^3 为例，详细介绍节点间数据传输的对称轴择选过程. 如图 10(a)所示，若节点 n(133)与节点 m(213)进行通信，首先根据节点 n 和 m 的文字串的第 1 位不同($s_3\neq t_3$)判断通信所在的 FRG 为 3($f=F=3$)，接着在最短路径路由的约束下可判断此时的通信类型为 TP_0，最后根据算法 1(行④)可判断对称轴为 $3A_3^2$(即 333 节点所在对称轴)；若节点 e(121)与节点 p(122)通信，首先根据节点 e 和 p 文字串的第 1 和第 2 位相同($s_3=t_3$，$s_2=t_2$)而第 3 位不同(即 $s_3\neq t_3$)，判断通信所在的 FRG 为 1(即 $f=1$)，接着设置初始对称轴为 $1A_1^2$(111 节点所在对称轴，算法 1 行⑪)，然后根据节点 e 中的字符 s_2 和 s_1 不相等(算法 1 行⑬)，从 $\mathbb{T}=\{1,2,3\}$ 中选择不等于 s_2 和 s_1 的字符 3(算法 1 行⑭)，最后修改初始对称轴为 $12A_3^1$(123 节点所在对称轴，算法 1 行⑮). 其余情况与上述相同就不在此赘述. 根据不同对称轴所对应非哈密顿边的特点不同，定义了不同层次的边，如图 10(b)所示.

3.3 数据通信缓冲区类型判别

路由核心计算在等价 TP_0 传输模式(MSG_1^F 内的节点与 MSG_2^F 中的节点通信记为 TP_0)中进行，如图 5(b)所示，但随 FRG 的位置不同，它的哈密顿环形状也会随之发生变化. 所以路由核心计算在得到转发端口后，还必须确定该数据将使用的缓冲区类型，即确定相应的传输属于 TP_0 样式下的哪个哈密顿环. 其中，缓冲区类型按照哈密顿环流所经过的最大哈密顿边或非哈密顿边层次编号，如图 8 所示. 第 1 层哈密顿环对应哈密顿边，缓冲区为 TYPE-I 型；第 2 ~ F 层环流对应第 2 ~ F 层非哈密顿边，相应的缓冲区类型是 TYPE-II~TYPE-F. 接下来以 FRG^3 为例，详细介绍缓冲区判断原理及过程.

如图 10(b)所示，若节点 $\bar{s}$ 与 $\bar{t}$ 通信，首先判断此



(a) 对称轴选择 (b) 最大边判断

Fig. 10 Example of symmetry axis selection and maximum edge determination in a deadlock-free strategy based on synchronous Hamiltonian rings

图 10 基于同步哈密顿环的无死锁策略中对称轴选择和最大边判断示例

时通信所在的最大FRG为 3, 在最短路径路由的约束下, 数据传输路径选择 *Path A*, 根据算法 1 获得相应的对称点 k 及其该点所在的对称轴; 其次判断节点 $\bar{s}$ 与 $\bar{t}$ 所在的 MSG 分别为 MSG_1^3 和 MSG_2^3, 此时 MSG_1^3 内的节点与 MSG_2^3 内的节点通过路径 *Path A* 通信, 则必然经过对称轴所对应的最大层次边 $b \leftrightarrow b'$; 最后确定节点 $\bar{s}$ 与 $\bar{t}$ 通信经过的最大哈密顿边层次为 4, 并将其分配到 4-Level H-ring 缓冲区中进行数据传输. 节点 $\bar{s}$ 与 $\bar{s}'$ 通信缓冲区判断类似, 其所经历的最大哈密顿边层次为 3. 综上所述, 在 TP_0 传输模式下, 对称轴分别为 $\bar{x}_{F\cdots f+1}A_1^f$, $\bar{x}_{F\cdots f+1}A_2^f$, $\bar{x}_{F\cdots f+1}A_3^f$ 下的 *Path A* 和 *Path B* 通信数据缓冲区类型判断算法 GetBufType(get buffer type)如下:

算法 2. 缓冲区类型判断算法 GetBufType.

输入: 焦点路由图 FRG^f, 当前传输路径 *Path*, 对称轴 *Axis*, 源节点 $\bar{x}_{F\cdots f+1}s_f\cdots s_2s_1$, 目标节点 $\bar{x}_{F\cdots f+1}t_f\cdots t_2t_1$;

输出: 当前通信经历的最高边层次 *MaxLevelNum*.

① begin
② if(*Path* == *Path B*)then
③ if($f == F$)then /*f和F来自FRG^f和源节点*/
④ return *MaxLevelNum* ← F;
⑤ else if(*Axis* == $\bar{x}_{F\cdots f+1}A_3^f$)then
⑥ return *MaxLevelNum* ← f;
⑦ else
⑧ return *MaxLevelNum* ← f+1; /*Axis 为 $\bar{x}_{F\cdots f+1}A_1^f$或$\bar{x}_{F\cdots f+1}A_2^f$*/
⑨ end if
⑩ end if
⑪ if(*Axis* == $\bar{x}_{F\cdots f+1}A_1^f$)then /**Path* == *Path A*/
⑫ for(i=1; $i \leqslant f-1$; i++)do
⑬ if(i mod 2 ≠ 0&&(s_{f-i} == 3||t_{f-i} == 2))then
⑭ *MaxLevelNum* ← $f-i+1$, break;
⑮ end if
⑯ if(i mod 2 ≠ 0&&(s_{f-i} == 1||t_{f-i} == 3))then
⑰ *MaxLevelNum* ← $f-i$, break;
⑱ end if
⑲ if(i mod 2 == 0&&(s_{f-i} == 1||t_{f-i} == 3))then
⑳ *MaxLevelNum* ← $f-i$+1, break;
㉑ end if
㉒ if(i mod 2 == 0&&(s_{f-i} == 3||t_{f-i} == 2))then
㉓ *MaxLevelNum* ← $f-i$, break;
㉔ end if
㉕ end for
㉖ return *MaxLevelNum*;
㉗ end if
㉘ if(*Axis* == $\bar{x}_{F\cdots f+1}A_2^f$)then
㉙ for (i=1; $i \leqslant f-1$; i++) do
㉚ if(i mod 2 ≠ 0&&(s_{f-i} == 1||t_{f-i} == 3))then
㉛ *MaxLevelNum* ← $f-i$+1, break;
㉜ end if
㉝ if(i mod 2 ≠ 0&&(s_{f-i} == 3||t_{f-i} == 2))then
㉞ *MaxLevelNum* ← $f-i$, break;
㉟ end if
㊱ if(i mod 2 == 0&&(s_{f-i} == 3||t_{f-i} ==

2))then
㊲ *MaxLevelNum* ← $f-i+1$, break;
㊳ end if
㊴ if($i \bmod 2 == 0 \&\& (s_{f-i} == 1 \| t_{f-i} == 3)$) then
㊵ *MaxLevelNum* ← $f-i$, break;
㊶ end if
㊷ end for
㊸ return *MaxLevelNum*;
㊹ end if
㊺ return *MaxLevelNum*← $f+1$; /**Axis* 为 $\bar{x}_{F\cdots f+1}A_3^f$ */
㊻ end

在算法 2 中，首先根据不同的传输路径（*Path A* 或 *Path B*）划分数据传输可能经过的最大哈密顿边层次（行②⑪）. 当路由算法选择 *Path B* 时，由于FRG^F的哈密顿环与其内的FRG^f哈密顿路径不同，因此需根据所在的焦点路由图分别进行数据类型判断（行③～⑩）；当路由算法选择 *Path A* 时，数据传输经过的最高层次边与FRG^f的 f 值、源节点在FRG^f中的位置和源节点所在FRG^f的子图对称轴均有关，需根据 f 值和 F 值进一步判断（行⑪～㊹）. 另外，根据 TriBA 的旋转对称性，可以将其他通信模式下的缓冲区类型计算通过置换群[3]$\mathbb{S}_3$变换至TP_0传输模式下进行，并通过$\mathbb{S}_3^{-1}$逆变换返回输出结果即可.

