分类号：TP302　　学校代码:10363
密　级：公开　　学　　号:2210330183

安徽工程大学
Anhui Polytechnic University

硕士学位论文

题目 可重构系统容错路由及映射方法研究

论文作者　胡宇杨
校内导师　代广珍(副教授)
校外导师　朱标

专业学位类别　电子信息（085400）
研究方向（领域）　人工智能与系统集成

论文提交日期：2023 年 5 月 31 日

分类号：TP302　　单位代码：10363
密　级：公开　　学　　号：2210330183

题　　目　可重构系统容错路由及映射方法研究

题　　目　Research on Fault Tolerant Routing and Mapping Methods for Reconfigurable Systems

学　生　姓　名：　胡宇杨
校　内　导　师：　代广珍副教授
校　外　导　师：　朱标
专业学位类别　：　电子信息
研究方向(领域)：　人工智能与系统集成

论文答辩日期：2024 年 5 月 16 日

安徽工程大学学位论文原创性声明

本人郑重声明：我恪守学术道德，崇尚严谨学风。所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已明确注明和引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品及成果的内容。论文为本人亲自撰写，我对所写的内容负责，并完全意识到本声明的法律结果由本人承担。

学位论文作者签名： 

日期：2024 年 5 月 29 日

安徽工程大学学位论文版权使用授权书

学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅或借阅。本人授权安徽工程大学可以将本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和汇编本学位论文。

保密 □，在 ____ 年解密后适用本版权书。

本学位论文属于

不保密 ☑。

学位论文作者签名： 

日期：2024 年 5 月 29 日

指导教师签名： 

日期：2024 年 5 月 29 日

可重构系统容错路由及映射方法研究

摘 要

粗粒度可重构系统具有可配置性、加速性、低功耗等特征，已经在多核加速、物联网系统等领域获得了较为广泛的应用，其互连结构较为丰富，常用的有网格型和路由型互连，该类互连具有灵活性、低功耗的优点，容错路由型可重构系统和网格可重构阵列的流水映射已经成为国内外学术界研究的热点，为此本文对该该类问题展开研究，主要研究内容列举如下：

(1) 针对路由型可重构系统的容错算法问题，设计了一种均衡的绕行容错路由算法，其依赖于内建自测试获取故障区域的位置信息，通过结合故障模型和绕行算法，并且使用了较低复杂度的绕行策略，以 8×8 的 2D Mesh 网络为实验仿真平台，相比较已有的容错路由算法，实验结果表明，新算法平均饱和注入率分别提高了 20.81%、20.89%，新算法不仅改善了故障模型周围的负载不均衡现象，而且能够有效的降低数据包的平均延迟，网络可以保持良好的通信性能。

(2) 针对网格型粗粒度可重构计算系统的流水映射问题，提出了可重构流水分段的时延量化方法，即首先将硬件指令流水的处理过程分为取指令 FI、指令译码 ID、计算操作数地址 CO、取数据 FD、硬件配置 HC、执行 EX、写回 WB 等 7 个阶段；然后综合考虑可重构阵列块间和块内映射依赖；最后在此基础上设计了一种低延迟映射算法，该算法采用路由布线与运算单元寄存器堆相结合的方式。基于一

组基准程序，取 $PEA_{3\times3}$ 和 $PEA_{4\times4}$ 对层流水算法进行时延量化统计，其中 EWF 的层流水时延分别为 74cycles、79cycles，FDCT 的层流水时延分别为 47cycles、39cycles，表明了时延量化方法的可行性；同时选取 $PEA_{4\times4}$ 进行映射算法的时延比较，相比较寄存器感知应用映射算法，新算法获得了流水总时延性能的优化，流水平均总时延降低了 30.1 %，从而验证了低延迟映射算法的有效性。

(3) 针对路由型粗粒度可重构计算系统的非流水映射时延消耗问题，给出了一种路由映射总时延的量化分析方法，该方法综合考虑了计算单元、路由数据传输等多个因素，并且进行了路由结构非流水映射时延的性能分析。通过 FIR、ROBERT 和 FFT8 等计算密集型任务，以 $PEA_{4\times4}$ 的路由型可重构计算系统为例，相比较不考虑容错路由的行放置映射，考虑路由时延的路由映射，在总时延方面平均多消耗了 25.1 %。根据路由型 CGRA 的延时量化分析得出，路由型 CGRA 在数据传输可靠性方面具备优势，但在传输效率方面仍存在挑战，同时表明了可容错路由型互连，在粗粒度可重构计算系统结构设计层面具有合理性和可实践性。

关键词：粗粒度可重构系统，容错路由，硬件流水，流水映射，路由映射

Research on Fault Tolerant Routing and Mapping Methods for Reconfigurable Systems

ABSTRACT

Coarse grained reconfigurable system with configurability, acceleration, low power consumption and other characteristics, has been in the multicore acceleration, Internet of Things systems and other fields to obtain a wider range of applications, the interconnection structure is richer, commonly used mesh-type and routing interconnections, the interconnections of this type of flexibility, the advantages of low-power consumption, fault-tolerant routing reconfigurable systems and mesh reconfigurable arrays of the flow of the mapping has become a domestic and foreign academic research Hot spot, so this paper on this type of problem to carry out research, the main research content is listed as follows:

(1) Aiming at the problem of fault-tolerant algorithms for routed reconfigurable systems, a balanced detour and fault-tolerant routing algorithm is designed, which relies on the built-in self-test to obtain the location information of the faulty region, and by combining the fault

model and the bypass algorithm and using a lower-complexity bypassing strategy, the 8×8 2D Mesh network is used as the experimental simulation platform, compared with the existing fault-tolerant routing algorithms, the experimental results show that The average saturation injection rate of the new algorithm is improved by 20.81% and 20.89%, respectively, and the new algorithm not only improves the load imbalance phenomenon around the fault model, but also effectively reduces the average delay of the packets, and the network can maintain good communication performance.

(2) Aiming at the flow mapping problem of grid-type coarse-grained reconfigurable computing system, a delay quantization method for reconfigurable flow segmentation is proposed, firstly, the processing of hardware instruction flow is divided into seven stages, such as fetch instruction FI, instruction decode ID, compute operand address CO, fetch data FD, hardware configure HC, execute EX, and write back to WB; and then, the inter- and intra-block mapping dependencies of the reconfigurable array are comprehensively considered; Finally, a low delay mapping algorithm is designed on this basis, which uses a combination of routing wiring and operation unit register heap. Based on a set of benchmark programs, $PEA_{3\times3}$ and $PEA_{4\times4}$ are taken to quantize the delay statistics of the layer-waterflow algorithm, in which the layer-waterflow delays of EWF are 74cycles and 79cycles, and the layer-waterflow delays of FDCT are 47cycles and 39cycles, which

indicate the feasibility of the delay quantization method; meanwhile, $PEA_{4\times4}$ is selected for the mapping algorithm's delay comparison, compared to the register-aware application mapping algorithm, the new algorithm obtains the optimization of the total delay performance of the flow water, and the average total delay of the flow water is reduced by 30.1%, thus verifying the effectiveness of the low delay mapping algorithm.

(3) Aiming at the problem of non-pipeline mapping delay consumption of routed coarse-grained reconfigurable computing system, a quantitative analysis method of total delay of routing mapping is given, which comprehensively considers several factors such as computing units, routing data transmission, etc, and performs a performance analysis of the non-pipeline mapping delay of routing structure. With computationally intensive tasks such as FIR, ROBERT and FFT8, the routed reconfigurable computing system of $PEA_{4\times4}$, for example, consumes on average 25.1% more in terms of total delay compared to the row-placement mapping that does not take into account the fault-tolerant routing, and the routing mapping that takes into account the routing delay. Based on the quantitative analysis of the delay of the routed CGRA, it is concluded that the routed CGRA has advantages in terms of data transmission reliability, but there are still challenges in terms of transmission efficiency, and it also demonstrates that the fault-tolerant

routed interconnections are reasonable and practicable at the level of coarse-grained reconfigurable computing system architecture design.

Hu Yuyang(Electronic Information)

Supervised by Prof. Dai GuangZhen,Zhu Biao

KEY WORDS: coarse grained reconfigurable system, fault tolerant routing, hardware pipelining, pipeline mapping, routing mapping

目 录

第 1 章 绪论

1.1 研究背景及意义

随着电子技术的快速发展，可重构计算应运而生。根据可重构处理器中处理单元的数据位宽度，可以分为细粒度和粗粒度，现场可编程门阵列(field programmable gate arrays , FPGA)即为细粒度可重构计算处理器，其具有强大的灵活性和处理性能，但对粗粒度运算的处理效率低下且功耗较大，对此粗粒度可重构系统结构(coarse grained reconfigurable architecture, CGRA)的提出可以有效地解决此类缺陷。CGRA 广泛地应用于多媒体、人工智能以及数据信号处理等方面，其结合了通用处理器(GPP)和专用集成电路(ASIC)的优点，既具有硬件的高效性，又具有软件的灵活性。本课题以二维路由型 CGRA 以及二维网格型 CGRA 为研究对象，对其进行容错路由和映射等相关研究。

本课题研究内容和意义如下所述：片上系统由于外界因素和自身因素的敏感容易导致一些长期故障或短期故障的可能性，为提高片上系统的可靠性，本课题研究路由型 CGRA，基于片上网络研究数据路由的容错策略，降低故障对网络性能的影响，因此对片上网络容错机制的研究具有一定的研究意义。对于编译层面的网格型粗粒度可重构单元阵列的映射研究，本文研究流水映射的流水执行成本，对流水映射在运算节点层面上进行指令级流水时延的量化评估，对流水映射算法的执行成本进行分析，能够有效地帮助优化映射和调度，提高系统性能和效率。

1.2 国内外研究现状

1.2.1 容错路由算法相关研究现状

随着集成技术的发展，片上网络 NoC 作为众核与多核之间的通信结构被提出，其具有可扩展性，可重用性，灵活性等优点。2D Mesh 是一种常用的 NoC 结构[1]，有规则的平面布局等优点，易于设计高效无死锁的路由算法，然而芯片对外界因素和自身因素的敏感容易导致一些长期故障或短期故障的可能性，越来越多的故障会使 NoC 的网络性能逐渐下降，因此对 NoC 容错机制的研究具有一定的研究意义。容错路由是国内外学者常用的容错机制[2]，一个好的容错路由算

法能够很好的缓解路由拥塞，以实现负载均衡，同时还需要尽可能的减少网络延时。路由算法首先需要解决网络死锁问题，解决网络死锁的方法不同可以把容错路由分为有虚通道[4]-[9]和无虚通道路由[10]-[28]，有虚通道路由将物理通道分为几个逻辑通道，通过切换不同的逻辑通道，来解决死锁问题，但是需要额外的硬件消耗来进行容错，而无虚通道路由采用转向模型来避免死锁[3]，使数据包在故障周围进行绕行传输，但是依然存在网络负载不均衡，传输距离过大等缺陷。有虚通道和无虚通道容错路由相关研究阐述如下：

关于有虚通道容错路由的研究：Bansidhar 等人[4]介绍了一种基于人工神经网络的虚拟通道(Virtual Channel, VC)分配技术，其提出的 VC 分配策略相较于 FCFS 的 VC 分配方案，所接收到的数据包数量增加了 5.75%，网络延迟减少了 24.4%。Ebrahimi 等人基于 Y 方向的 VC 分配提出了一种容错路由算法[5](Tolerating both Faulty Links and Routers, TFLR)，其在路由器的 Y 方向使用了虚拟通道，来避免死锁，同时以简单的路由算法保持了网络的性能，只要存在路径，数据包就会通过最短路径进行路由。基于全局拥塞感知（Global Congestion Awareness, GCA）提出了一种的轻量级自适应路由算法[6]，实验表明相较于 RCA 在延迟方面提高了 15%，实际工作负载比 DAR 算法下降 8%。同时 Eltars 等人提出了一种新的流量控制方案[7](Partial Packet Forwarding, PPF)，该方案优化了 VC 重用策略，并提供一个机制把长数据包分割成能够独立路由到目的地的块，结果表明，相比于 WPF 其饱和度平均上升了 23.8%。对一种无虚通道路由算法进行 VC 分配[9]，相较于有 4 个 VC 的 Chalasani 算法，本文所提方法可以将平均延迟降低约 93%。

关于无虚通道容错路由的研究：有学者提出了一种无虚通道绕行容错路由策略[10]，其在故障点周围建立绕行环路。该算法在无故障情况下采用 XY 规则传输数据包，当遇到故障节点时，通过绕行环路传输数据包，从而提高网络性能，该算法复杂度低，绕行规律强，但只能对单故障节点进行路由容错。由此姚磊等人[11]通过结合故障区域与绕行策略提出了一种多故障点的容错路由算法(Fault-Tolerant Routing Algorithm, FTRA)，该算法在垂直方向上由北到南传输，数据包沿 XY 路由方式朝最近的顶点(NW, NE)传输，然后再进行绕行，其网络负载均衡度得到了很大的改善，但是在数据包由南到北传输时，绕过故障区域的传输路径总是先沿 XY 传输到顶点 SW，然后进入绕行，这样传输使网络负载较高

的节点集中在绕行环路的西边界和南边界，部分数据包的传输距离过长，导致网络负载不均衡。李娇等人[12]提出了一种无虚通道容错路由算法，该算法通过添加旁路结构，使故障点与相邻节点连接，继而传输数据包，使路由平均延迟分别降低 4.35%和 20.20%。同时针对三维片上网络将 XY, XZ, YZ 分为奇偶平面，提出一种转向均衡的感知容错路由算法[13]，实验表明，该算法能够有效降低三维网络的延时，并提高了网络可靠性。为了实现三维路由路径的多样性，提出了一种具有路径多样性的流量均衡容错路由算法[14]，实验结果表明了算法具有转向均衡的特点，并且在吞吐量上优于 VMCD 算法。对于不规则的拓扑结构，有一种低开销的无虚通道隔离路由(Low-cost Isolation Routing Algorithm, LIRA)算法[15]，其结合了奇偶(Fault-Tolerant Odd-Even, FTOE)转向模型，通过不规则区域的边界绕行传输数据包，以提高系统利用率。文献[16]基于路径计数器提出了一种容错最小路由方法，可以有效地标记每个无法组成容错最小路径的节点，从而很容易找到一个具有低时间复杂度的容错最小路径。由于容错路由会产生较大的流量负载，Taherkhani 等人提出了一种拥塞感知路由算法，该算法首先将 NoC 分割成多个子网，将局部和全局的思想优势结合起来，使该算法在平均延迟方面得到了显著改进[17]。Guan 等人研究了一种容错及拥塞平衡的路由算法[18]，通过在非故障区域和故障区域的信道计算来选择较低负载区域作为最优路由路径，以平衡流量负载并避免网络拥塞，与相关工作比较，该算法的吞吐量分别提高了 6.92%和 10.7%。通过对故障块周围的虚拟边界绕行，一种无死锁的自适应容错路由算法(F-Route-NoC-Mesh, FRNM)[19]开发了一种基于多故障点的凸形故障模型的方法来平衡网络流量，并提高网络性能，但创建虚拟的故障块边界会大大增加算法的复杂度。Bhanu 等人[20]针对专用片上网络，提出了一种应用映射启发式路由算法和基于路由表的容错路由算法，与其他相关工作相比，所提方法针对特定链路和任意链路故障的实验结果在性能指标上有显著的改进。同时该学者提出了基于 NoC 的容错环面拓扑设计[21]，以针对环面拓扑上的容错应用映射问题，提出了 ILP 和 PSO 技术，基于 FPGA 的实验结果表明，所提方法获得更优越的通信延时。针对 NoC 中链路和路由器故障引起的路由问题，有学者提出了一种基于强化学习的容错路由算法[22](Reinforcement Learning based Fault-Tolerant Routing, RL-FTR)，通过仿真结果表明 RL-FTR 算法在存在故障的情况下始终提供源路由