3.4 哈密顿环内数据传输方向划分

通过对数据传输划分独立的缓冲区类型，可以使TC^F中不同的数据通信限制在自己独立的通道中而不相互干扰. 但如果一个独立的传输通道中既存在沿哈密顿环的顺时针传输又存在逆时针传输（如图 8 所示），当 2 个不同方向的数据传输相遇时，就可能会出现阻塞和资源竞争，从而导致长时间等待、高负载时甚至可能引发死锁. 因此，为了避免出现上述问题，通过采用循环链表将TC^F上的同一缓冲区中的数据传输在通信发起起点时确定消息所属的传输方向，从而将其划分到哈密顿顺时针传输通道或哈密顿逆时针传输通道上，并且后续的传输始终在同一通道中进行，避免在同一哈密顿环的传输通道中因出现 2 个不同方向的传输而可能产生的资源竞争问题.

本文以TC^3为例（如图 8 所示）阐述划分方法：

1）建立循环链表. 以 111 节点为链表的头节点，按照顺时针方向依次链接环上的所有节点，尾节点为 112.

2）循环动态查询. 假设源节点为 A，目标节点为 B，当通信建立时，在通信起点的链表上以从前往后的方向（顺时针）和从后往前的方向（逆时针）分别查找节点 A，查到后标记为新的节点 A'和 A''，以 A'和 A''为新起点分别从顺时针方向和逆时针方向计算距离节点 B 的距离，记为 *Dis_A′* 和 *Dis_A″*

3）划分传输方向. 比较 *Dis_A′* 和 *Dis_A″* 的距离，若 $Dis_A' \leqslant Dis_A''$ 则该数据传输方向为顺时针方向（标记为 1），否则为逆时针方向（标记为 0）.

由于数据传输方向的划分只需要在通信发起时确定，因此只需要查询 1 次即可，该方法不同于基于查表的路由需要多次查询以确定每一次的转发端口.

4 基于同步哈密顿环的无死锁路由机制

4.1 LA4T 路由算法及其改进

LA4T 路由算法是一种前瞻式路由算法，该算法能够根据当前节点转发端口、尖端节点名以及所在的焦点路由图FRG^f，推断下一节点的转发端口，从而提高数据包转发效率[30,38]. 但该算法没有考虑通信数据自身所隐含的信息（例如数据包跳数），导致仍然需要在每次转发时计算 *Path A* 和 *Path B* 路径长度并进行对比，这相对来说比较耗时. 在 LA4T 算法的基础上为通信数据增加了当前数据包跳数变量 *Hops*，通过判断 *Hops* 是否等于 0，来判断当前数据包所处节点是否为通信源节点，若为通信源节点则需计算路径长度并进行对比，输出当前节点和下一节点的转发端口；否则直接根据当前数据包所携带的转发端口推断下一节点的转发端口并输出. 改进的 LA4T 在传输模式为 TP_0 时的具体算法如下.

算法 3. 改进后的 LA4T 路由算法.

输入：起始节点 $s = s_F''s_F'\cdots s_1''s_1'$，目标节点 $t = t_F''t_F'\cdots t_1''t_1'$，当前节点 $c = c_F''c_F'\cdots c_1''c_1'$，当前焦点路由图$FRG^f$级数 f，当前数据包跳数 *Hops*，当前节点转发端口 *cur-outport*；

输出：下一节点转发端口 *next-outport*.

① begin
② if(*Hops* == 0) then /*当前节点为源节点*/
③ $Dis(Path\ A) = s_{f-1}'\cdots s_1' + t_{f-1}''\cdots t_1'' + 1$;
④ $Dis(Path\ B) = (\bar{s}_{f-1}''\cdots\bar{s}_1''\oplus\bar{s}_{f-1}'\cdots\bar{s}_1') + (\bar{t}_{f-1}''\cdots\bar{t}_1''\oplus\bar{t}_{f-1}'\cdots\bar{t}_1') + 2^{f-1} + 1$;
⑤ if $Dis(Path\ A) \leqslant Dis(Path\ B)$ then
⑥ *cur-outport* = 2; /*更新当前节点转发端口*/
⑦ *next-outport* = *reasoning*(s, v_2^f); /*计算下一节点转发端口*/

⑧ else then
⑨ *cur-outport* = 3;
⑩ *next-outport* = *reasoning*(s, v_3^f);
⑪ end if
⑫ return *next-outport*;
⑬ end if
⑭ if(*cur-outport* == 2)then/*当前节点非源节点*/
⑮ *next-outport* = *reasoning*(c, v_2^f); /*推理下一节点转发端口*/
⑯ end if
⑰ if (*cur-outport* == 3) then
⑱ *next-outport* = *reasoning*(c, v_3^f);
⑲ end if
⑳ return *next-outport*;
㉑ end

在算法 3 中, 当前焦点路由图FRG^f可根据当前节点和目标节点名进行计算, 当前数据包跳数 *Hops* 和当前节点转发端口 *cur-outport*(端口命名见图 5(a))可从头节点直接获取, 并且默认值为 0. 在计算路径长度 *Dis*(*Path A*)和 *Dis*(*Path B*)时需将节点名的二进制数$s'_{f-1}\cdots s'_2 s'_1$等转化为十进制数, 符号“+”表示普通的数值加号, 符号“⊕”表示逻辑位或运算, $\bar{s}$表示逻辑位取反运算. *reasoning*(c, v_2^f)表示根据当前节点 c 和尖端节点v_2^f(f为当前节点焦点路由图级数, 2 表示当前尖端节点为 2)来推理下一节点转发端口. 另外, 可以将其他通信模式下的路由计算通过置换群$\mathbb{S}_3$变换至TP_0传输模式下进行, 并通过逆变换返回输出端口即可. 由于篇幅有限, 关于 LA4T 算法的其他详细信息请参见文献 [38].

当通信数据进入节点后(*Hops*=0), 其转发目标端口需要根据当前节点和目标节点的编码信息进行路由计算, 由于 TriBA 节点拥有的 4 个端口(包括当地内核)接收到的数据都需经过相同算法计算, 因此使用流水线方式实现相应的计算, 如图 11 所示. 考虑到流水线过程的整个设计规模较大, 因篇幅有限, 仅对流水过程中的 4 个重要模块功能做简要介绍:

1) 数据准备阶段(PREP). 该阶段主要完成当前通信模式、当前焦点路由图的层数计算, 对应算法 3 的部分输入.

2) TP_0模式等价距离计算阶段(EDIS). 该阶段主要计算当前节点和目标节点在TP_0传输模式下的等价节点($\mathbb{S}_3$变换后的节点), 然后计算等价节点之

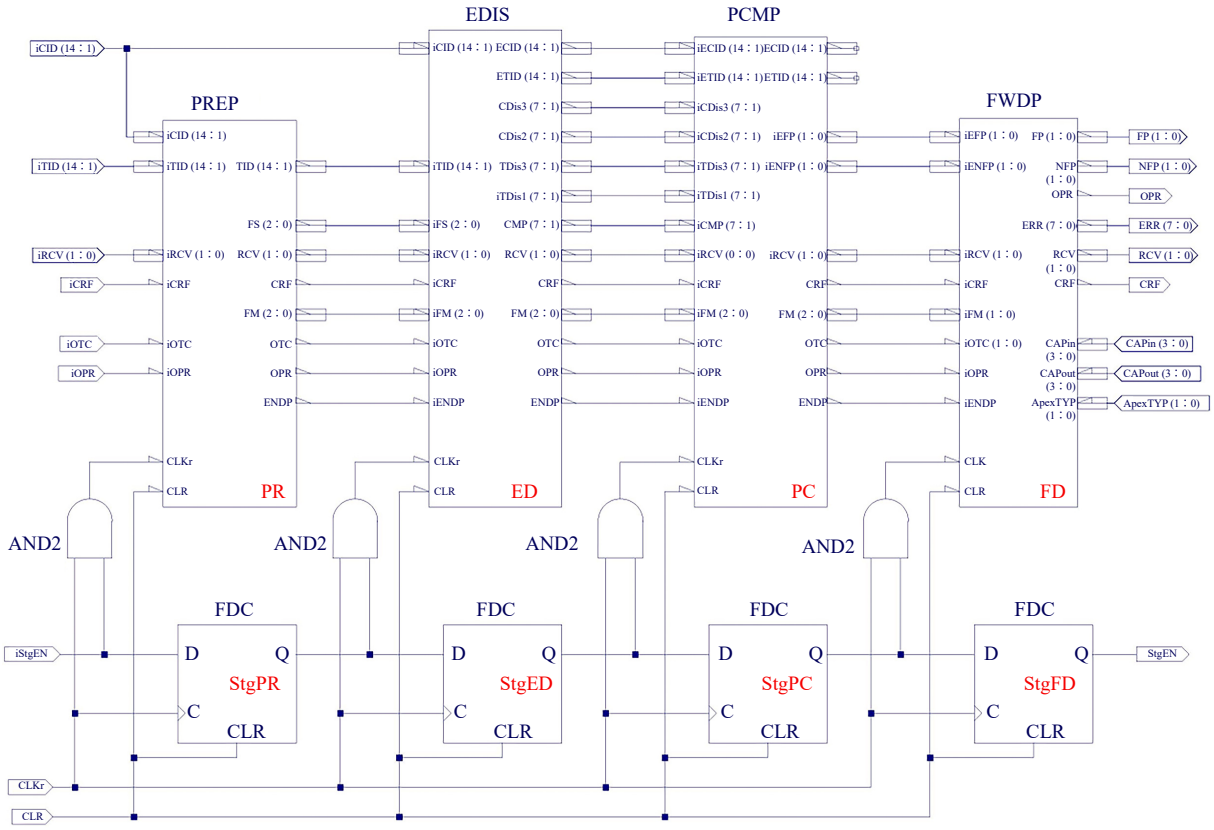


Fig. 11 Block diagram of the pipeline hardware implementation for routing computation

图 11 路由计算流水线硬件实现模块图

间的路由距离. 对应算法 3 的行③④, 其中$\mathbb{S}_3$变换相关计算可参考 LA4T 算法原文[38].

3) 等价模式备选路径比较阶段(PCMP). 该阶段的主要完成对 *Path A* 和 *Path B* 备选路径的长度比较, 进而给出等价模式中的转发端口, 以及预测下个节点处的备选转发端口(该端口与 *Path A* 和 *Path B* 相关, 在下个流水段 FWDP 才能具体给出转发端口的选择). 对应算法 3 中行⑤~⑫的部分内容.

4) 转发端口判定阶段(FWDP). 将 PCMP 输出的在等价模式TP_0下的转发端口通过$\mathbb{S}_3^{-1}$逆变换到原模式, 作为备选转发端口. 接着判断当前节点若是尖端时需要对消息转发进行特殊处理, 最终根据所满足的具体条件给出最终转发端口. 对应算法 3 中行⑤ ~ ⑲部分内容.

4.2 无死锁路由机制

在通信发起节点, 首先需要根据当前节点和目的节点的信息计算此次数据传输在其生命周期中将要经历的最大哈密顿边层次, 进而为其分配相应的缓冲区, 并规定在其生命周期结束前始终保持缓冲区不变, 保证不同缓冲区之间的独立数据传输; 其次, 需要计算数据传输方向, 并为其分配虚拟通道, 规定每次数据转发需在规定的虚拟通道中进行, 从而能够有效避免在同一个哈密顿环中因相向的数据传输而产生的资源竞争问题. 根据以上原理, 设计了缓冲区选择和控制(buffer select and control, BufSanCon)算法, 具体算法内容如下:

算法 4. 缓冲区选择和控制算法 BufSanCon.

输入: 起始节点 $s=\bar{x}_{F\cdots f+1}s_f\cdots s_2s_1$, 目标节点 $t=\bar{x}_{F\cdots f+1}t_f\cdots t_2t_1$, 焦点路由图$\mathrm{FRG}^f$;

输出: 当前传输缓冲区类型 *Buffer_Type* 和传输虚拟通道 *VC*.

① begin
② *Buffer_Type*, *Ham_OriDirection* ← −1;
③ *Path*, *TP* ←计算当前传输路径和通信模式;
④ *Axis* ← call(DetAxiSym); /*计算对称轴*/
⑤ *Buffer_Type* ←call(GetBufType); /*计算缓冲区类型*/
⑥ *Ham_OriDirection* ←call(RecurLinList); /*计算传输方向, RecurLinList 为循环链表*/
⑦ if(*Ham_OriDirection* == 0)then
⑧ *VC* ←分配 *Buffer_Type* 中的逆时针 *VC*;
⑨ else then
⑩ *VC* ←分配 *Buffer_Type* 中的顺时针 *VC*;
⑪ end if
⑫ return *Buffer_Type*, *VC*;
⑬ end

通过将改进的 LA4T 路由算法和 BufSanCon 算法结合在一起, TriBA 能够获得新的无死锁路由机制, 命名为哈密顿最短路径路由(Hamiltonian shortest path routing, HamSPR)机制.

4.3 无死锁说明

形成死锁的 4 个必要条件为互斥、请求与保持、不可剥夺以及循环等待, 如果打破其中的任意一个就可能预防死锁的发生[41]. 本文从消息的传输行为出发, 提出了基于同步哈密顿环的死锁预防策略来分别解决可能出现的路由级和协议级死锁, 具体无死锁说明如下:

1) 采用分组传输和分时复用的策略预防协议级死锁. 在TC^F中, 将所有的数据传输按照其在整个传输生命周期过程中经历的最大哈密顿边层数进行划分, 形成若干组, 并为不同的组分配独立的缓冲区, 规定数据在整个传输生命周期中保持组别不变, 保证不同组之间的独立数据传输, 不会相互交叉和干扰. 另外, 采用分时复用的策略解决不同组之间对于同一物理通道的竞争. 通过制定上述策略, 从而打破形成死锁的必要条件之一——循环等待.

2) 采用同步且划分传输方向的策略预防路由级死锁. 在同一哈密顿环内, 规定数据在同一时刻进行同步向前传输, 预防前后数据因竞争同一虚拟通道而可能出现的请求和保持现象. 另外, 通过将同一哈密顿环内不同方向(顺时针和逆时针)的数据传输限制在自己的独立通道中, 保证在同一个缓冲区中的顺时针和逆时针传输相互独立且互不交叉, 进一步预防了在同一缓冲区中数据传输可能出现的资源竞争. 通过制定上述策略, 能够打破形成死锁的必要条件之一——请求和保持.

5 实验评估

在本节中, 通过使用修改后的 GEM5[42] 仿真平台中内嵌的 Garnet 3.0 片上网络模拟器[43] 和 DSENT 工具[44], 分别在系统仿真(system emulation, SE)模式下使用合成流量以及在全系统(full system, FS)模式下使用 PARSEC[45] 基准测试集测试了 HamSPR 路由机制的性能. 另外, 在仿真过程中为了实现独立的、可操作的不同通信通道以减少其他通信的干扰, 使用了虚拟网络和虚拟通道技术.