器和目标路由器之间的最优路由路径。一种网络自适应容错路由算法[23]绕过网络拥塞区域，平衡流量负载，与高效的动态自适应路由算法相比，该算法在功耗更低的情况下，可达到更高的传输效率。一种通用的路由设计方法[24]可以只用几个步骤构建高度灵活的路由算法。根据该方法设计 2D Mesh NoC 的示例容错路由算法，证明了其在片上环境中的可行性。Bhanu 等人[25]提出了一种基于 ZMesh 拓扑的 NoC 容错路由技术。该技术针对链路故障进行容错。通过将该算法与现有的网格拓扑匹配技术进行比较，在链路发生故障的情况下，该算法可以有效的将数据传输到目的地。Fang 等人[26]提出了一种基于区域分区的拥塞感知路由算法，旨在平衡网络负载和降低网络延迟。该算法针对不同的区域选择不同的策略进行路由决策。在最佳情况下，该算法的饱和吞吐量比 XY 算法好 39%，平均好 29%。一种动态容错的路由算法 Track[27]采用基于逻辑的路由，在 8×8 2D Mesh NoC 中，可将单链路和双链路故障的延迟降低 42%，并将吞吐量提高 22%，同时降低了功耗、面积等硬件开销。针对 NoC 中的网络拥塞问题，有学者提出了一种基于模糊逻辑的拥塞控制路由算法[28]，该算法利用拥塞信息，选择一条受益于模糊逻辑不确定性的路径。随着流量的增加，该算法平均时延降低 14.88%，能耗降低 7.98%，吞吐量提高了 14.9%。

1.2.2 可重构计算映射及相关研究现状

粗粒度可重构体系结构 CGRA 具有高能效性、高性价比、灵活性及低功耗等优势，适用于高性能计算和并行计算的各种领域和应用，其能够加速计算任务，提高计算效率，而且能够适应动态变化的计算任务。可重构计算系统的映射质量直接关联着 CGRA 的计算性能，针对映射等问题 Chen 等人[32]提出一种基于行并行可重构体系结构的深度优先贪婪映射(Depth first greedy mapping, DFGM)算法，采用点对点的映射方法进行贪婪映射，以减少 PEA 的数量，相比于层划分映射算法和层贪婪划分映射算法，有更少的配置时间以及行配置块之间的通信成本。同时在文献[33]中提出基于循环子图的贪心映射算法(Loop subgraph level greedy mapping, LSLGM)，与循环子图的层映射算法和满态映射算法相比具有明显的优势。相关典型研究主要表现为如下几个方面：

关于 CGRA 映射研究：Chin 等人提出了一种整数线性规划(Integer Linear programming, ILP)方法[34]用于 CGRA 映射，在一个开源的 CGRA 评估框架中，

映射器接受应用程序和体系结构描述作为输入，如果映射确实可行，则可以生成最优映射，证明了该方法在多种 CGRA 结构上的有效性。文献[35]提出了一种基于整数线性规划的映射技术，在不同的 CGRA 架构上进行了基准评估，所提出的方法与通用 CGRA 映射相比具有明显的改进。一种基于图的映射算法[36]通过利用图遍历技术将数据流输入图映射到 CGRA 架构，并且该方法还在 GPU 上执行的基于图的贪婪启发式并行化。其实验结果表明，与 CGRA-ME 框架下的模拟退火和整数线性规划布局算法进行比较，该方法可以生成最佳映射，并提高了编译速度。Mahdi 等人[37]设计了一种有效的全局启发式方法，称为 EPIMap。首次提出使用重新计算作为资源限制问题的解决方案，可以通过重复计算实现更好的映射。Dave 等人[38]介绍了 RAMP 算法，该算法在调度步骤之前，智能地探索了在利用 CGRA 资源的同时映射数据依赖的各种选择，以提高映射质量。实验表明基于 MiBench 的关键循环 RAMP 比顺序执行加速了 23 倍，比最先进的技术加速了 2.13 倍。对于 CGRA 资源的有效利用，Kou 等人[39]提出了一种面向 CGRA 的传输感知有效循环映射(TAEM)算法，该方法可以有效地利用 CGRA 上的异构资源，加快了编译速度。与 RAMP 技术相比，TAEM 能够将编译时间减少 11.1 倍，同时保持相同或更好的循环映射性能。CGRA 编译器可以将计算密集型循环高效地映射到二维阵列中，但会遇到给定节点的映射失败。于是有一种映射方法 PathSeeker[40]被提出，该方法分析映射失败并对调度进行局部调整以获得映射，其与最先进的算法相比在较低的编译时间内可以实现了更好的映射质量，并且在不同大小的 CGRA 中具有良好的伸缩性，在 4×4 CGRA 上，编译速度比 RAMP 提高了 10 倍，性能提高了 3%。由于 DFG 映射问题的挑战性，Liu 等人[41]基于强化学习(Reinforcement Learning, RL)提出了一种 RLMap 方案，该方案将 DFG 作为 RL 中的智能体在 CGRA 上进行映射表述，可以适应不同的体系结构以及具有高效的映射性能。

关于 CGRA 的资源划分研究：一种改进层划分的算法[42](improved level based partitioning, ILBP)能够根据层划分算法节点分配过程并进行调整，相比较于传统层划分算法，可以充分利用可重构的硬件资源。通过对节点的动态调度，一种划分块数最小化的硬件任务划分(area estimation with multi−objective optimization, AEMO)映射算法[43]有效地减少了可重构系统的配置时间，相较于多目标时域划

分、簇划分、层划分等划分算法，AEMO 具有更小的配置次数。为提高映射效率，文献[44]通过对阵列中未利用的处理单元来贪心映射，基于此提出一种划分式的映射算法，其在划分块数方面获得了较好的改进。同时提出一种行列剪枝映射(row column pruning mapping, RCPM)算法[45]，相较于放置路由算法，其拥有更高的映射效率。针对可重构阵列信息跨层交流的功耗较大问题，一种添加过渡节点的算法[46]减少了运算单元阵列冗余过多的问题，相较于现有的先进算法，所提算法的执行周期分别在 $RCA_{5\times5}$ 和 $RCA_{8\times8}$ 上改进了 23.3%和 30.5%；功耗降低了 15.7%和 18.6%。相比于分裂压缩内核映射算法，一种多目标优化映射算法 MOM[47]平均执行周期降低了 20.6% 和 21.0%。文献[48]提出了三种行并行流水架构阵列，并且提出一种流水映射算法，其在减少块延时上具有可行性。同时评测了三种可重构阵列互连结构[49]，相较于路由型和总线型结构，点到点互连结构的延时最小。

关于 CGRA 的加速技术研究：Kan 等人[51]利用 FPGA 开发了一个基于弹性神经网络的多粒度可重构加速器 (Multi-grained Reconfigurable Accelerator, MGRA)，与传统的 ALU 和最先进的近似计算单元相比，MGRA 大大降低了硬件利用率。一种针对 MIPS 处理器的加速器[52]为了加速整数运算和逻辑运算，其 CGRA 与处理器的主 ALU 连接，实验结果表明，所提出加速器显著的提高了处理器速度。针对可重构架构上映射复杂数据流的问题，文献[53]描述了一个数据分析的可重构架构的工具链，以实现设计空间和探索，并且描述了可重构架构软件堆栈的组件。CGRA 有作为轻型 DNN 加速器的潜力[54]，有学者以 CGRA 作为卷积神经网络(Convolution Neural Network, CNN)加速器的硬件，在 CGRA 能耗降低的情况下，所提出的 CNN 网络性能明显提高。

关于 CGRA 的编译技术研究：Quan 等人提出了一种架构和编译技术[55]，其通过引入动态时间流水线将多个阶段时间复用到同一个 CGRA 上，从而使 CGRA 更具有高效性。同时为了提高 CGRA 编译性能，一个全面的通用编译流程[56]通过开发了一个循环转换模型来存储访问操作以及提供一个冲突感知的空间映射算法来提高片上全局存储器(On-chip Global Memory, OGM)和 PEA 之间的网络带宽利用率。针对加速器执行“if-then-else”构造的分支循环，Mahdi 等人[57]提出一种在 CGRA 上映射带有条件的循环的编译器技术，该技术通过同时向运算单

元 PE 发出两条指令来缓解条件加速问题。同时有学者提出了一种新的具有分支循环结构的解决方案[58]，通过将分支结果通信到 CGRA 的指令获取单元来加速控制流循环，从而克服了与现有技术相关的低效率。Da 等人[59]提出了一种新的 CGRA 架构，该架构实现了在低功耗环境下的完全控制数据流图(Control Data Flow Graph , CDFG)的映射。与完全预测技术相比，所提出的方法克服了现有预测方法的局限性和低效性，提供了 1.6 倍的性能增益。通过实现深度卷积（Depthwise Convolution, DWC）和点卷积（Pointwise Convolution, PWC）内核的高性能映射，提出了 DWC 映射的跨通道优化，以进一步提高 DWC 中的 PE 利用率。同时改进一种专业化 CGRA 架构，能够为计算内核提供更高的性能[60]。实验结果表明，所改进的 CGRA 可以在面积延迟方面提供了 8~18 倍的改进。Lu 等人[61]提出了一种空间计算体系结构中的分支预测器，可以根据控制流的特点配置成不同的预测模式。实验结果表明，该分支预测器在粗粒度可重构阵列平台上的预测精度优于所有传统的预测器。以微小面积增量为代价，与现有技术相比，预测精度提高了 5.09%。Mahdi 等人[62]根据运算单元自带的寄存器，提出了一种寄存器感知应用映射算法(Register-Aware Application Mapping, REGIMap)，同时从循环映射问题的通用公式中提炼高效的启发式方案，极大地改进了 CGRA 的编译调度技术。然而使用分支编译循环是静态调度 CGRA 的一个挑战，为了有效地提取分支特征并选择合适的方法，Yuan 等人[63]提出了一种混合编译器框架，并对 CGRA 架构进行了轻量级修改，其中包括了性能评估模型、DFG 和映射的优化平衡，以此减少了编译时间和功耗，提高了 CGRA 的性能。同时以 II 作为主要标准提出了一种动态流水算法[64] (Dynamic-II Pipeline, DIP)，以解决分支编译技术处理不规则分支时的性能问题，并且根据所提的混合编译框架选择和实现相应的分支处理方法，与部分预测和完全预测相比较，DIP 的总执行时间平均减少了 27.21 %和 22.04%。利用分解映射策略可以将 CGRA 模调度（MS）映射分解为时间映射和空间映射。Zhao 等人主要强调开发了系统的时间映射流[65]，以提高映射性能，并结合开发的一种快速和高效的空间映射算法，能够在一定的编译时间预算内生成较高的映射成功率。故数据流图的划分是高效率映射必不可少的方法，有国际学者在中提出了基于遗传算法的新划分方法[66]，以消除无法映射的 DFG，从而提高映射质量。

关于 CGRA 性能和功耗优化研究：文献[67]提出了一种将 Dual-Vdd 分配集成到循环流水线映射中的低功耗优化方法。实验结果表明，该优化方法在不降低性能的前提下，在最小化功耗方面是有效的。基于 CGRA 的故障问题，Ganghee L 等人[68]提出了一种 CGRA 瞬时故障恢复时间模型，该容错 CGRA 基于三模冗余和编码技术进行错误检测和纠错。研究结果表明，任务分区对于限制执行时间较长的应用程序恢复非常重要，并且纠错码对于高辐射环境中执行时间较长的任务或任务分区程度较高的任务的实用价值有限。同时针对老化问题给系统可靠性带来的威胁，有学者提出了一种启发式的应力感知循环映射算法[69]，其核内应力优化策略为流水线技术，核间应力优化则开发了一种多映射调度方法，使 PE 的使用多样化。实验表明，该方法可以有效地缓解 CGRA 上的老化问题，同时保持优良的性能。为了缓解 CGRA 上的老化，一个利用感知的分配策略[70]通过引入对重新配置逻辑和数据路径的重要扩展，以平衡单个 FU 的应力恢复率，从而降低整个设计的老化率。在最先进的 CGRA 中实施该方法，其寿命提高了 2.2 倍，且面积增加不到 10%。

1.3 论文的主要工作

本工作针对粗粒度可重构计算体系结构进行通信路由研究、流水映射及路由映射的延时量化研究。具体如下所述：

(1) 为了片上系统的可靠性，研究路由型可重构阵列的容错路由算法，分析数据包的传输原理，通过结合故障模型和绕行算法，保证了网络通信稳定的同时，进行多故障路由器的容错路由算法设计，并且使用 Booksim2.0 仿真器对网络流量负载、饱和注入率等参数进行分析。

(2) 研究网格型可重构阵列的流水映射时延，根据硬件指令级流水对软件流水的时延进行分析评估，将流水映射的执行成本从块间依赖映射和块内依赖映射两个维度进行综合，从而提出一种基于流水映射的时延量化评估方法，并根据该评估时延提出了一种映射方法，通过一组基准程序集，获得流水执行时延的量化统计，验证了映射方法的有效性。

(3) 针对路由型可重构阵列，通过分析计算单元、路由传输等因素，给出了路由映射总时延的评估方法，并且基于一组基准程序集进行量化测试，表明了基于路由型 CGRA 映射的延时量化评估方法是可行的。

1.4 论文的组织结构

本文分为 6 个章节来阐述，具体的论文章节安排如下：

第 1 章，绪论。首先对可重构计算系统的进行研究背景和意义的阐述，再叙述本文的主要工作，最后介绍本论文的组织结构。

第 2 章，可重构相关知识及国内外现状。通过网格型 CGRA 和路由型 CGRA，描述相关研究知识，然后国内外现状围绕片上容错路由算法和可重构计算映射两个方向展开叙述。

第 3 章，基于结合 NoC 的路由型 CGRA，对于多故障路由器进行容错路由算法的设计，分析数据包的传输原理，通过结合故障模型和绕行算法，保证了网络通信稳定的同时，提高了网络性能。并通过 Booksim2.0 仿真器，验证了该算法的有效性。

第 4 章，针对网格型 CGRA，提出了一种流水执行成本的量化方法，通过引入硬件指令级流水对软件流水映射的时延进行量化评估，且通过该评估时延提出一种低延时的映射方法，通过一组基准程序验证了量化方法的可行性及映射方法的有效性。

第 5 章，针对路由型 CGRA，给出了一种路由映射的时延评估方法。根据量化结果表明了基于路由型 CGRA 映射的延时量化评估方法在实践中具有可行性。

第 6 章，总结和展望。回顾已经完成的工作并做出总结和评价，同时对未来的研究趋势进行评估和规划。

第 2 章 可重构计算平台相关研究知识

随着集成技术的进步，人们不断寻求高性能、低功耗和低成本的集成微系统。在这个求知过程中，软件与硬件的融合成为关键因素，因此可重构计算技术应运而生，并在各个领域中占有重要地位。可重构体系结构内部拥有可以被重构的互联网络，即可重构处理单元阵列(Processing Element Array, PEA)，对于可重构计算简单的来说就是用软件去设计硬件，即一个 PEA 根据软件的需求，调整其内部的互联网络，改变成与其相互适应的硬件平台，并实现其功能。粗粒度可重构系统结构 CGRA 具有配置、流处理等特征，同时兼并同构单元阵列的运算性质以及异构单元阵列交叉处理数据的能力。本章节首先介绍了可重构计算的硬件体系架构，在此基础上讨论了网格型 CGRA 和路由型 CGRA 的两个阵列结构及相关研究知识，然后从容错路由和可重构系统映射两个研究方向上来阐述国内外学者的研究现状。

2.1 可重构体系架构简介

可重构计算根据应用的不同对可重构硬件计算平台进行配置，完成相应的计算任务。一个可重构硬件体系架构的主要构成为主存储器(Main Memory, MM)、主处理器(Main Processor, MP)、直接存储器访问(Direct Memory Access, DMA)、可重构处理单元(Reconfigurable Processing Unit, RPU)等结构，如图 2-1 所示。RPU 通常由可重构处理单元阵列 PEA 和可重构阵列配置控制器(Reconfigurable Array Configuring Controller, RACC)等组成，其中 PEA 负责数据流的并行处理，RACC 执行控制流来配置 PEA 的组织管理工作及任务映射的调度。PEA 可以按照需求进行重构，以适应不同的计算任务及数据流，其一般有存储器、配置接口、数据接口等模块，配置接口负责从 RACC 获取配置字，继而配置 PEA 的功能，调度阵列上的任务执行顺序，而数据接口完成对外部数据的访存和写回从而获得数据流。RACC 包括配置接口、配置存储模块、配置控制器模块，配置接口负责向 PEA 发送配置字，配置控制器负责把外部的配置信息转换成内部的配置字及控制信号。不同的可重构计算单元阵列具有各自独特的优势，本文将着重研究二维网格型计算单元阵列和二维路由型计算单元阵列。在研究过程中，将探讨流水映

射和容错路由等相关技术研究，以深入了解其原理和应用。

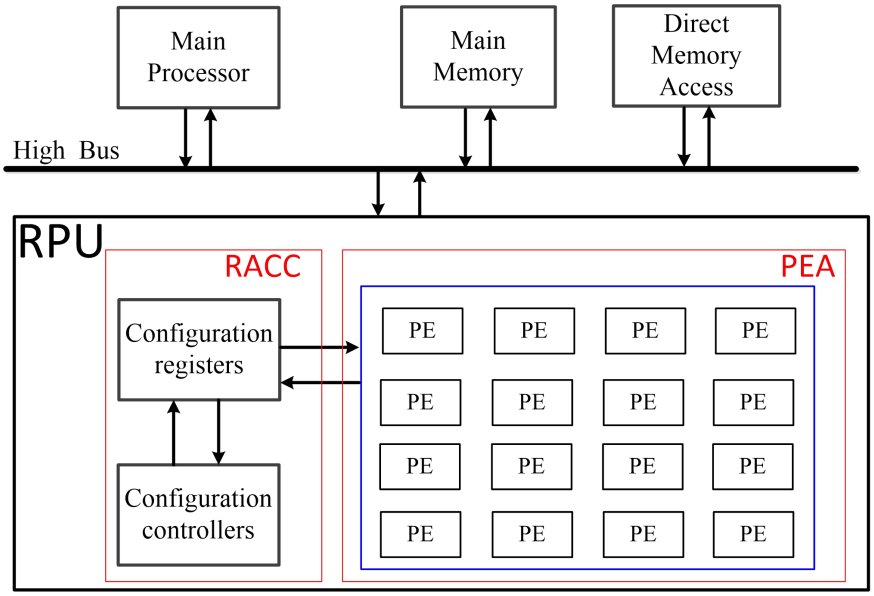


图 2-1　CGRA 体系结构

2.2 可重构计算单元阵列流水概述

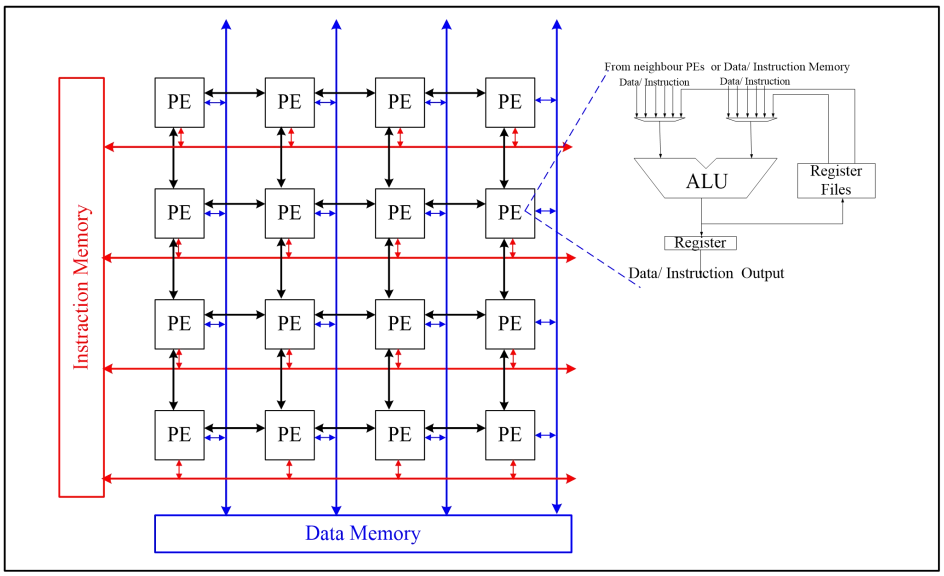


图 2-2　网格型 CGRA 互连结构

网格型 CGRA 其计算单元布置在一个规则的二维网格上，PEA 由若干个同构或异构的处理单元(Processing Element, PE)通过水平和垂直的总线相互连接，以实现计算单元之间的通信。其中 PE 一般由算数逻辑单元(Arithmetic and Logic Unit, ALU)和寄存器堆构成，可以执行并行的算术和逻辑运算，用于数据处理和计算任务，具体如图 2-2 所示。网格型结构具有扩展性、灵活性、数据并行性等

优势，其扩展性可以在水平和垂直方向扩展计算单元和互联资源，以应对更复杂的计算任务；灵活性可以实现对数据通路的灵活配置，以适应不同的计算任务需求；数据并行性体现在 CGRA 同时执行多个计算任务，实现高度的数据并行计算。网格型结构通常具有紧凑的布局和资源共享，可以减少功耗并提高能效，因此也具有低功耗的特性。

2.2.1 硬件流水

一条完整的指令可以分成以下几个步骤，具体为：取指令(Fetch Instruction, FI)、指令译码(Instruction Decode, ID)、计算操作数地址(Computing Operand_address, CO)、取数据(Fetch Data, FD)、硬件配置(Hardware Configuration, HC)、执 行(Execute, EX)、写回(Write Back, WB)，具体分为 7 级流水线。FI、ID、CO、FD、HC、EX、WB分别在A、B、C、D、E、F、G七个不同的硬件上执行，假设每个机器周期包含 4 个时钟周期(cycle)，在定长的机器周期情况下，在没有进行指令拆分和配备独立硬件资源之前，运行一条完整的指令大致需要 7 个机器周期，7 条指令需要的时钟周期数为 49cycles，经过指令拆分和配备独立硬件资源之后，7 条指令在没有断流理想的情况下需要的时钟周期数为 13cycles，其加速比为 73%。功能单元 (Function Unit, FU)可扩展为 7 个能够独立运行的子部件，具体如下图所示：

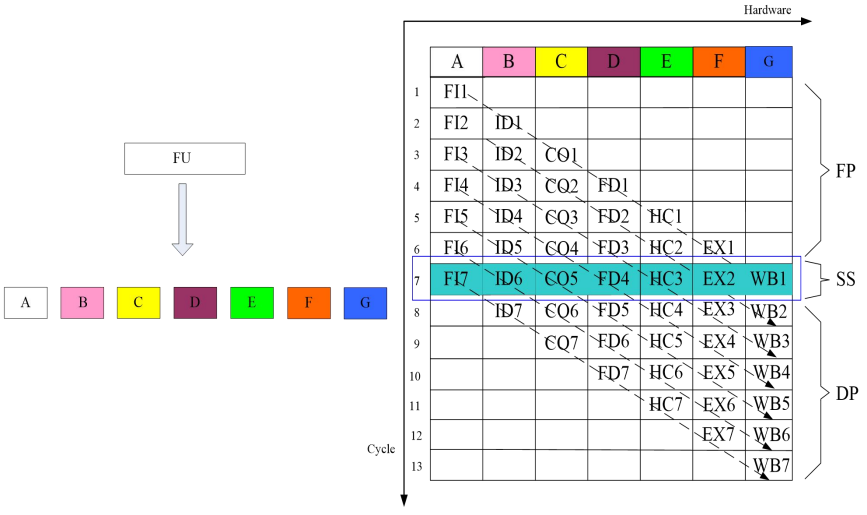


图 2-3　硬件流水

指令的流水执行如图 2-3，在第七个周期上，取指FI7 在A上执行，该周期上

同时有第六条指令的译码ID6，第五条指令的计算操作数地址CO5，第四条指令的取数FD4，第三条指令的硬件配置HC3，第二条指令的执行EX2 以及第一条指令的写回WB1，在其它独立的硬件B、C、D、E、F、G上执行，这就是一次流水过程，如上图框出的时间段，相比前面的六个周期和后面的六个周期，执行效率极大提升，即该周期也叫流水线稳定阶段(Steady State, SS)；前六个周期的硬件资源利用率是逐步增加的，称为流水线装载阶段(Fill Pipeline, FP)；后面六个周期，其硬件资源的利用率逐步下降，也称为流水线排空阶段(Drain Pipeline, DP)。

2.2.2 软件流水

CGRA 软件流水是在粗粒度可重构体系架构中，优化计算性能和吞吐量的一种技术。该技术通过将多个指令或计算任务重叠执行，以在一个时钟周期内实现更高的指令级并行性。具体为一长串独立的算子在硬件资源上进行循环级的并行流水，其中硬件资源为可以运算完 DFG 算子的最少资源组合。与硬件流水一致，流水形成的稳定阶段称为核心，因其中包含的算子恰好能够等效为一个完整的循环。

2.3 路由型可重构计算单元阵列

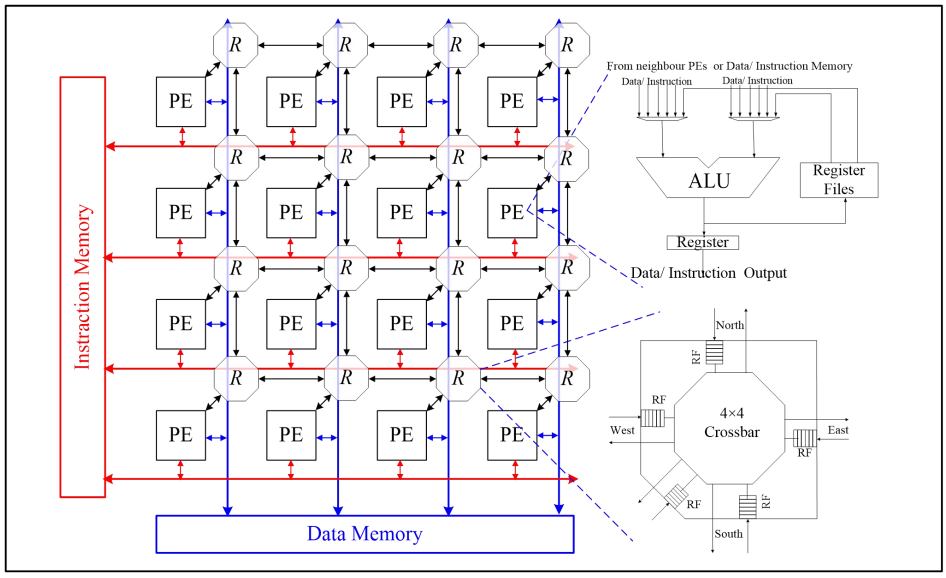


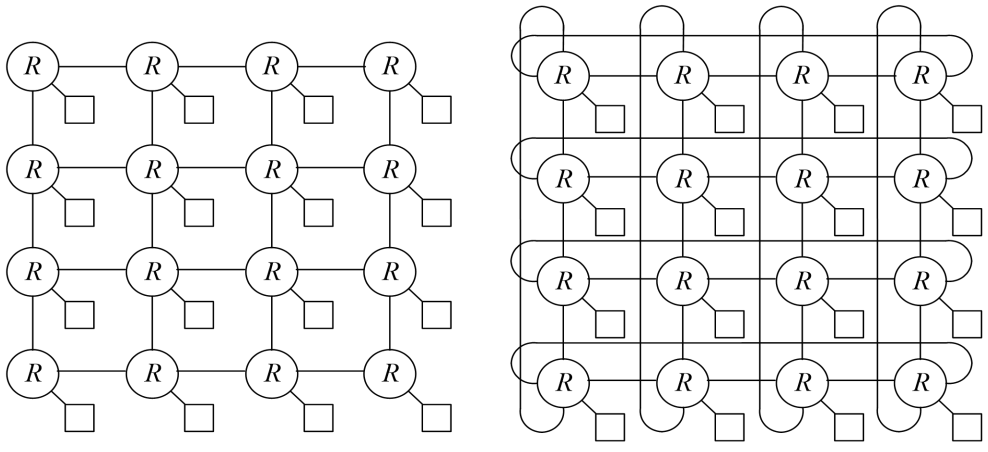
图 2-4　路由型 CGRA 互连结构[50]

片上网络(Network on Chip, NoC)作为一种多核间的通信结构，将其与 CGRA 相结合可以组成路由型 CGRA。相对于网格型 CGRA，其着重解决数据传输和通信稳定等问题，通过配置路由器，实现高效的数据传输，即数据传输的容错性能

增加，通信可靠性增强。其中 PE 的组成与网格型相同，但 PE 与 PE 之间的互联方式不同，增加了路由器(Router)进行数据的通信传输。图 2-4 为一个 4×4 的 PEA 阵列，数据的传输首先由 PE 进行计算，然后传输到路由器中，进行路由传输到目的 PE 相对应的路由节点。路由器有五个端口，其中一个连着本地 PE，其他四个连接相邻的路由节点，寄存器文件堆(Register Files, RF)将待转发的数据包进行暂时性的存储，而交叉开关矩阵连接着不同的端口，通过路由协议的控制，选择输入缓存中的数据，并将其发送到指定的目标端口。当然，片上网络的设计需要考虑拓扑结构、流控制机制，并且基于拓扑结构进行路由算法设计。