图 12 为 HamSPR 在 GEM5 中实现的路由器微架

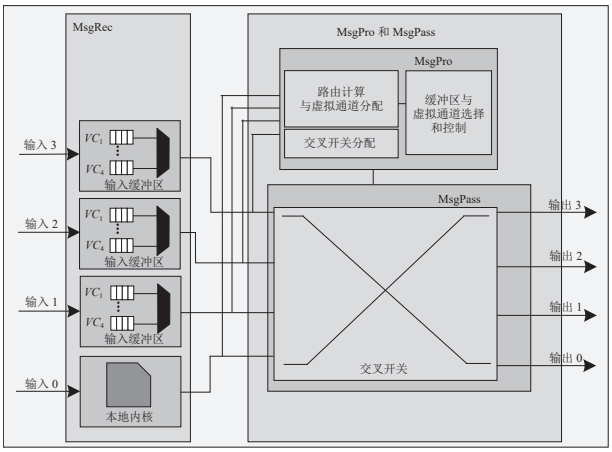


Fig. 12 Router microarchitecture implemented in GEM5 for HamSPR

图 12 HamSPR 在 GEM5 中路由器微架构的实现

构, 该路由器由 3 部分组成, 分别为消息接收(message receive, MsgRec)单元、消息处理(message process, MsgPro)单元和消息转发(message pass, MsgPass)单元. MsgRec 单元分别由 3 个输入缓冲区(虚拟网络)和 1 个本地内核组成, 其中每一个缓冲区中包含 4 个虚拟通道(virtual channel, VC), 而本地内核负责消息的接收和新消息的发送. MsgPro 单元则主要负责消息头节点信息的处理, 包括路由计算、虚拟通道分配、缓冲区及虚拟通道的选择和控制以及交叉开关分配, 其中缓冲区及虚拟通道的选择和控制只针对新的消息头节点, 并且其在交叉开关分配之前. MsgPass 单元则由一组 4×4 交叉开关组成, 负责消息的转发.

5.1 实验配置

为了全面评估 HamSPR 的性能, 选取了 TriBA 中具有代表性的查找表路由算法 Table、最短路径路由算法 SPR4T 和改进后的向前查找路由算法 LA4T 作为对比方法. Table 采用基于不同链路权重的防死锁方法[11], SPR4T 和 LA4T 采用基于尖端数据分类的防死锁方法[39].

在 SE 模式下, 选取位补码(bit-complement)、位旋转(bit-rotation)、统一随机(uniform-random)和转置(transpose)这 4 种常用的流量模式, 以平均数据包延迟(总延迟周期/接收数据包)、吞吐量(数据包/(节点×周期))和平均数据包功耗(总功耗/接收数据包)为评价指标评估 4 种不同路由算法的性能. 在 FS 模式下, 分别测试 PARSEC 中 12 种来自不同领域的应用程序的运行时间、平均数据包功耗、平均数据包延迟和平均数据包跳数(总跳数/接收数据包). 另外, 以 Table 算法测试数据为基准, 分别计算了其他算法的归一化运行时间和归一化平均数据包功耗.

仿真实验配置如下: 处理器: Intel® Xeon® CPU E3-1225 v5 @3.31 GHz; 缓存: L1-256 KB, L2-1.0 MB, L3-8.0 MB; 操作系统: Ubuntu (20.04.6 LTS); DSENT 工艺模型为 11 nm (tri-gate). 另外, 为每个缓冲区设置了 1 个空槽, 以保证哈密顿环中的数据传输能够正常地运行而不发生阻塞. 基于 X-Y 和 West-first 算法的 2D-Mesh 除了使用 2D-Mesh 的网络拓扑、X-Y 和 West-first 路由算法外, 其余在 FS 模式下的仿真参数均与 TriBA 相同. SE 模式和 FS 模式下的具体仿真参数见表 2. 其余仿真参数均遵循 GEM5 仿真器以及 Garnet 3.0 和 DSENT 中指定的默认值.

Table 2 Simulation Parameters in SE and FS Modes

表 2 SE 模式和 FS 模式下的仿真参数

模式	参数	数值
SE	节点数	27
	仿真和预热/周期	40 000, 1 000
	虚拟网络和通道	3, 4
	路由器延迟/周期	4
	线路延迟/周期	1
	死锁阈值/周期	150
	缓存一致性协议	Garnet Standalone
	数据包交换	Wormhole
	Flit 大小/b	128
	SA 分配策略	Round-Robin
	CPU 频率/GHz	1.0
FS	ISA	x86
	系统内核	x86-Linux-kernel-4.19.83
	CPU 类型	x86KVMCPU, 2.0 GHz
	L1 cache	Private, 16 KB, 2-way associative, 64 B block size
	L2 cache	Shared, 2 MB, 8-way associative, 64 B block size
	内存类型	SimpleMemory, 3 GB
	缓存一致性协议	MESI Two Level
	基准测试集规模	simsmall

5.2 实验结果及讨论

5.2.1 合成流量

在 SE 模式下, 所有模拟核心不执行实际指令, 它们只充当流量注入器, 将数据包引入网络. 这种方法使系统独立于指令集体系结构(ISA), 并使内存子系统保持静默, 从而实现对模拟系统中网络性能的更精确评估[42]. 如图 13 所示, 评估了 4 种不同路由算法分别在 bit-complement, uniform-random, transpose, bit-rotation 流量模式下的平均数据包延迟和吞吐量.

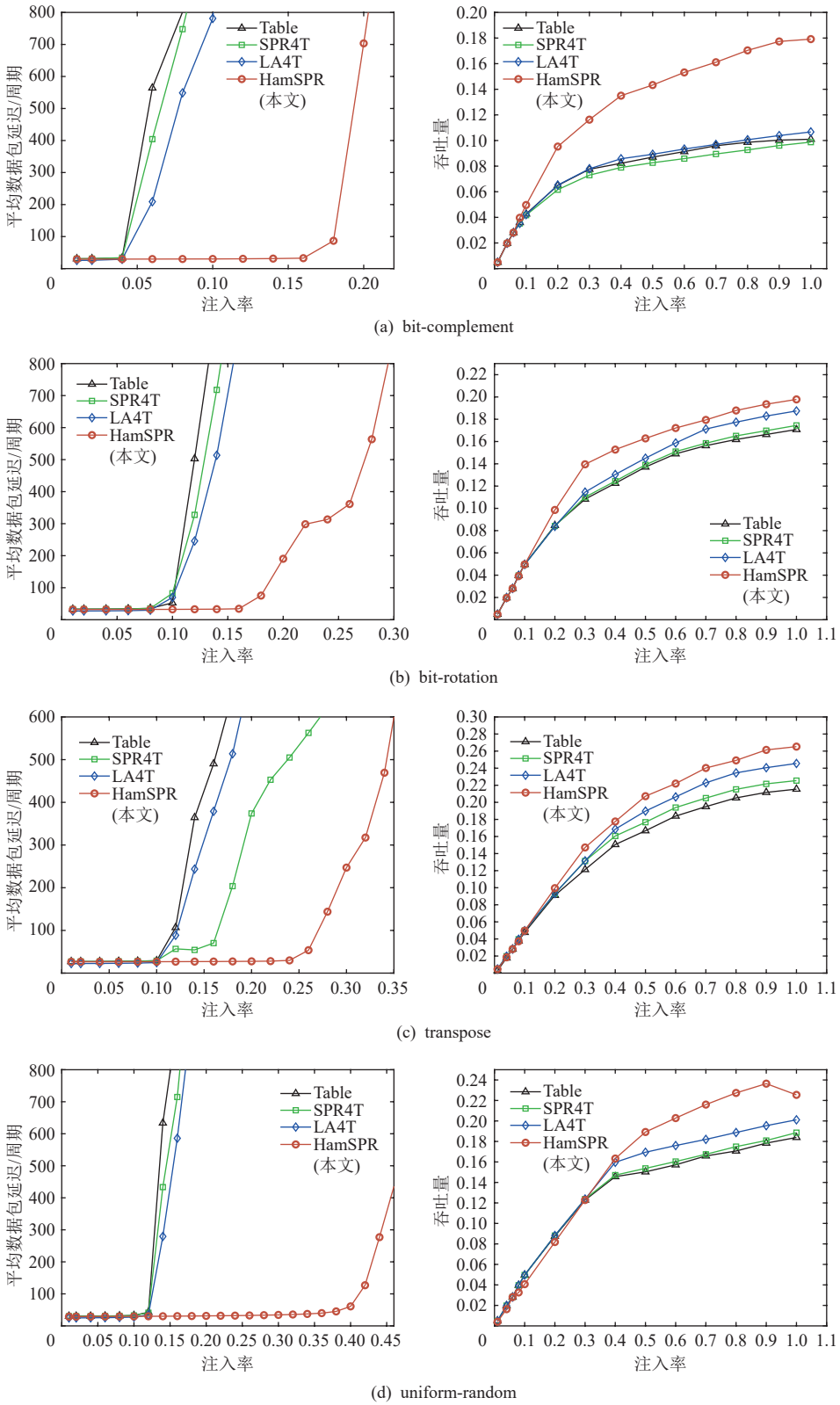


Fig. 13 Average data packet latency and throughput of different routing algorithms in TC^3 under synthetic traffic patterns