2.3.1 拓扑结构

片上网络作为多核和众核间的一种通信网络，其中网络拓扑确定了网络中路由器与链路之间的物理布局及互联方式。拓扑结构可以决定一条消息经过的路由跳数及路由线路，其对网络整体的质量有着重要的影响，且路由及映射算法的设计也依赖于拓扑结构。片上网络中常用的拓扑结构有 Mesh、Torus 等结构。如图 2-5 所示，2D Mesh 是最直观、最常用的一种结构，节点与链路组成了规则的二维网格结构，每个节点与周围的节点相连接，具有规则的平面布局等优点，易于设计高效无死锁的路由算法。2D Torus 结构针对 2D Mesh 的边界节点增加了环形链路，其可以解决边界路由的通信问题，且降低了网络直径，但是该结构具有环形链路会导致网络中链路长度不一，会使流控制变得复杂，同时会加大了路由算法设计的复杂度，也会给处理器的布局布线带来难度。从实际的角度出发，大多数 NoC 使用 2D Mesh 拓扑，以方便板上的拓扑布局及电路映射。



(a) Mesh　　(b) Torus

图 2-5　拓扑结构

2.3.2 流控制

流控制是指在消息通过网络时如何分配资源，即负责为等待的数据包分配相应的缓冲区和链路资源。良好的流控制机制，可以降低信息传输延时，并通过数据对缓冲区和链路的有效共享来提高网络吞吐量。流控制机制包括存储转发流控、虫孔流控、虚通道流控等，下面重点介绍存储转发和虫孔流控。

当一条消息注入网络时会被分段成数据包，其中数据包是网络传输最小的通信单元，微片(filt)是划分数据包的最小单位。一个数据包由包含源、目的节点地址的头微片(head filt)、包含数据主体的若干个体微片(body filt)以及表示数据包传输完成的尾微片(tail filt)组成。

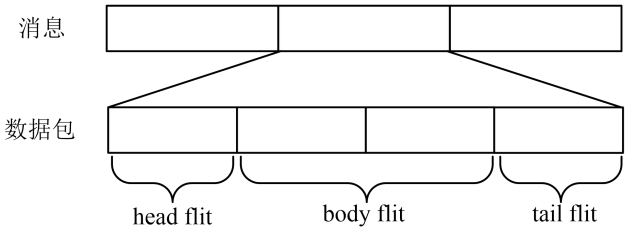


图 2-6　数据包格式

(1)存储转发流控

存储转发流控是基于数据包的流控技术，具体为每个路由节点在完全接收整个数据包之后才会向下一个节点转发，这会导致较高的延迟以及大量的存储空间。如图 2-7 示例：一个数据包从节点 0 传输到目的节点 9，其经过节点 0、1、3、5、9，对每个路由节点来说，一旦接受到 tail flit 即表示该数据包已被接受并且缓存，此时就可以根据 head flit 进行分配链路，将数据包发向下一个节点。H 为 head flit，B 为 body flit，T 为 tail flit，每个节点都必须等到 H、B、T 都到达后才会进行下一跳的传输，因此会带来串行化延时，对一个由 4 flit 组成的数据包，其每一跳的传输延时为 4 个周期。

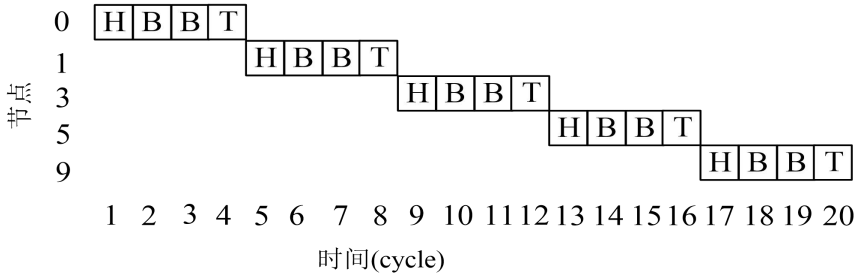


图 2-7　存储转发流控示例

(2)虫孔流控

虫孔流控是基于微片的流控技术，其降低了对存储空间的需求，可以有效地降低数据包的传输延时，具体如图 2-8。在虫孔流控中，下游节点如有 flit 大小的空闲缓冲区资源，则当前节点在接受到整个数据包之前就允许将部分 flit 向下游节点进行发送，其有效地使用了缓冲区资源，但无法有效地利用链路带宽，即在数据包传输期间一旦遇见数据包阻塞的情况时，数据包所占用的物理链路都将处于闲置状态无法继续传输。因为虫孔流控在 flit 粒度上面分配缓冲区资源，由多个 flit 组成的数据包可能会跨越多个路由器，一旦发生阻塞，很多物理链路都将进入闲置状态。如图示例所示：当 head flit 在节点 1 传输到节点 3 的过程中遇到阻塞，由于节点 1 缓冲区被占满，剩余的微片会在节点 0 处被阻塞，图中灰色区域为该数据包所占用的闲置通道，当该数据包没有完全传输完成时，无法给其他数据包进行传输。

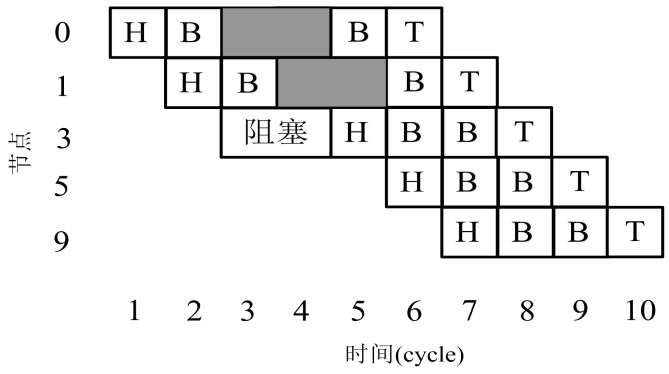


图 2-8　虫孔流控示例

2.3.3 路由

路由算法直接影响了片上网络的性能，其运用不同的策略确定了数据包从起始节点到目的节点之间的路由路径。路由算法的设计依赖于拓扑结构，一个好的路由算法，应尽可能的限制算法复杂度使网络流量均匀分配，从而提高网络性能，同时也需要保证网络没有死锁的情况。

确定性路由为源节点向目标节点移动的数据包只通过唯一确定的路径传输，XY 路由就是常用的确定性路由，首先使数据包在 X 维度上路由到目的节点的所在列，然后继续在 Y 维度上传输到目的节点，由于该类算法只与源节点和目的节点的位置有关，未考虑网络状态的信息，所以在面对网络拥塞及路由器故障的

情况，会导致网络性能的降低。自适应路由(Aadaptive Routing, AR)可以动态地根据当前网络流量情况，选择源节点和目的节点之间的路径进行传输，可以有效的响应网络的拥塞状况，也能够利用路径多样性提供负载平衡和故障容错，但是该类算法复杂度高，且具有较大的面积和功耗开销。

随着片上网络路由算法的研究，容错路由的研究具有一定的意义，对于瞬时故障和永久故障等原因造成的网络连接失效等情况，容错路由算法的研究可以确保网络的可靠性，提高网络性能。

2.4 本章小结

本章节首先从可重构计算系统的体系架构进行介绍，对可重构计算单元阵列进行分类描述；然后针对网格型可重构计算单元阵列介绍了硬件流水和软件流水的相关研究背景；最后基于路由型可重构计算单元阵列介绍了片上网络的拓扑结构、流控制、路由等相关研究知识。研究表明粗粒度可重构流水映射较为成熟，本文对流水映射的时延成本展开研究。路由型 CGRA 的容错通信研究大多数学者从无虚通道和有虚通道两个方向上进行算法设计，由于有虚通道技术需要额外的硬件消耗，对此本文基于无虚通道展开容错路由的通信研究。

第 3 章 可重构系统多故障节点容错路由算法

3.1 引言

NoC 作为一种多核间的通信结构被提出，其具有可扩展性，可重用性，灵活性等优点，然而芯片对外界因素和自身因素的敏感容易导致一些长期故障或短期故障的可能性，越来越多的故障使 NoC 的网络性能逐渐下降，因此对 NoC 容错机制的研究具有一定的研究意义，对此本章节提出了一种多故障点的无虚通道容错路由算法。

3.2 问题定义

定义 1 故障模型：一个合适的故障模型在容错路由算法的设计中，对算法设计的复杂度及有效性有决定性作用。本文采用矩形故障模型，矩形的凸性有利于使用较少虚通道或无虚通道简单、有效的特点，在故障区域周围路由信息，以避免死锁。故障模型由相连的节点和链路构成，其中节点分为故障节点、安全节点、不安全节点，具体定义如下：

安全节点：所有无故障的节点均初始化为安全节点。

不安全节点：(1)如果一个安全节点的直接邻居有两个及以上的故障节点或不安全节点，则将该节点声明为不安全节点。(2)如果一个安全节点在 *X*(*Y*)维度上的直接邻居有一个故障节点或不安全节点，并且在 *Y*(*X*)维度上距离为 2 的邻居是故障节点或不安全节点，则该节点被声明是不安全节点。

故障块：将故障节点和不安全节点进行划分的区域。在故障块中，除了包含故障节点之外，还包含不安全节点，以保证故障块为矩形区域。故障块的划分可以描述为不安全点被声明的过程。

定义 2 绕行环路：故障区域周围的安全节点及链路构成的环路被称为绕行路径。故障区域位于网络内部时，绕行环路有四个顶点东北(NE)、西北(NW)、东南(SE)、西南(SW)。绕行环路上的节点称为边界节点。*C*、*S* 和 *D* 分别表示当前节点、源节点、目的节点。

定义 3 节点转向模型：本文规定数据在节点的传输方向为 *NE*(由北向东)、

WS(由西向南)、*ES*(由东向南)、*NW*(由北向西)、*SE*(由南向东)、*WN*(由西向北)、*EN*(由东向北)、*SW*(由南向西)，具体如图 3-1 所示。

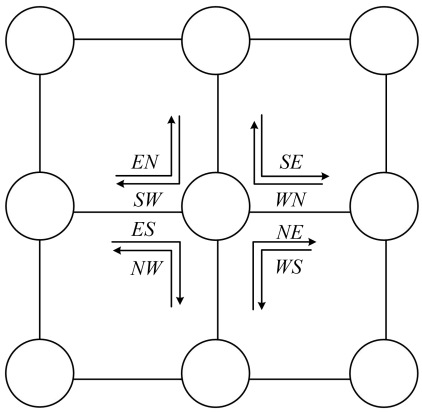


图 3-1　节点转向模型示意图

定义 4 网络坐标系建立：对 2D-Mesh 网络进行坐标系建立如图 3-2 所示，*X*、*Y* 表示其横纵坐标，*X* 正方向为东，负方向为西，*Y* 正方向为北，负方向为南。X_S、Y_S，X_D、Y_D 分别表示源节点和目标节点的横纵坐标。各顶点的横纵坐标表示为 X_{NE}、Y_{NE}，X_{NW}、Y_{NW}。

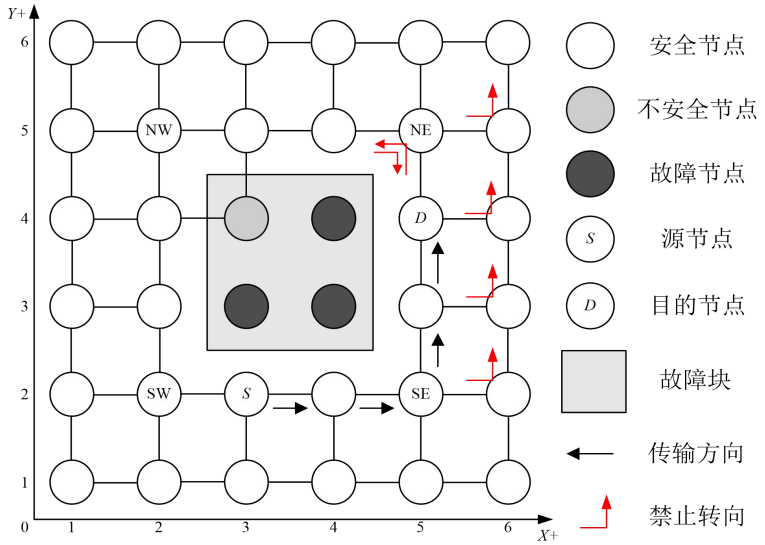


图 3-2　路由节点坐标及相关说明

定义 5 死锁：死锁可以定义为资源无限循环等待的一个依赖环，每个数据包都占有一部分资源，并且需要等待获取环中其他数据包所占有的资源，从而导致整个依赖环中数据包无法传输。

定义 6 饱和注入率：注入率表示网络在单位时间内可处理的数据包个数，饱和注入率η即为网络时延在零负载时延两倍情况下所对应的注入率；N_{total_node}

表示网络中节点总数；T_{total_cycle} 表示数据包经过通信网络的时间；LEN_i 表示在 T_{total_cycle} 时钟周期内到达目的节点的第 i 个数据包的长度；n 表示所有到达目的节点的数据包个数，具体见公式 3-1。

$$\eta = \frac{\sum_{i=1}^{n} LEN_i}{N_{total\_node} \times T_{total\_cycle}} \tag{3-1}$$

3.3 实验动机和容错路由算法设计

3.3.1 实验动机

容错路由是国内外学者常用的容错机制，一个好的容错路由算法能够很好的缓解可能出现的路由拥塞，以实现负载均衡，同时还需要尽可能的减少网络延时。容错路由算法可以分为有虚通道和无虚通道路由，有虚通道路由需要额外的硬件消耗来进行容错，而无虚通道路由采用转向模型来避免死锁，使数据包在故障周围进行绕行传输，但是依然存在网络负载不均衡，传输距离过大等缺陷，对此解决无虚通道路由的缺陷等问题成为了研究者们的一项挑战。本章节设计了一种均衡的绕行容错路由(Balanced Detour and Fault-Tolerant routing, BDFT)算法，其均衡了故障区域周围的网络负载，且降低了网络时延。

3.3.2 容错路由算法设计

对多故障路由进行容错机制的设计，一个合适的故障模型很重要，其主要作用是对容错路由算法的复杂度进行了简化。然而故障绕行策略则是容错路由算法中的关键部分，其直接决定了算法的容错能力。本文依赖于内建自测技术(Built-in Self-Test, BIST)[29]将故障信息向 $X=X_{SW}$，$X=X_{SE}$，$X=X_{SE}+1$ 列的所有节点传输，并存储在故障存储器内。本算法把矩形故障模型与容错绕行相结合，使数据包进行了均衡的绕行路由，并缩短绕行距离，从而均衡故障区域周围的链路负载以及有效的降低了网络通信时延。该算法设计分为三个步骤：(一)数据包路由时遇不到故障；(二)数据包路由时从从 X 方向进入绕行环路；(三)数据包路由时从从 Y 方向进入绕行环路。根据上述三个步骤细分为五种策略对多故障容错路由算法进行设计。具体情况策略如下所示：

(一)数据包正常路由

策略一：数据包正常路由即数据包在传输的过程中遇不到故障的情况。在本

章所提算法中，数据包传输遇不到故障时，按 XY 路由算法传输。但有一种特殊情况，如果 D 位于 $X_D=X_{SE}+1$ 列并且 $Y_D \geqslant Y_{SE}$，若 S 位于 D 的西南方向，则数据包先到达 $X=X_{SE}$ 列，再向北传输到 D 的所在行，最后向 D 传输。如果数据包遇到故障，算法分别对数据包在 X 方向遇到故障和在 Y 方向遇到故障的绕行进行了优化。

(二)数据包在 X 维度的绕行优化方法

数据包从 X 方向进入绕行环路，以 S 和 D 的相对位置来表示两种情况，从东侧进入绕行环路和从西侧进入绕行环路。

情况一：S 在故障区域所在行东侧($X_S \geqslant X_{SE}, Y_{SE} < Y_S < Y_{NE}$)，$D$ 位于故障区域西侧或故障区域所在列($X_D \leqslant X_{SW}$ 或 $X_{SW} < X_D < X_{SE}$)。

情况二：S 在故障区域所在行西侧($X_S \leqslant X_{SW}, Y_{SW} < Y_S < Y_{NW}$)，$D$ 位于故障区域东侧或故障区域所在列($X_D \geqslant X_{SE}$ 或 $X_{SW} < X_D < X_{SE}$)。

在第一种情况下，按 FTRA 算法，数据包沿 X 方向进入绕行环路。如果 D 在故障区域所在列南侧或在故障区域西侧，则数据包在经过第一个顶点(SE)后按 XY 路由方法向 D 传输。如果 D 在故障区域所在列北侧，则数据包经绕行环路绕行到顶点(NW)然后按 XY 路由方法向 D 传输。这种路由方式使得部分数据包传输的绕行距离过大，增大了绕行环路上的拥塞程度。对此本文算法允许了故障区域东侧 $X=X_{SE}$ 列部分节点($Y \leqslant Y_{SE}$)的 SE 转向，故障区域东侧 $X=X_{SE}+1$ 列部分节点($Y \geqslant Y_{NE}$)的 NW 转向，禁止了故障区域东侧 $X=X_{SE}+1$ 列部分节点($Y \geqslant Y_{SE}$)的 EN 转向。这样可以在保证网络无死锁的情况下，均衡网络负载，缩短部分数据包向目的节点的绕行距离。具体策略如下：

策略二：对于 S 位于绕行环路东侧($X_S \geqslant X_{SE}+1, Y_{SE} < Y_S < Y_{NE}$)，D 位于故障区域北边所在列或位于故障区域的西北方向($X_{NW} < X_D < X_{NE}, Y_D \geqslant Y_{NW}$ 或 $X_D \leqslant X_{NW}, Y_D \geqslant Y_{NW}$)，数据包沿 X 方向传输到 $X=X_{SE}+1$ 列，然后按 YX 路由方式向 D 传输，如图 3-3 中 S_1 到 D_1。算法其余绕行情况，数据包进入绕行环路经过第一个顶点后，按 XY 路由方式向 D 传输。

策略三：对于第二种情况，如果 D 位于 $X_D=X_{SE}+1$ 列并且 $Y_D \geqslant Y_{SE}$，则数据包经过绕行环路到达 $X=X_{SE}$ 列，再传输到 D 的所在行，向 D 传输。这是因为本文为了避免死锁，禁止了故障区域东侧 $X=X_{SE}+1$ 列部分节点($Y \geqslant Y_{SE}$)的 EN 转向。

算法其余绕行情况，数据包进入绕行环路经过第一个顶点后，按 XY 路由方式向 *D* 传输。

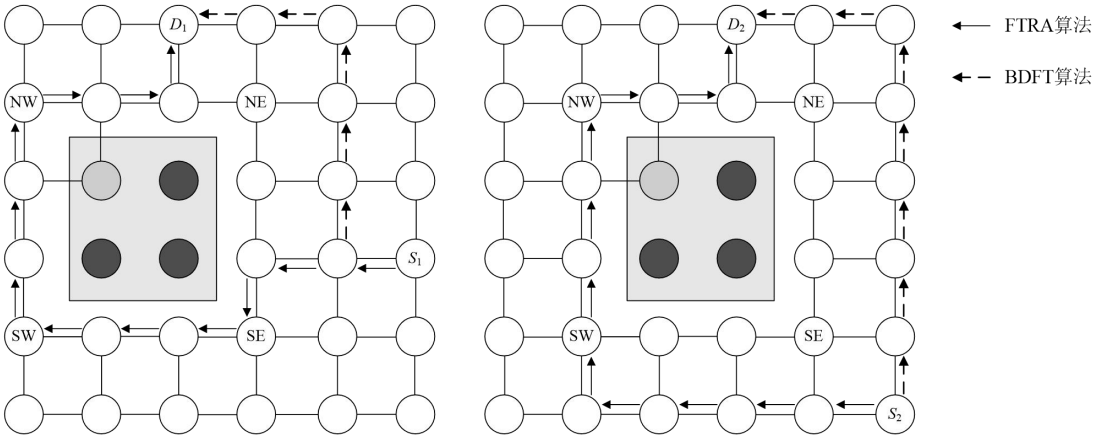


图 3-3　部分数据传输路径对比

(三)数据包在 Y 维度的绕行优化方法

数据包从 Y 方向进入绕行环路，以 S 和 D 的相对位置来表示。

情况三：*S* 位于故障区域南侧($Y_S \leqslant Y_{SW}$)，*D* 位于故障区域北侧且位于故障区域所在列($Y_D \geqslant Y_{NW}, X_{NW} < X_D < X_{NE}$)。

情况四：*S* 位于故障区域北侧($Y_S > Y_{NW}$)，*D* 位于故障区域南侧且位于故障区域所在列($Y_D \leqslant Y_{SW}, X_{SW} < X_D < X_{SE}$)。

对于第三种情况，按 FTRA 算法，数据包先按 XY 的路由方式传输到顶点 SW，继而再从绕行环路的西边界到达顶点 NW，然后按 XY 路由方式传输到 *D*，如图 3-3 中 S_2 到 D_2。对于该种情况数据包的传输均从绕行环路的西边界绕行，增大了西边界的拥塞程度。对此本文算法策略如下：

策略四：对于 *S* 位于故障区域东南方向($X_S \geqslant X_{SE}, Y_S \leqslant Y_{SE}$)，除了绕行环路顶点 SE 外，数据包沿 *X* 方向传输到 X_{SE}+1 列，继而按 YX 路由方式向 *D* 传输，如图 3-3 中 S_2 到 D_2 所示。其余情况与 FTRA 算法一致。

策略五：对于第四种情况，本文算法与 FTRA 算法一致，数据包先按 XY 路由方式传输到最近的顶点，然后进入绕行环路向 *D* 传输。

算法流程图描述见图 3-4:

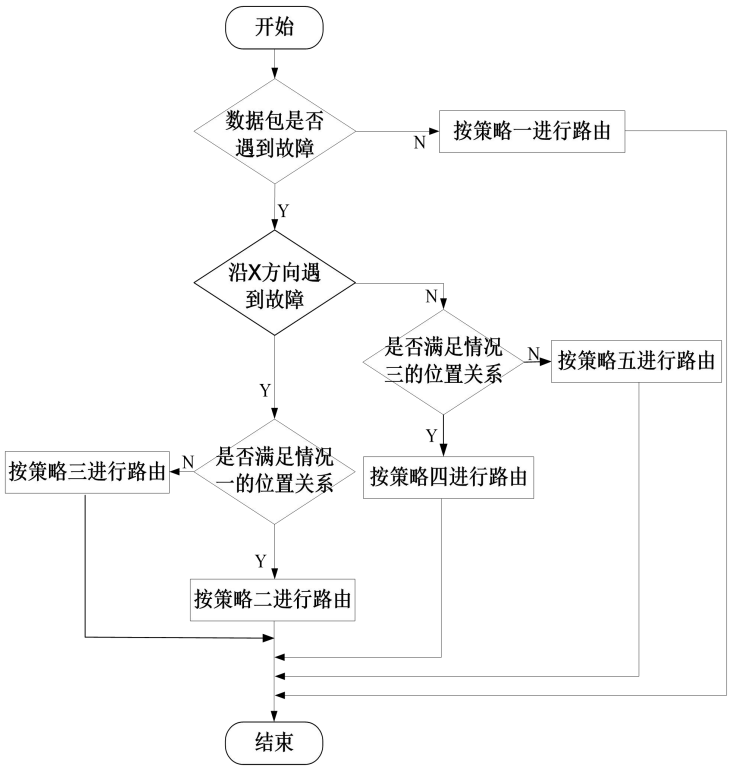


图 3-4　BDFT 流程图

3.3.3 算法无死锁证明

对 BDFT 算法进行无死锁证明，采用了 Dally 等提出的通道依赖图(Channel Dependency Graph, CDG)的方法[30]。本文所提算法为无虚通道路由算法，即只需证明算法在网络中的 CDG 无环路，则该算法是无死锁的。以 5×5 的 2D Mesh 网络为例，假设故障点为 n_7、n_8、n_{12}、n_{13}，其网络连接图如图 3-5 所示。

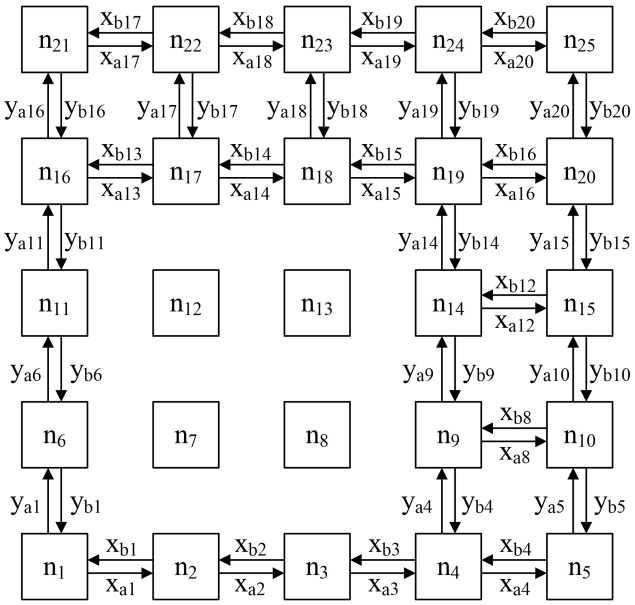


图 3-5　网络连接图

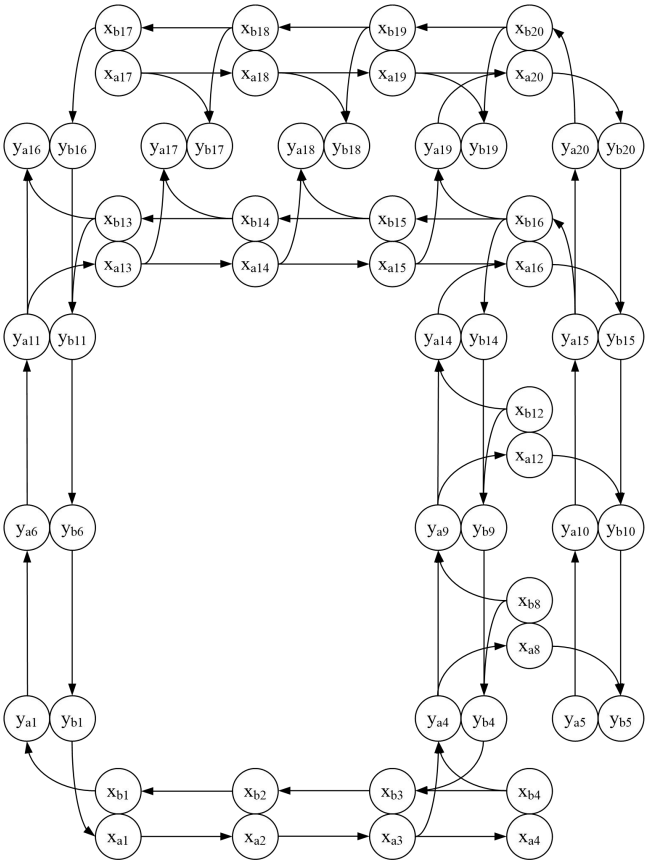


图 3-6　通道依赖图

对于所提算法，数据包传输遇不到故障时，按 XY 路由算法传输，则禁止了数据由 *Y* 到 *X* 的转向。为了数据包更好的绕行，除了在绕行环路上增加 y_{b4} 到 x_{b3}，y_{b1} 到 x_{a1}，y_{a11} 到 x_{a13} 的转向，而且增加了转向 y_{a15} 到 x_{b16}，y_{a20} 到 x_{b20}，y_{a4} 到 x_{a8}，y_{a9} 到 x_{a12}，y_{a14} 到 x_{a16}，y_{a19} 到 x_{a20}。为了不出现环路同时禁止了转向 x_{a15} 到 y_{b14}，x_{a4} 到 y_{a5}，x_{a8} 到 y_{a10}，x_{a12} 到 y_{a15}，x_{a16} 到 y_{a20}。由图 3-6 可以看出 BDFT 算法的 CDG 无法形成封闭的环路，则所提算法无死锁。

3.4 实验仿真和性能分析

3.4.1 模拟器

本实验采用仿真器 Booksim 2.0[31]对算法的性能进行仿真与验证，Booksim 2.0 是用 C++实现的面向片上网络的开源模拟器，并在微片级建模网络，是一个 NoC 网络性能仿真的平台。实验使用静态分析和动态分析对算法进行评估。仿真的主要参数配置如表 3-1 所示。

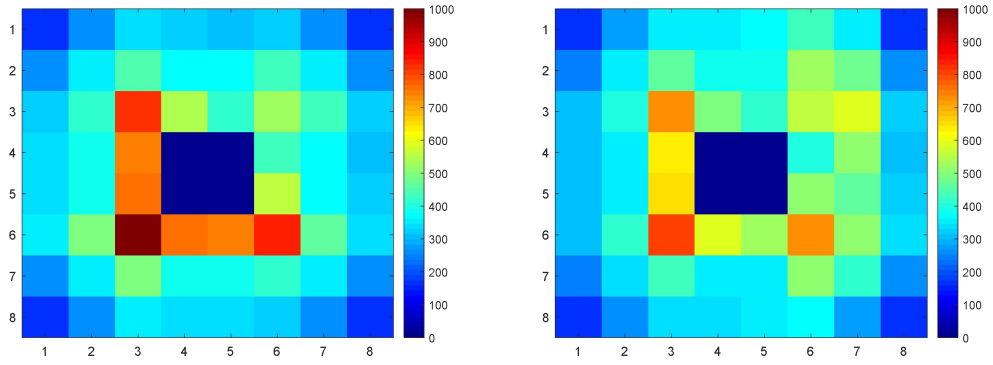
表 3-1　仿真器的参数配置

网络参数	结果
拓扑结构	8×8 2D Mesh
输入信道缓存大小	8 flits
数据包长度	2 flits
网络流量模式	Uniform
仿真时间	3000 cycles