图 13 TC^3 中不同路由算法在合成流量模式下的平均数据包延迟和吞吐量

由于 Table 未对不同的数据传输进行分类，只是根据不同链路的权重来判断无死锁的传输路径，并且在数据传输过程中由于多次计算路径权重而可能产生较高的延迟，使其性能不如其他 3 种算法. 而采用基于尖端数据分类的 SPR4T 和 LA4T 算法表现出了比 Table 更优的性能. 其中，由于 LA4T 采用了前瞻路由

策略, 相比于 SPR4T 能够有效减少路由器流水线级数、提高数据转发效率, 从而实现了更低的延迟和更高的吞吐量. 尽管该类防死锁方法能够根据数据包目的地址将所有的数据传输分为 3 类, 并且每类中的数据传输独占各自的缓冲区, 有效地防止协议级死锁. 但是 LA4T 没有考虑不同传输类之间的潜在数据干扰以及同类之间的缓冲区竞争问题. HamSPR 相比于其他 3 种方法表现了更优的性能, 这主要受益于高效的“环流传输”策略. HamSPR 采用了以下 4 种策略保障了网络中数据包能够沿着环流快速地传输: 1)根据数据包经历的最大哈密顿边层次为其分配独立的存储转发缓冲区, 避免了不同缓冲区之间的数据传输干扰; 2)对同一缓冲区的不同数据传输划分了传输方向并分配独立的虚拟通道, 避免同一缓冲区内数据传输对虚拟通道的竞争; 3)路由器采用全局时钟同步策略, 保证了环流中的数据同步向前传输; 4)在每个缓冲区中设置了空槽, 从而确保即使在负载较高的情况下环流依然能够正常的运行.

相比于 Table, SPR4T, LA4T 方法, HamSPR 在 4 种流量模式下分别实现了 18.18%～65.40% 的平均数据包延迟降低以及 8.00%~59.17% 的吞吐量提升. 另外, 除了 HamSPR 方法, 在 bit-complement 流量下的其他算法均表现出了较低的吞吐量, 其原因是数据包在网络中出现了较高的等待延迟. 当尝试将死锁阈值修改为文献 [46] 中的 128 周期时, 其他 3 种算法在较高的注入率(大于 0.8)就会因数据包等待周期超过死锁阈值而产生死锁. 在 uniform-random 流量模式下, 当注入率超过 0.9 时 HamSPR 的吞吐量呈现出了下降的趋势, 其原因可能是在有限的缓冲区数量下当向网络中注入的数据包超过网络的饱和值时, 哈密顿环流中传递的数据包一直处于满载的状态, 此时需要等待环流中出现新的空闲缓冲区才能向其中注入新的数据包, 从而导致网络的吞吐量略微降低.

5.2.2 PARSEC 测试集中的实验

从图 14 中能够发现, 每个程序的性能改进并不相同, 这与各个程序的通信模式相关[45]. HamSPR 在通信密集型的程序中表现出了较明显的性能优势, 例如在 canneal 程序中与其他 3 种算法相比, HamSPR 分别降低了 8.60%, 12.72%, 2.84% 的程序运行时间. 而对于像 streamcluster 和 x264 类型的程序, 它们要么在多个 CPU 之间表现出高通信需求(高负载), 要么在短时间内产生更多的网络流量(突发流量), 使其性能变化较明显, 相比于基线分别降低了 13.20% 和 14.90% 的运行时间. HamSPR 相比于其他算法的性能优势可归因于 2 点: 1)HamSPR 中的不同哈密顿环之间相互独立, 减少了不同数据包之间的相互干扰, 提高了数据传输速度; 2)哈密顿环中的各个数据包同步向前传输, 避免了因前方缓冲区被占用而出现等待的现象, 降低了数据包之间的等待延迟.

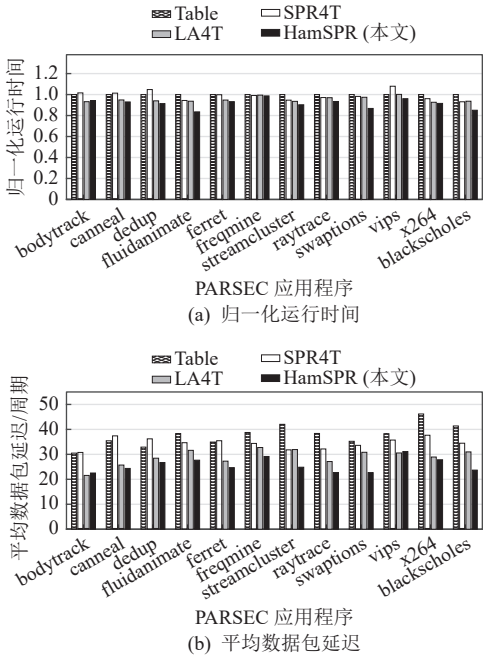


Fig. 14 Normalized runtime and average data packet latency of different routing algorithms in TC^3 under PARSEC applications

图 14 TC^3 中不同路由算法在 PARSEC 应用程序下的归一化运行时间和平均数据包延迟

从互连网络中获得较明显的性能改进的应用程序往往与较低的数据包延迟和平均数据包跳数有关[30]. 如图 14 和图 15 所示, HamSPR 相比于其他算法在平均数据包延迟和平均数据包跳数方面分别实现了平均 22.20% 和 16.11% 的降低. 由于 HamSPR 采用了同步哈密顿环的数据传输方式, 与其他路由算法相比不需要费时地查找路由表操作, 并且在每一次转发时不需要相对复杂的计算步骤, 其采用的高效的同步传输机制和独立的数据传输方式使其能够实现较快的数据传输. 因此, 相比于其他算法, HamSPR 能够有效提高数据传输效率, 与基线 x264 相比最高可实现 42.75% 的平均数据包延迟降低. 此外, 由于受到应用程序通信模式的影响, HamSPR 在部分应用程序上表现出了与其他路由算法相同或略高的平均数据包延迟, 例如 blackscholes 和 swaptions. 这可能是由于当应用程序的通信密度较低、通信之间的干扰较小, 并且对于缓冲区的竞争不高时, 整个网络处于未饱和

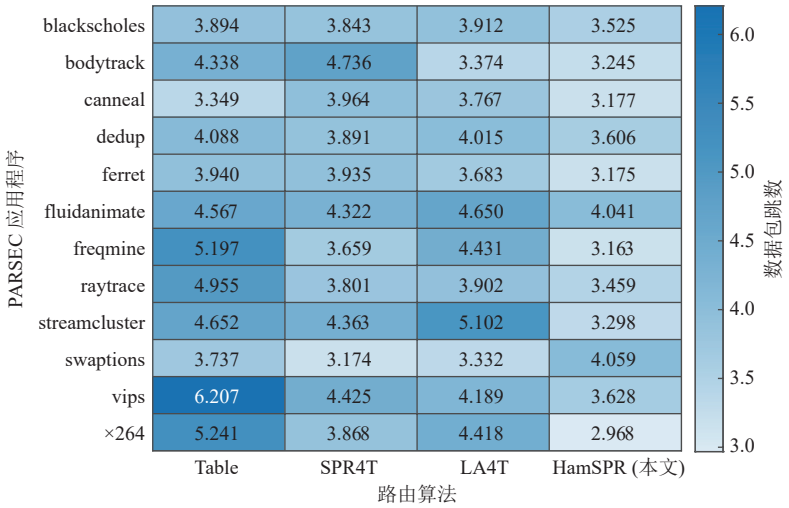


Fig. 15 Average data packet hops of different routing algorithms in TC^3 under PARSEC applications

图 15 TC^3 中不同路由算法在 PARSEC 应用程序下的平均数据包跳数

的状态. 因此不能完全发挥出基于哈密顿环的数据传输方式的优势, 导致在平均数据包延迟上 HamSPR 与其他路由算法基本保持一致.

为了进一步分析基于 HamSPR 算法的 TriBA 架构性能, 在一个包含 9 个核心的网络配置中比较了使用 X-Y 和 West-first 算法的 2D-Mesh(Mesh-XY, Mesh-West-first)和使用 HamSPR 算法的 TriBA(TriBA-HamSPR)在运行 PARSEC 测试集下的性能. X-Y 和 West-first 算法作为 2D-Mesh 典型的无死锁分布式确定性和自适应路由算法, 被广泛地研究和使用[47-48]. 图 16 展示了 Mesh-XY, Mesh-West-first, TriBA-HamSPR 在运行不同应用程序时的性能比较. 对于各种应用程序, TriBA-HamSPR 相对于 Mesh-XY 和 Mesh-West-first 在程序运行时间和平均数据包延迟方面分别实现了 1.00% ~ 10.04% 和 5.34 %~ 20.32% 的降低, 特别在通信密集型应用程序(如 dedup 和 swaptions)中尤为显著. 这主要归因于 TriBA 拓扑具有的哈密顿特性和旋转变换特性[3]. 基于哈密顿的路由算法在解决潜在的死锁问题的同时能够借助哈密顿环流实现较高的通信速率, 而特有的旋转对称性降低了路由的复杂性和计算负载[3], 在一定程度上提高了数据包的传输效率.

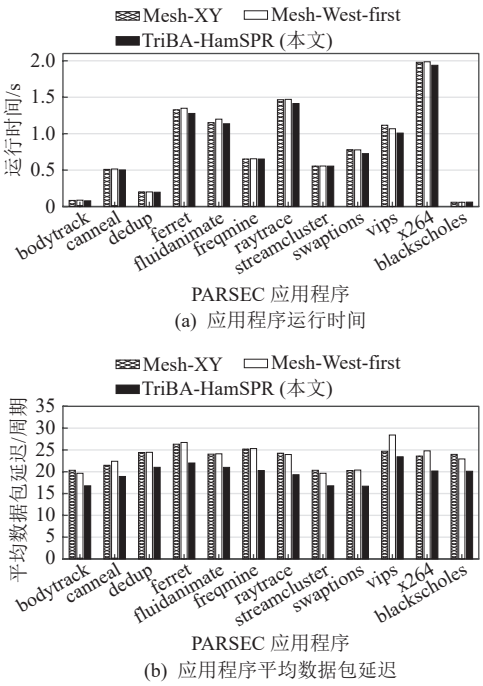


Fig. 16 Runtime and average data packet latency of routing algorithms with different architectures under PARSEC applications

图 16 不同架构的路由算法在 PARSEC 应用程序下的运行时间和平均数据包延迟

5.2.3 可扩展性分析

在 SE 模式下, 测试了 HamSPR 相比其他算法在不同网络规模下的平均数据包延迟和吞吐量的性能变化情况, 如表 3 所示. 其中平均数据包延迟的降低和吞吐量的提升分别为 4 种不同流量模式下的平均值. 从表 3 中的方差数值可知, 随着网络规模的变化, HamSPR 算法相比于 Table, SPR4T, LA4T 算法在平均数据包延迟和吞吐量上的性能变化并未发生较大的波动, 即 HamSPR 及其防死锁原理的网络性能并未受到网络规模的较大影响, 具有良好的可扩展性.

5.2.4 硬件开销分析

在本节中, 利用 DSENT 在 11 nm 工艺技术下评估了 NoC 中路由器和链路的总功耗以及路由器的面积. 图 17(a)展示了 4 种路由算法在 4 种流量模式下

Table 3 Performance Variation of HamSPR Compared with Other Algorithms with Different Network Scales

表 3 HamSPR 在不同网络规模下相比其他算法的性能差异

网络规模	平均数据包延迟降低/%			吞吐量提升/%		
	Table	SPR4T	LA4T	Table	SPR4T	LA4T
9	46.67	43.84	40.75	27.29	26.29	21.21
27	48.28	45.78	39.46	27.35	27.35	19.69
81	45.35	43.94	37.64	25.86	26.56	18.74
243	45.95	42.98	37.17	26.08	26.44	19.22
方差	1.205 3	1.042 2	2.063 1	0.466 2	0.169 1	0.859 7

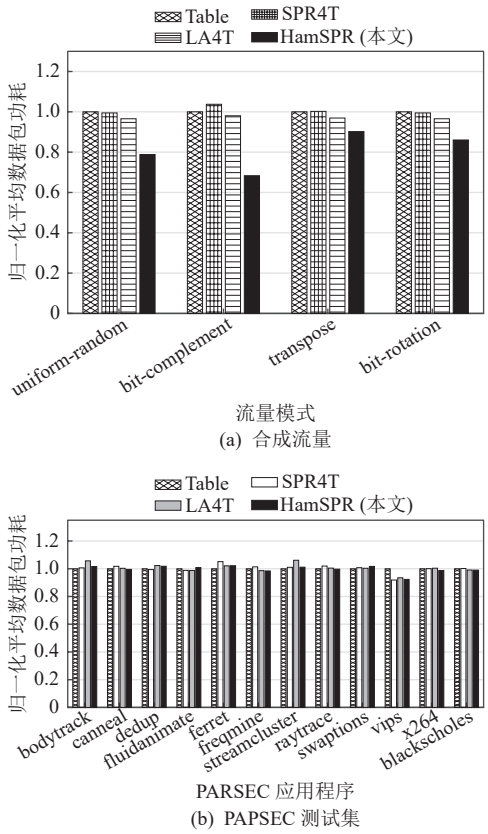


Fig. 17 Normalized average data packet power consumption of different routing algorithms in TC^3 under the synthetic traffic and PARSEC applications

图 17 TC^3 中不同路由算法在合成流量和 PARSEC 应用程序下的归一化平均数据包功耗

的归一化平均数据包功耗, 相比于基线(Table 算法), HamSPR 在 4 种不同注入率下的平均数据包功耗实现了平均 19.16% 的降低. 以上功耗的降低可归因于 HamSPR 能够借助哈密顿环流实现更高的数据包传输速度, 从而随着注入率的增加能够获得比其他算法更优的吞吐量. HamSPR 在 PARSEC 基准测试集下的平均数据包功耗几乎等于或者小于其他 3 种算法, 如图 17(b)所示. 这表明在运行相同的应用程序时, 本文设计的基于同步哈密顿环的无死锁策略即使同时采用了不同的方案来消除潜在的路由级死锁和协议级死锁, 也没有因此给整个网络带来明显的功耗增加.