3.4.2 静态分析

静态分析是为了测试网络负载的均衡度。具体测试方法是对网络中每对源节点到目标节点进行数据包的发送且只发送一个数据包，数据包遵循路由算法进行传输，继而在每一个节点处统计经过该节点的数据包数量。图 3-7 为一个 8×8 2D mesh 网络且有一个 2×2 的故障区域位于网络中心的流量负载分布图，分别对 FTRA 算法和 BDFT 算法的均衡度进行了直观的展现。

从图 3-7(a)可以发现，FTRA 算法的网络负载主要集中在故障块绕行环路的西边界和南边界，其东边界和北边界负载较轻，因此 FTRA 算法具有较差的网络负载均衡度。从图 3-7(b)可以看出，BDFT 算法具有较为均衡的网络负载，该算法把部分沿绕行环路西边界和南边界的数据包分散到了故障区域东侧传输，分散了故障区域周围的负载，从而改善了故障区域周围的网络负载不均衡现象，均衡了整个网络的负载。



(a)FTRA　　(b) BDFT

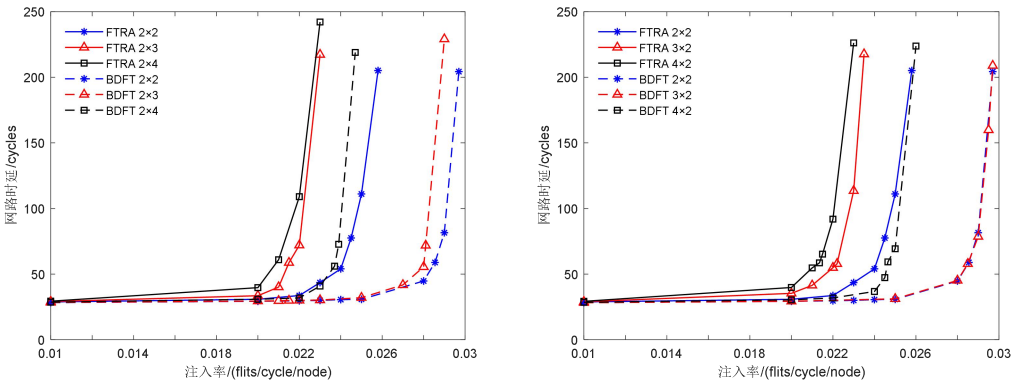
图 3-7　流量负载分布图

3.4.3 动态分析

动态分析采用的合成网络流量模式为均匀流量模式 Uniform。仿真在 8×8 的 2D Mesh 网络中进行，每条输入通道的缓存大小为 8 个 flits，数据包长度为 2 个 flits。对于每个注入率，仿真时间持续 3000 cycles，前 1000 cycles 属于预热周期不计入仿真结果。本章以饱和注入率η作为评价指标，随着注入率的增大网络进入饱和的越晚则η越大，说明网络性能越好。

图 3-8 展示了网络时延和注入率的关系曲线，从图 3-8(a)中可以看出，当位于网络中心的故障区域横向扩展为 2×2，2×3 和 2×4 时，在注入率较低时算法平均时延基本一致，伴随注入率的增加会有一个迅速提升的过程，且所提算法更晚进入饱和状态。随着故障区域横向扩展，当网络时延为 57cycles 时，与 FTRA 算法相比，BDFT 算法的饱和注入率分别提高了 18.75%，30.23%和 12.85%。从图 3-8(b)中可以看出，随着位于网络中心的故障区域纵向扩展为 2×2，3×2 和 4×2 时，当网络时延为 57cycles 时，与 FTRA 算法相比，BDFT 算法的饱和注入率分别提高了 18.75%，28.37%和 15.49%。这是因为 BDFT 算法分散了部分数据包的绕行传输，缩短了数据包的绕行距离，使网络负载更加均衡化。为了更精确的对比，表 3-2 和表 3-3 分别列出了故障区域横向扩展与纵向扩展的饱和注入率对比，其平均饱和注入率分别提高了为 20.81%、20.89%。

从上述分析可以看出无论如何扩展故障区域面积，比起 FTRA 算法，BDFT 算法都有更高的饱和注入率和更低的延时。即 BDFT 算法具有更好的网络性能。



(a) 故障区域横向扩展比较　　(b) 故障区域纵向扩展比较

图 3-8　网络时延比较曲线

表 3-2　故障区域横向扩展饱和注入率对比η(flits/cycle/node)

	η		
	BDFT	FTRA	Δ%
故障区域 2×2	0.0285	0.024	+18.75
故障区域 2×3	0.028	0.0215	+30.23
故障区域 2×4	0.0237	0.021	+12.85
平均	0.0267	0.0221	+20.81

表 3-3　故障区域纵向扩展饱和注入率对比η(flits/cycle/node)

	η		
	BDFT	FTRA	Δ%
故障区域 2×2	0.0285	0.024	+18.75
故障区域 3×2	0.0285	0.0222	+28.37
故障区域 4×2	0.0246	0.0213	+15.49
平均	0.0272	0.0225	+20.89

3.5 本章小结

本章通过对 FTRA 算法的改进提出了一种无虚通道容错路由算法，算法针对数据包在绕行路径上绕行而增加绕行距离问题做出改进。算法通过允许部分转向，减少了数据包的绕行，从而缩短绕行距离。同时为了避免死锁问题，添加了禁止转向，使得网络负载更加均衡化。仿真结果分析表明，相比于参考算法，所提算法能够有效的降低数据包的平均延迟，提高饱和注入率，使网络保持良好的性能。

第 4 章 网格型 CGRA 流水映射的延时评估

4.1 引言

随着计算系统的深入发展，并行处理数据运算已成为当前主流的计算模式之一。第 3 章对路由型 CGRA 的容错性能进行了探讨，而本章节将在网格型 CGRA 上展开流水映射的研究。基于 CGRA 平台的流水映射算法大多考虑运算节点并行执行的启动间隔 II，在流水执行成本方面研究较少。针对此缺陷本章节在运算节点层面上进行指令级流水时延的分析，提出一种流水执行成本的量化方法，同时根据该评估时延设计一种低延时的映射算法。

4.2 问题定义

定义 1 数据流图：DFG可以用四元组G=(V, E, W, D)来表示，同时针对每个有向边赋以权值(dif , min)；顶点集V= {vi| vi是有序运算符，1≤i≤n}；边集E={eij| eij =<vi, vj>,1≤i, j≤n}, vi, vj为两个有依赖的运算节点，eij则表示vi, vj的依赖有向边；权集W={wi | 表示vi消耗的硬件资源面积，1≤i≤n}；D表示为运算节点的执行时延集，di∈D代表vi运算节点的延迟；dif表示循环间依赖跨度的迭代距离，min为依赖中前驱算子的执行时延。

定义 2 启动间隔：II 为实际上两个循环间的启动间隔，II 越小则吞吐量及性能越好。MII 为 II 理论上两个循环间的最小启动间隔，其直接由循环 DFG 的依赖性决定如公式 4-1；*ResMII* 是基于硬件资源计算出来的最小间隔时间如公式 4-2；根据循环 DFG 的依赖关系而计算出来的最小迭代间隔 *RecMII* 如公式 4-3 所示。其中，n 为 DFG 的节点个数，N_{PE} 为 CGRA 的硬件资源数量；$\min_\theta$表示 DFG 中循环间依赖跨度的迭代距离，dif_θ表示 DFG 中循环上所有操作的总时延。

$$MII = max(ResMII, RecMII) \tag{4-1}$$

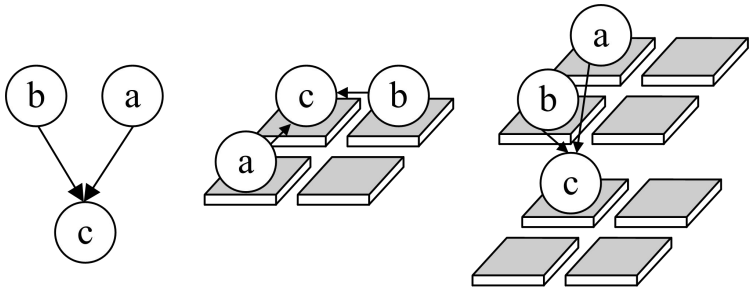
$$ResMII = n / N_{PE} \tag{4-2}$$

$$RecMII = max(\forall cycle\theta)[\min_\theta / dif_\theta] \tag{4-3}$$

定义 3 可重构处理单元阵列流水：PEA一条指令的流水执行划分成七级流水，分别为取指令(Fetch Instruction, FI)、指令译码(Instruction Decode, ID)、计算

操作数地址(Computing Operand_address, CO)、取数(Fetch Data, FD)、硬件配置(Hardware Configuration, HC)、执 行(Execute, EX)、写回(Write Back, WB)。具体功能为：FI 从指令存储器取出一条指令并暂时存入指令部件的缓冲区；ID根据取址信号来确定特定的功能和对应的操作数地址；CO的功能是计算操作数的有效地址；FD根据操作数地址从PE寄存器堆或内存来读取相应的数据；HC通过CGRA的配置寄存器来定义PE计算以及路由等功能；EX根据指令类型执行算术逻辑或复杂混合运算；WB将PE计算得到的结果写回数据存储单元或传输到下一依赖的PE寄存器堆以作为数据来源。

定义 4 依赖映射：按照映射算法把DFG节点映射在可重构处理阵列上，根据节点映射的位置不同可以分为块内映射和块间映射；相互依赖的节点映射在不同块的计算单元阵列上表示为块间依赖映射，映射在同一块计算单元阵列上的映射表示为块内依赖映射，如图 4-1 所示。



(a)双目运算　(b)块内依赖映射　(c)块间依赖映射

图 4-1　依赖映射示意图

定义 5 PEA 的度：在同一块二维 PEA 中将 PE 可以依赖传输的数量称为 PEA 的度，PE 的位置不同其度的大小不同，最小的度为 2，最大的度为 4，度的具体坐标如图 4-2 所示：

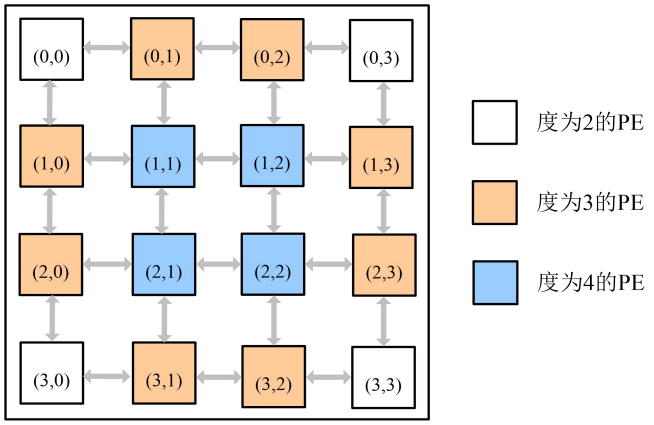


图 4-2　PEA 的度

4.3 流水延时量化及映射算法设计

4.3.1 实验动机

可重构计算处理器由于运算阵列的结构设计，其运算单元的配置包含了对指令级并行的支持，具有并行化的特点，PEA可以同时执行不同的指令，实现指令级流水并行计算。流水映射算法可以减少启动间隔II来增强映射效果，但流水成本的考虑略显不足，对此研究流水映射时延具有一定意义，本章通过对块内依赖映射和块间依赖映射进行流水成本的综合量化，从而达到对流水映射的时延评估，同时针对该评估时延提出一种低延时的映射算法(LDM)。

4.3.2 补码运算时延

本章节对粗粒度可重构阵列进行定点数的指令级流水时延分析，采用操作数位数为 8 位的带符号补码加法器、减法器、乘法器和除法器的仿真时延，具体时延及其它操作运算时延Δt_E(单位为 cycle)如表 4-1 所示，并且配置操作时延为 1 cycles。

表 4-1　运算时延(cycles)

操作类型	加法	减法	乘法	除法	移位	取模	异或	赋值	取值	判断
Δt_E(cycles)	2	3	10	11	1	1	1	1	1	3

4.3.3 量化方法设计

软件流水调度算法的具体思想：根据算子依赖关系，固定单次循环内的算子顺序，使用相同的启动间隔II依次进行第二次、第三次循环。大多数流水算法改善了调度结果，减少了启动间隔II，增强了软件流水的映射效果；但从硬件流水的指令级分析，流水执行成本没有具体的考虑，粗粒度可重构执行单元的具体运算不同，其执行时延不同，会导致流水时延不同。下面将提出一种流水成本的量化方法，对流水映射算法进行时延评估分析。

量化方法：将指令流水的处理过程分为m个阶段(m = 7)，即指令七级流水；分别为取指令FI、指令译码ID、计算操作数地址CO、取数据FD、硬件配置HC、执行EX、写回WB。假设$PEA_{a\times b}$的大小A_{PEA}= a×b，有m段，n个计算任务，运算节点的执行时间为Δt_E，不同操作类型的Δt_E不一定相等，具体为上述补码运算时

延，其他流水段的时间为Δt(Δt=1cycles)，流水映射的执行成本$S_{pipeline}$分为以下两种情况进行量化，具体见公式 4-4，4-5。

(1)块间依赖映射

采用并行化指令流水，即块间节点的IF、ID、CO、FD、HC、EX和WB可以在同一周期内完成。设有n个计算任务，指令流水段数为m，执行时间为EX_i ($i \in [1,n]$)，其他流水段为Δt，且$EX_i=\Delta t_E \geq \Delta t$，流水成本如公式 4-4 所示；$EX_{node_max}$为PEA块间依赖节点的最大执行时延，其决定了块间依赖流水段的执行时延，则$\theta=EX_{node_max}-\Delta t$；若$EX_{node_max}=\Delta t_E=\Delta t$，则$\theta=EX_{node_max}-\Delta t=0$；若$EX_{node_max}=\Delta t_E>\Delta t$，则$\theta=EX_{node_max}-\Delta t>0$。

块间有依赖流水成本：

$$S_{pipeline}= m\cdot\Delta t+\theta,\ \theta=EX_{node\_max}-\Delta t \tag{4-4}$$

(2)块内依赖映射

将循环DFG映射到 7 段流水PEA，针对于块内有依赖映射，要保证指令流水时延最小化，首先将当前节点的FI、ID和CO段与上个依赖节点的HC、EX和WB段并行执行，然后使当前节点的取数据FD段在上个依赖节点的写回WB段后执行；这样既保证了指令流水时延最小化，又使流水顺利执行；块内无依赖映射，则根据DFG依赖路径的父节点进行指令级并行流水。PEA块内依赖映射的最大依赖距离为$L_\max$，$EX_1,EX_2,...,EX_m$为该距离内m个节点的执行时延($EX_i=\Delta t_E$, $i=1,2,...,m$)。设块内依赖的第一个节点有d个节点块间并行执行，即当$level_i=1$时，那么$T_1=(m-1)\Delta t+\max\{EX_1,EX_2,...,EX_d\}$；块内依赖的第二个节点有e个节点块间并行执行，且依赖于第一个节点，FD_{level_2}在WB_{level_1}之后执行，则流水段会少 3Δt，当$level_i=2$时$T_2=(m-1)\Delta t-3\Delta t+\max\{EX_1,EX_2,...,EX_e\}$；依此类推，则对于块内依赖的最后一个节点，$level_i=L_\max$时，有$T_{L_\max}=(m-1)\Delta t-3\Delta t+\max\{EX_1,EX_2,...,EX_m\}$；那么累加和可以得出$S_{pipeline}=T_1+T_2+...+T_{L_\max}$。