NoC 中的路由器主要由缓冲区、交叉开关、交换分配器、路由计算等功能模块组成. 通过使用 DSENT 计算 TriBA 架构在 GEM5 中的路由器面积可知, 缓冲区占据了整个路由器面积的 67.67%, 交叉开关和交换分配器分别占据了路由器面积的 19.36% 和 3.27%, 而其余组件(消息处理单元等)共同占据了剩余 9.70% 的面积. 相比于 Table, SPR4T, LA4T 算法, HamSPR 的 BVSC 计算单元通过增加计算和逻辑电路以实现缓冲区及虚拟通道的选择和控制, 导致路由器面积相比于其他算法平均增加了约 1.50%. 考虑到 HamSPR 整体性能优势, 额外的面积开销所带来的影响是可以接受的.

6 结 论

本文针对具有哈密顿特性的基三众核架构提出了基于同步哈密顿环的无死锁路由策略, 该策略能够将网络中的数据传输根据其所经历的最大哈密顿边层次进行分组, 并且组内的数据传输独用各自的传输通道, 同时结合了全局时钟同步和局部类气泡流控方法, 这样做具有以下好处: 1)不同组间的数据传输相互独立, 避免了不同类型数据传输的相互干扰和资源竞争, 在解决潜在的协议级死锁的同时提高了数据传输速度; 2)不同组内的数据传输独用各自的虚拟通道且同步向前传输, 避免了同一组内前后数据传输因竞争缓冲区而产生的路由级死锁, 并降低了数据传输延迟; 3)由于是从拓扑结构自身出发设计的死锁解决方案, 因此对于任何的流量场景不需做任何的优化即可解决死锁, 不影响架构的通用性. GEM5 仿真实验结果表明, 本文所提出的方法与 TriBA 中现有的解决方案以及采用 X-Y 和 West-first 无死锁路由的 2D-Mesh 架构相比均具有明显的性能优势. 本文工作进一步提高了 TriBA-NoC 架构的通信性能和可用性, 使其更具潜力应用在未来的 MPSoC 上. 此外, 由于在其他众多拓扑结构中, 如 2D-Mesh, Torus 等都存在哈密顿特性, 因此本文提出的方法对于其他众核架构, 如 Godson-T 同时解决同类问题具有较高的借鉴意义.

另外，本文提出的基于同步哈密顿环的无死锁策略还有需要完善的地方：1）目前该策略面向拓扑已知的 TriBA 架构，影响了其适用性；2）对于大型或者超大型规模的网络，需要较多的缓冲区资源. 未来将做 3 点改进：1）修改当前的对称轴、缓冲区判断等算法，使其能够适用于任何具有哈密顿特性的多核架构；2）采用转注策略，动态调整网络的拥塞程度，最小化使用缓冲区资源；3）识别故障链路，并借鉴循环网络中的容错机制以避开故障链路，提高网络的容错性.

作者贡献声明：李春峰提出研究思路、设计算法和实验方案、进行实验验证和数据分析，并撰写和修改论文；Karim Soliman 参与实验验证和数据分析；计卫星参与论文审阅与修订；石峰负责全文结构设计与指导，并审阅与修订论文.

参 考 文 献

[1] Yazdanpanah F, Afsharmazayejani R. A systematic analysis of power saving techniques for wireless network-on-chip architectures[J]. Journal of Systems Architecture, 2022, 126: 102485

[2] Della Vecchia G, Sanges C. A recursively scalable network VLSI implementation[J]. Future Generation Computer Systems, 1988, 4(3): 235−243

[3] Shi Feng, Chen Xu, Yin Fei, et al. A shortest path routing mechanism based on S3 for TriBA-Net[J]. SCIENTIA SINICA Informationis, 2018, 48(1): 100−114 (in Chinese)
(石峰, 陈旭, 尹飞, 等. 基于 S3 变换的 TriBA-Net 最短路径路由机制[J]. 中国科学: 信息科学, 2018, 48(1): 100−114)

[4] Zhang Yang, Wang Da, Ye Xiaochun, et al. A global hierarchical adaptive routing mechanism in many-core processor network-on-chip[J]. Journal of Computer Research and Development, 2016, 53(6): 1211−1220 (in Chinese)
(张洋, 王达, 叶笑春, 等. 众核处理器片上网络的层次化全局自适应路由机制[J]. 计算机研究与发展, 2016, 53(6): 1211−1220)

[5] Feng Yinxiao, Xiang Dong, Ma Kaisheng. Heterogeneous die-to-die interfaces: Enabling more flexible chiplet interconnection systems[C]//Proc of the 56th Annual IEEE/ACM Int Symp on Microarchitecture. New York: ACM, 2023: 930−943

[6] Bakhouya M, Suboh S, Gaber J, et al. Analytical performance comparison of 2D mesh, wk-recursive, and spidergon nocs [C/OL]// Proc of the 19th IEEE Int Symp on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW). Piscataway, NJ: IEEE, 2010 [2024-05-22]. https://ieeexplore.ieee.org/abstract/document/5470784

[7] Shi Feng, Wei Zenghui, Wang Yizhuo, et al. A layout and routing scheme named tMesh for a multi/many-core architecture TriBA-CMPs: China, CN107526894A[P]. 2017-12-29 (in Chinese)
(石峰, 魏增辉, 王一拙, 等. 一种多/众核架构 TriBA-CMPs 的布局布线方案 tMesh: 中国, CN107526894A[P]. 2017-12-29)

[8] Braham C C, Ji Weixinga, Zhang Lingyu. The design and implementation of a hierarchical network on chip[J]. Procedia Engineering, 2012, 29: 2966−2974

[9] Feng Shi, Yang Zhang. Design and evaluation of low-latency and shortest-path routing algorithm for triplet-based hierarchical interconnection network[J]. Journal of Testing and Evaluation, 2013, 41(4): 541−550

[10] Khan HUR, Shi Feng, Ji Weixing, et al. Computationally efficient locality-aware interconnection topology for multi-processor system-on-chip (MP-SoC)[J]. Chinese Science Bulletin, 2010, 55(29): 3363−3371

[11] Wang Zuo, Zuo Qi, Li Jiaxin. Triplet-based topology for on-chip networks[J]. WSEAS Transactions on Computers, 2009, 8(3): 516−525

[12] Xue Licheng, Shi Feng, Ji Weixing. 3D floorplanning of low-power and area-efficient network-on-chip architecture[J]. Microprocessors and Microsystems, 2011, 35(5): 484−495

[13] Zhang Yang, Shi Feng, Zuo Qi, et al. Express router microarchitecture for triplet-based hierarchical interconnection network[C]//Proc of the 14th IEEE Int Conf on High Performance Computing and Communication & 9th IEEE Int Conf on Embedded Software and Systems. Piscataway, NJ: IEEE, 2012: 295−302

[14] Wang Xiaojun, Shi Feng, Zhang Hong. KLSAT: An application mapping algorithm based on Kernighan–Lin partition and simulated annealing for a specific WK-recursive NoC architecture[C]//Proc of the 16th IFIP WG 10.3 Int Conf on Network and Parallel Computing. Berlin: Springer, 2019: 31−42

[15] Zhang Hong, Wang Xiaojun. KGT: An application mapping algorithm based on Kernighan–Lin partition and genetic algorithm for WK-recursive NoC architecture[C]//Proc of the 17th Int Conf on Intelligent Computing Theories and Application (ICIC). Berlin: Springer, 2021: 86−101

[16] Shi Feng, Li Jiaxin, Bai Ziru. Performance of triplet-based interconnection strategy for multi-core on-chip processors[C]//Proc of the 11th IEEE Int Conf on High Performance Computing and Communications. Piscataway, NJ: IEEE, 2009: 163−170

[17] Wang Xiaojun, Shi Feng, Zhang Hong. A novel speedup evaluation for multicore architecture based topology of on-chip memory[C]//Proc of the 10th Int Symp on Parallel Architectures, Algorithms and Programming. Berlin: Springer, 2019: 35−47

[18] Feng Yinxiao, Xiang Dong, Ma Kaishen. A scalable methodology for designing efficient interconnection network of chiplets [C]//Proc of the 2023 IEEE Int Symp on High-Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2023: 1059−1071

[19] Kasan H, Kim J. VVQ: Virtualizing virtual channel for cost-efficient protocol deadlock avoidance[C]//Proc of the 2023 IEEE Int Symp on High-Performance Computer Architecture (HPCA). Piscataway, NJ:

IEEE, 2023: 1072−1084

[20] Romanov A Y, Myachin N M, Lezhnev E V, et al. Ring-Split: Deadlock-free routing algorithm for circulant networks on chip[J]. Micromachines, 2023, 14(1): 141

[21] Kwauk G, Kang S, Kasan H, et al. Boomgate: Deadlock avoidance in non-minimal routing for high-radix networks[C]//Proc of the 2021 IEEE Int Sym on High-Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2021: 696−708

[22] Flich J, Skeie T, Mejia A, et al. A survey and evaluation of topology-agnostic deterministic routing algorithms[J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 23(3): 405−425

[23] Wu Yibo, Wang Liang, Wang Xiaohang, et al. A deflection-based deadlock recovery framework to achieve high throughput for faulty NoCs[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 40(10): 2170−2183

[24] Razavi S, Sarbazi-Azad H. The triangular pyramid: Routing and topological properties[J]. Information Sciences, 2010, 180(11): 2328−2339

[25] Wang Nengchung, Yen Chengpang, Chu Chihping. Multicast communication in wormhole-routed symmetric networks with Hamiltonian cycle model[J]. Journal of Systems Architecture, 2005, 51(3): 165−183

[26] El-Obaid A. Three-dimension Hamiltonian broadcast wormhole-routing[J]. International Journal of Computer Networks & Communications, 2015, 7(3): 31−46

[27] Chen Xu, Shi Feng, Yin Fei, et al. An efficient distributed minimal routing algorithm for triplet-based WK-recursive network[C]//Proc of the 20th IEEE Int Conf on High Performance Computing and Communications; 16th IEEE Int Conf on Smart City; 4th IEEE Int Conf on Data Science and Systems (HPCC/SmartCity/DSS). Piscataway, NJ: IEEE, 2018: 547−554

[28] Fan Dongrui, Zhang Hao, Wang Da, et al. Godson-T: An efficient many-core processor exploring thread-level parallelism[J]. IEEE Micro, 2012, 32(2): 38−47

[29] Parasar M, Farrokhbakht H, Jerger N E, et al. DRAIN: Deadlock removal for arbitrary irregular networks[C]//Proc of the 2020 IEEE Int Symp on High Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2020: 447−460

[30] Dally W J, Towles B P. Principles and Practices of Interconnection Networks[M]. Amsterdam: Elsevier, 2004

[31] Ramey C. Tile-GX100 manycore processor: Acceleration interfaces and architecture[C/OL]//Proc of the 23rd IEEE on Hot Chips Symp. Piscataway, NJ: IEEE, 2011[2024-05-22]. https://ieeexplore.ieee.org/abstract/document/7477491

[32] Wentzlaff D, Griffin P, Hoffmann H, et al. On-chip interconnection architecture of the tile processor[J]. IEEE Micro, 2007, 27(5): 15−31

[33] Sodani A, Gramunt R, Corbal J, et al. Knights landing: Second-generation intel Xeon Phi product[J]. IEEE Micro, 2016, 36(2): 34−46

[34] Rahman R. Intel Xeon Phi Coprocessor Architecture and Tools: The Guide for Application Developers[M]. Berlin: Springer, 2013

[35] Qiao Baojun, Shi Feng, Ji Weixing. A new hierarchical interconnection network for multi-core processor[C]//Proc of the 2nd IEEE Conf on Industrial Electronics and Applications. Piscataway, NJ: IEEE, 2007: 246−250

[36] Wang Zuo, Shi Feng. A shortest path routing algorithm in triplet-based network[J]. Transactions of Beijing Institute of Technology, 2009, 29(5): 410−414

[37] Zhang Yang, Shi Feng, Ji Weixing, et al. SPORT: A shortest path routing algorithm for triplet-based hierarchical interconnection[J]. Transactions of Beijing institute of Technology, 2013, 33(1): 57−61

[38] Chen Xu, Shi Feng, Yin Fei, et al. A novel look-ahead routing algorithm based on graph theory for triplet-based network-on-chip router[J]. IEICE Electronics Express, 2018, 15(8): 1−12

[39] Shi Feng, Ruan Shenqiang, Yin Fei, et al. A deadlock prevention method for inter-core networks based on apex-subgraph classification for data transmission in triplet-based architecture: China, CN112698960B[P]. 2022-08-09 (in Chinese)
(石峰, 阮生强, 尹飞, 等. 基于尖端子图分类传输数据的基三核间网络防死锁方法: 中国, CN112698960B[P]. 2022-08-09)

[40] Tutte W T. Graph Theory[M]. Cambridge, UK: Cambridge University Press, 2001

[41] Levine G N. The classification of deadlock prevention and avoidance is erroneous[J]. ACM SIGOPS Operating Systems Review, 2005, 39(2): 47−50

[42] Binkert N, Beckmann B, Black G, et al. The GEM5 simulator[J]. ACM SIGARCH Computer Architecture News, 2011, 39(2): 1−7

[43] Bharadwaj S, Yin J, Beckmann B, et al. Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling[C/OL]//Proc of the 57th ACM/IEEE Design Automation Conf (DAC). Piscataway, NJ: IEEE, 2020[2024-05-22]. https://ieeexplore.ieee.org/abstract/document/9218539

[44] Sun Cheng, Chen C H O, Kurian G, et al. DSENT—A tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling[C]//Proc of the 6th IEEE/ACM Int Symp on Networks-on-Chip. Piscataway, NJ: IEEE, 2012: 201−210

[45] Zhan Xusheng, Bao Yungang, Bienia C, et al. PARSEC3.0: A multicore benchmark suite with network stacks and SPLASH-2X[J]. ACM SIGARCH Computer Architecture News, 2017, 44(5): 1−16

[46] Parasar M, Jerger N E, Gratz P V, et al. SWAP: Synchronized weaving of adjacent packets for network deadlock resolution[C]//Proc of the 52nd Annual IEEE/ACM Int Symp on Microarchitecture. New York: ACM, 2019: 873−885

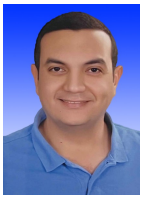
[47] Joshi B, Thakur M K. Genetic algorithm-and cuckoo search algorithm-based routing optimizations in network-on-chip[J]. Arabian Journal for Science and Engineering, 2023, 48(8): 9635−9644

[48] Luo Yaoying, Meyer M C, Jiang Xin, et al. A hotspot-pattern-aware routing algorithm for networks-on-chip[C]//Proc of the 13th IEEE Int Symp on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). Piscataway, NJ: IEEE, 2019: 229−235



Li Chunfeng, born in 1995. PhD candidate. Student member of CCF. His main research interests include computer architecture and high-performance computing.

李春峰, 1995 年生. 博士研究生. CCF 学生会员. 主要研究方向为计算机体系结构、高性能计算.



Karim Soliman, born in 1991. PhD candidate. Student member of CCF. His main research interests include computer architecture and high-performance computing.

Karim Soliman, 1991 年生. 博士研究生. CCF 学生会员. 主要研究方向为计算机体系结构、高性能计算.



Ji Weixing, born in 1980. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include computer architecture, high-performance computing, and code detection and analysis.

计卫星, 1980 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为计算机体系结构、高性能计算、代码检测与分析.



Shi Feng, born in 1961. PhD, professor, PhD supervisor. His main research interests include computer architecture and high-performance computing.

石　峰, 1961 年生. 博士, 教授, 博士生导师. 主要研究方向为计算机体系结构、高性能计算.