块内有依赖流水成本：

$$S_{pipeline}=L\_\max\cdot m\cdot\Delta t-3(L\_\max-1)\Delta t+\sum_{level\_i=1}^{L\_\max}\{\max\{EX_1,EX_2,...,EX_m\}-\Delta t\} \tag{4-5}$$

举例：根据所提出的量化评估方法，基于七级指令流水对寄存器感知映射算法REGIMap进行流水时延量化示例，如图 4-3 所示：图a为给定的循环DFG，a、b、c、d为四个算子。针对该DFG，图b为REGIMap在 1×2 CGRA上的映射结果，启动间隔II为 2，算子a到b，b到c，c到d为块内无依赖，块间有依赖执行，而算子a到d运用了自带的寄存器进行传输，为块内有依赖执行，则算子d的取数应该在算子a的写回之后。下面分别对DFG算子赋予加、减、乘、除运算及混合运算，可以得出具体的流水时延成本和流水时空图。

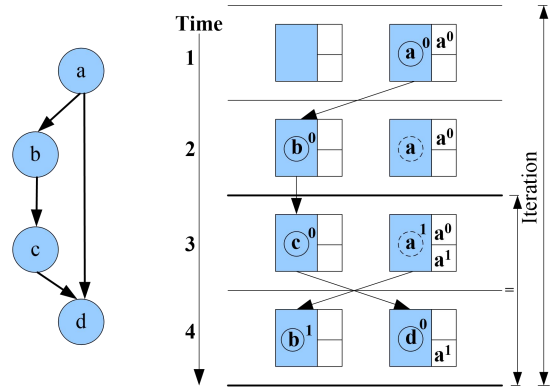
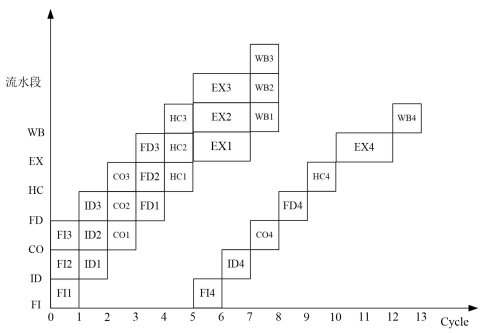
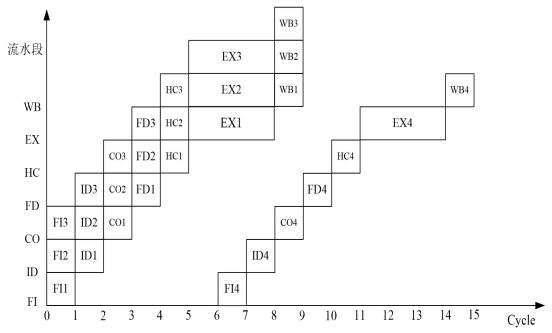


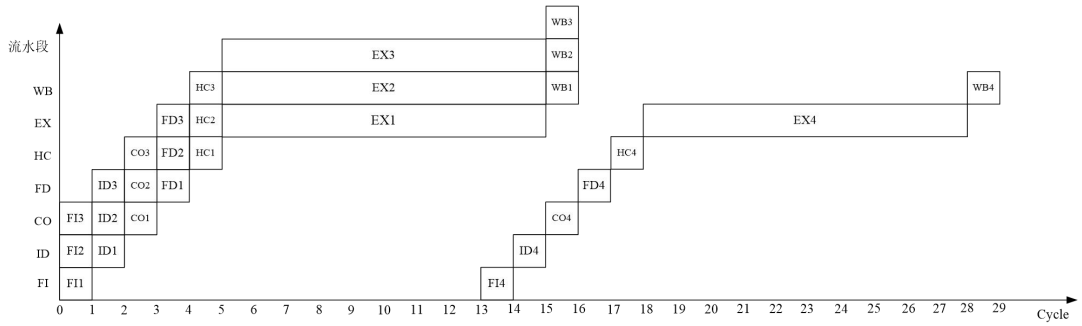
图 4-3　REGIMap 映射



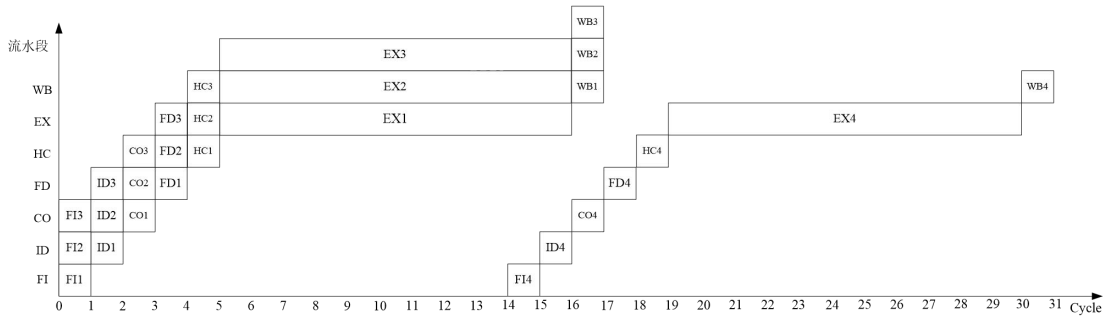
(a) 加法运算



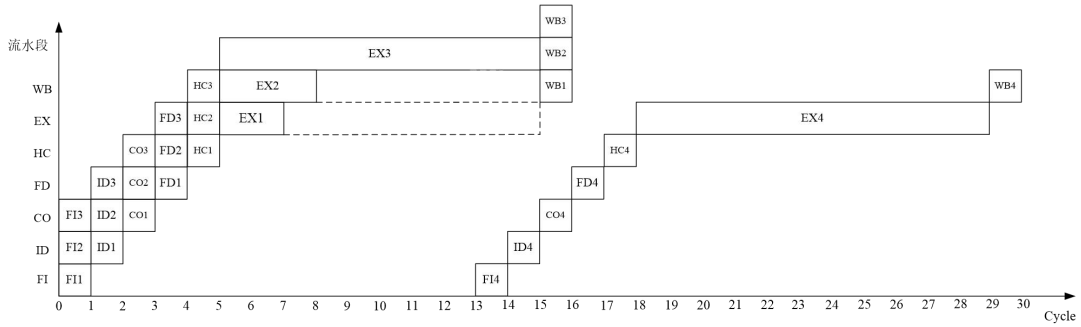
(b) 减法运算



(c) 乘法运算



(d) 除法运算



(e) 混合运算

图 4-4　流水量化时空图

REGIMap为一种流水映射算法，该算法在传递数据时利用了运算单元自带的寄存器来改善调度结果。从硬件流水的指令级分析，该算法的流水执行成本根据粗粒度可重构执行单元的运算不同，会导致流水时延不同。流水时空图可以展现出DFG流水映射的流水成本及成本评估的过程；从时空图的分析可以看出块间依赖的映射并行执行，其中执行时延最大的运算决定了整个块间依赖映射的并行时延，而块内依赖映射又取决于上一节点的并行时延，因为块内相互依赖的节点需要满足取数与写回之间的传递。从图 4-4 的量化结果来看，DFG执行加法计算时延为 13 cycles，执行减法计算时延为 15 cycles，加减混合计算时延为 15 cycles，乘法计算 29 cycles，除法计算 31 cycles，乘除混合计算 31 cycles，四则混合计算时延为 30 cycles。

4.3.4 映射算法设计

针对流水映射的执行成本 $S_{pipeline}$，本章节考虑了运算节点的时延均衡化等要素，设计了一种低延时映射算法(Low Delay Mapping, LDM)，具体如下所示：

LDM 针对网格型 PEA，采用以度为 2 的点($PEA_{(0,\ 0)}$)为调度起点进行映射。为了减少 $S_{pipeline}$，LDM 采用路由布线的映射方式，同时以规定路由布线的距离

来增加映射效果，小范围的跨层操作我们采用路由布线的映射方式，大范围的跨层映射我们利用运算单元的寄存器堆来传输。这样在保证一定映射效果的同时，减少了流水执行成本 $S_{pipeline}$，LDM 算法的伪码简介如表 4-2 所示。

如图 4-5 所示 DFG，其为两次循环展开图，将其进行 LDM 映射，映射目标为 2×2 CGRA，同时限制路由布线的距离为 3，如 v_1-v_4、v_4-v_6，对路由布线距离超过 3 以上的跨层操作利用运算单元的寄存器传输，如 v_1-v_6。最终 LDM 映射结果如图 4-6 所示，同时给出了数据的传输路径。

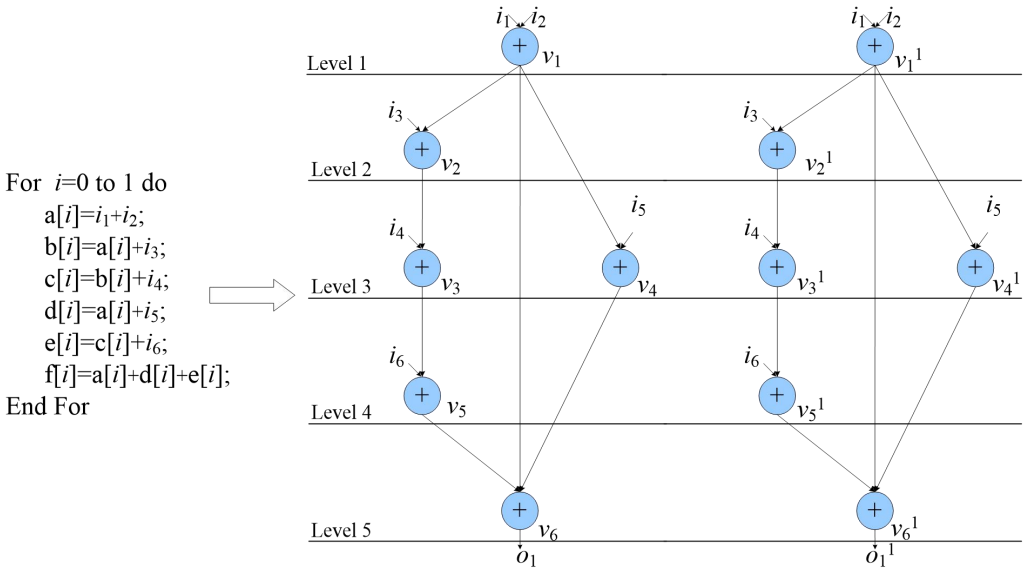
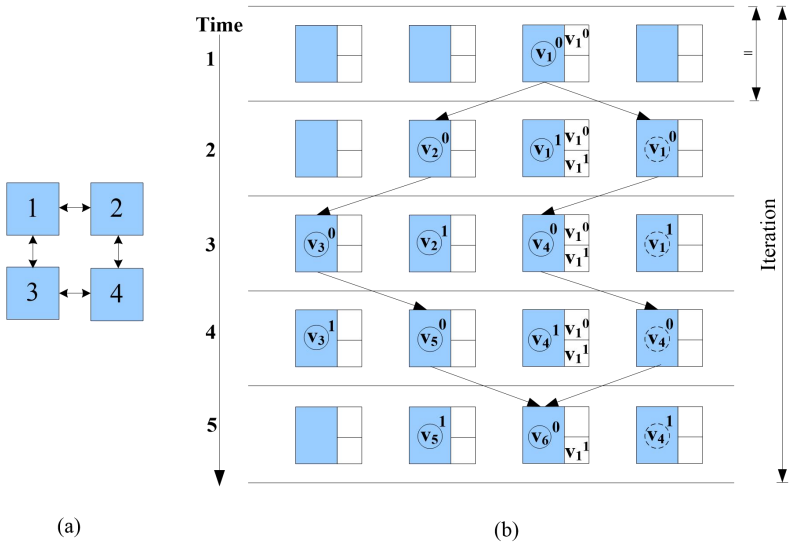


图 4-5　DFG 的提取



(a)　　(b)

图 4-6　LDM 映射

表 4-2　LDM 算法伪码

算法名称：LDM
输入：DFG
输出：$S_{pipeline}$
约束：A_{RPU}=PEA$_{row\times col}$，定点处理阵列块内运算节点顺序没有发生非法依赖关系
Step1:读入循环 DFG; PEA 矩阵初始化; M=0, n=0, $nodenum$=0; $S_{pipeline}$=0;
Step2: for v_i =1 to n **do**
{
if (节点 v_i 的入度为 0)
节点 v_i 加入就绪队列;
else break;
}
Step3: while (n==$nodenum$)
{
for (i=0;i<$nodenum$;i++)
{
Step4: 就绪队列中选取节点号最小的当前节点 v_i，调度放入 PEA 中度为 2 的位置(PEA$_{(0,0)}$);
Step5: **if** (节点 v_i 与其直接后继层差=1）
{依次调度 v_i 直接后继放入下一块 PEA;}
else if (节点 v_i 与其直接后继层差≤3)
{ 若下一块 PEA 有合法位置，用其某个 PE 作为路由节点传输;}
else{ 将节点 v_i 的直接后继存入 PE 的寄存器堆;}
}
Step6: Refresh PEA;
if(节点全部映射完毕){ **break**;}
}
Step7: 输出流水时延的值 $S_{pipeline}$;
Step8: end LDM

4.4 实验结果和比较分析

4.4.1 实验基准

本章节以灰度直方图统计(Gray Histogram Statistics, GHS)、图像测量(Image Measurement, IM)等图像运算和椭圆波滤波器(Elliptic Wave Filter, EWF)、快速离散余弦变换(Fast Discrete Cosine Transform, FDCT)等计算密集型任务为基准程序。为测试量化评估方法，对层流水映射算法进行流水执行成本统计，同时根据该量化方法对REGIMap、LDM进行流水执行成本的分析比较，具体基准程序集如表 4-3 所示：

表 4-3　基准程序集

基准	总数	加法	减法	乘法	除法	赋值	取内容	判断
GHS	24	8	—	4	—	4	8	—
IM	40	8	—	—	—	16	8	8
IPM2	40	8	8	4	—	8	8	4
FDCT	42	13	13	16	—	—	—	—
EWF	34	28	—	6	—	—	—	—
FFT8	36	12	—	12	12	—	—	—

4.4.2 实验结果

为了便于流水执行成本的量化评估测试，本研究选择了一系列基准程序，包括FDCT、EWF和FFT8 等计算密集型任务，并在表 4-3 中列出。同时为了测试量化评估方法研究不同A_{PEA}(即PEA的面积)值的性能效果，本节选择了A_{PEA}= 9 ($PEA_{3\times3}$)和A_{PEA}= 16 ($PEA_{4\times4}$)两种情况，针对不同的A_{PEA}值，采用层流水映射算法进行流水执行成本的量化评估测试。并且基于A_{PEA}= 16 的情况，对REGIMap、LDM算法进行流水执行成本$S_{pipeline}$的比较分析。

(1)层流水量化统计

基于一组测试基准程序，在$PEA_{3\times3}$ 和$PEA_{4\times4}$ 上，对层流水算法进行量化评估统计，EWF的层流水时延分别为 74cycles和 79cycles，FDCT的层流水时延分别为 47cycles和 39cycles，IM的层流水时延分别为 31cycles和 27cycles，具体如图 4-7 和图 4-8 所示。由于层流水映射块内依赖较为集中，由此流水成本较高，根据实验结果表明，所提量化方法在网格型粗粒度可重构系统结构上具有可行性。

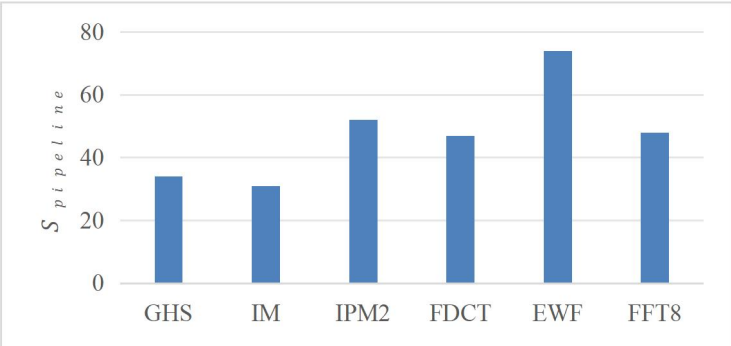


图 4-7　A_{PEA}=9 时 层流水 $S_{pipeline}$(cycles)量化结果

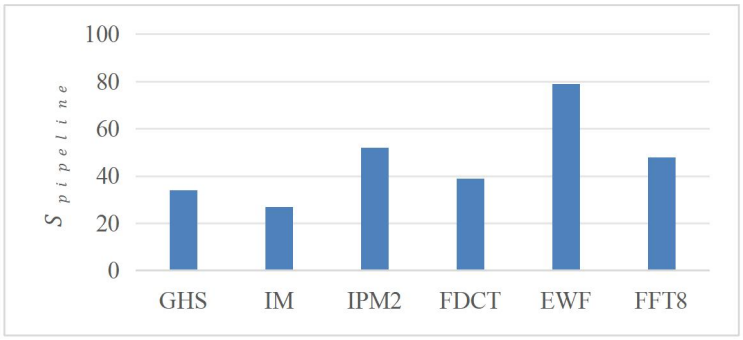


图 4-8　A_{PEA}=16 时 层流水 $S_{pipeline}$(cycles)量化结果

(2)$S_{pipeline}$比较

由图 4-9 可以看出，在A_{PEA}=16 时，相较于REGIMap，LDM算法获得了$S_{pipeline}$的优化，针对本章基准程序，平均时延成本降低了 30.1%，这是因为LDM减少了块内串行依赖的时延，增加了指令流水并行化，由此可见LDM针对$S_{pipeline}$的有效性要优于REGIMap，具体统计如表 4-4 所示。

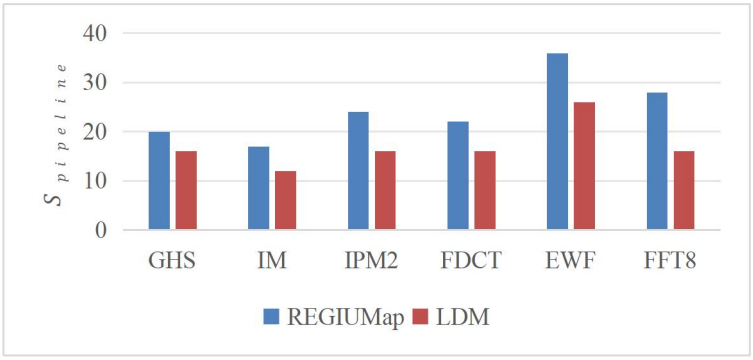


图 4-9　A_{PEA}=16 时 REGIMap, LDM 算法的 $S_{pipeline}$(cycles)对比

表 4-4　A_{PEA}=16 时 REGIMap, LDM $S_{pipeline}$(cycles)统计

映射基准	$S_{pipeline}$		
	REGIUMap	LDM	△%
GHS	20	16	-20.0
IM	17	12	-29.4
IPM2	24	16	-33.3
FDCT	22	16	-27.3
EWF	36	26	-27.8
FFT8	28	16	-42.9
平均△%	—	—	-30.1

4.5 本章小结

本章节对流水映射的执行时延成本进行研究，通过硬件指令级的流水分析，提出了一种流水成本的量化评估方法，该方法使用七段指令流水将流水映射的执行成本从块间依赖映射和块内依赖映射两个维度进行综合，从而对流水映射的总时延进行量化评估。同时针对该评估时延提出了一种低延时的映射算法。基于测试基准程序表明，所提量化方法在网格型粗粒度可重构系统结构上具有可行性，且验证了映射算法的有效性。

第 5 章 路由型 CGRA 映射的延时评估

5.1 引言

随着可重构多核处理器研究的兴起，本论文的第 3 章重点探讨了路由型 CGRA 的容错通信性能，而第 4 章则专注于对网格型 CGRA 的流水执行成本进行研究。相比较于网格型 CGRA，路由型 CGRA 的 PE 之间通过路由器相连，增加了数据传输的可靠性，然而也增加了传输时延。因此，本章节旨在对路由型 CGRA 进行映射时延的量化评估。

5.2 路由传输延时的量化评估

5.2.1 量化分析

路由型 CGRA 互连结构如图 5-1 所示，其数据的传输过程涉及多个模块和组件；首先处理单元 PE 对数据进行计算，然后将计算结果传输到路由器中。路由器结构如图 5-2，其拥有五个端口，其中一个连接到本地处理单元 PE，其余四个连接相邻的路由节点。在传输过程中，寄存器文件堆(Register Files, RF)用于临时存储待转发的数据包。交叉开关矩阵连接着不同的端口，通过路由协议的控制，它选择输入缓存中的数据，并将其发送到指定的目标端口。如此数据由本地 PE 经过路由器包括交叉开关矩阵等组成部分，最终到达目标 PE 所对应的路由节点。

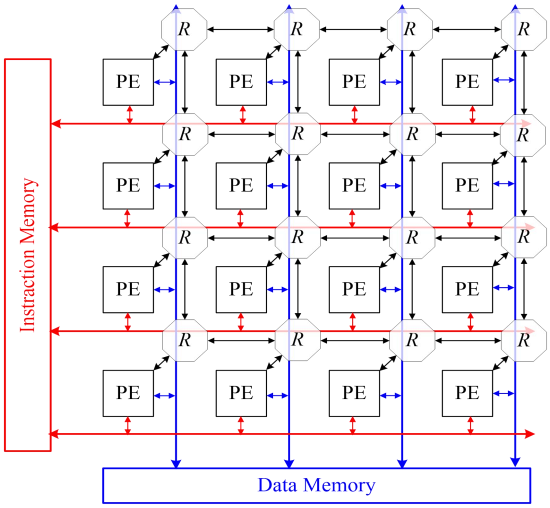


图 5-1　路由型 CGRA 互连结构

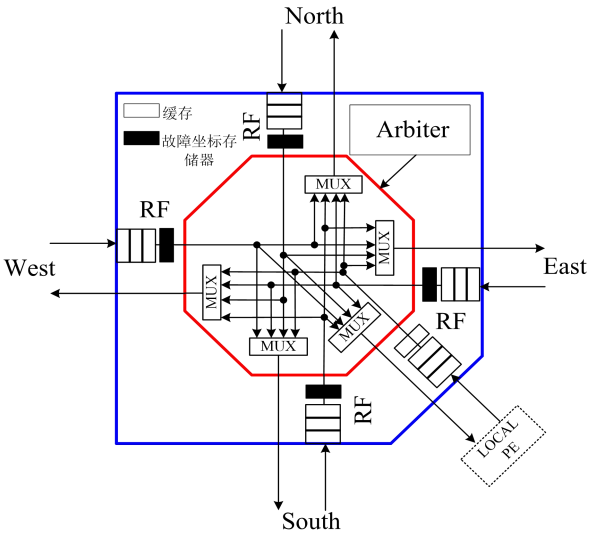


图 5-2　路由器结构

路由型 CGRA 中数据经过当前路由器的接收缓存直至到达下游路由器，这一过程称为最小传输单元(minimum transmission unit, MTU)。该过程 T_{MTU} 具体分为六个串行累加步骤，包括路由器的缓存写入 T_{BW}、路由计算 T_{RC}、虚通道分配 T_{VA}、交叉开关分配 T_{SA}、交叉开关传输 T_{ST} 以及链路传输 T_{LT}，如公式 5-1。一个具有 F 个 flit 的数据包，在不考虑阻塞的情况下，按照基于数据包的存储转发流控，经过 n 个 MTU 的路由传输时延 T_r 如公式 5-2 所示。

$$T_{MTU}= T_{BW}+T_{RC}+T_{VA}+ T_{SA}+T_{ST}+T_{LT} \tag{5-1}$$

$$T_r=n\times T_{MTU}\times F \tag{5-2}$$

相较于网格型 CGRA，路由型 CGRA 在数据传输方面增加了路由时延，而其处理单元的计算时延与网格型 CGRA 相同，与算子的具体运算相关。总时延 T_s 代表了从数据离开源计算单元，到达目标计算单元，并在目标计算单元完成计算操作所需的总耗时。路由型 CGRA 映射总时延 T_s 由关键路径中的路由传输时延 T_r 和处理单元的运算时延 $S_{pipeline}$ 累加计算得到，其中 $S_{pipeline}$ 为关键路径中的总运算时延，按第四章的补码运算进行块内或块间时延评估，则路由映射总时延 T_s 如公式 5-3 所示。

$$T_s= T_r+S_{pipeline} \tag{5-3}$$

5.2.2 评估分析

以图 5-3 所示的 DFG 为例，将其以度为 4 为映射起点($PEA_{(1,\ 1)}$)，按行优先级进行放置到网格型 CGRA 结构和路由型 CGRA 结构中。在此过程中，设置数

据包的 F 为 3，以及一个最小传输单元时延 T_{MTU} 为 1 个时间周期。最终的映射结果如图 5-4 所示，并且给出了数据在各个结构中的传输路径。

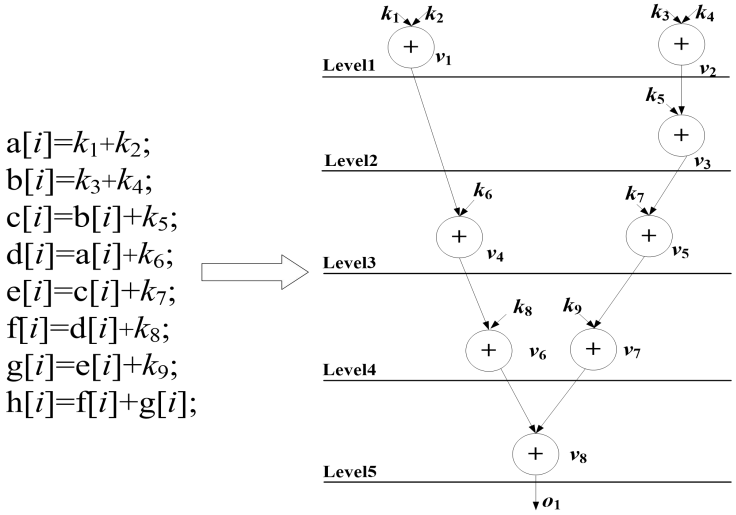
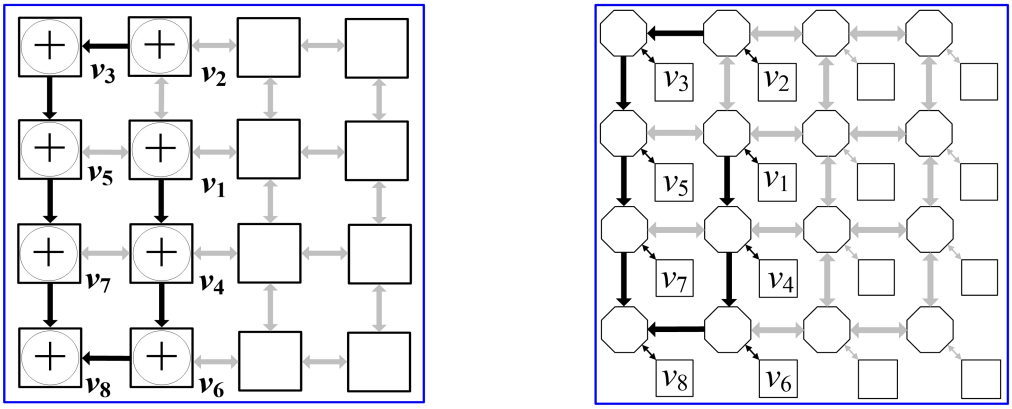


图 5-3　DFG 的提取



(a)网格型 CGRA 映射　　(b)路由型 CGRA 映射

图 5-4　网格型 CGRA 和路由型 CGRA 的映射结果

网格型 CGRA 和路由型 CGRA 的映射结果，如图 5-4 所示。网格型 CGRA 占用了 8 个处理单元，而路由型 CGRA 占用了 8 个处理单元和 8 个路由器。

针对网格型 CGRA，按第四章给出的延时量化方法，该 DFG 执行完所有的运算节点时延 $S_{pipeline}$ = 28 cycles。

针对路由型 CGRA，根据本章节给出的路由延时量化方法，该 DFG 的路由传输时延 T_r 由关键路径中经过的路由最小传输单元等构成，T_r = 12 cycles，处理单元的运算时延为 $S_{pipeline}$，则该路由型 CGRA 的总延时 T_s = 40 cycles。

相对于网格型 CGRA，在路由型 CGRA 中，更灵活的通信网络结构和路由算法，能够提供更可靠的数据传输。然而，路由型 CGRA 在传输效率方面的表

现相对较低，这是由于路由算法和较长的路径长度导致的。对于路由型 CGRA，通过优化路由传输时延和提高处理单元的运算并行度，可以降低总时延，提高 CGRA 的性能和效率。需要注意的是，对于不同的路由架构和具体的应用场景，总时延和关键路径的具体情况可能会有所不同。所以在实际设计中，需要根据具体需求进行详细建模和分析。

5.3 路由映射的延时量化实验

5.3.1 实验基准

本章节针对路由型CGRA，以基于 8 采样的快速傅里叶变换(Fast Fourier Transformation, FFT8)、中值滤波器(Median Filter, MF)、有限冲激响应滤波器(Finite Impulse Response Median Filter, FIR)等计算密集型任务为基准程序，以度为 4 为映射起点($PEA_{(1,\ 1)}$)，按行放置算法进行路由映射的时延成本统计。基准程序集如表 5-1 所示：

表 5-1　基准程序集

基准	总数	加法	减法	乘法	除法	赋值	取内容	判断
ROBERT	29	9	2	4	3	5	6	—
FFT8	36	12	—	12	12	—	—	—
FIR	26	17	—	9	—	—	—	—
MF	19	—	—	—	—	—	—	19
BETREE32	31	—	—	—	—	—	—	31

5.3.2 实验结果

为了便于路由型CGRA映射的延时量化评估，本研究选择了一系列基准程序，包括FIR、ROBERT和FFT8 等计算密集型任务，如表 4-2 所示。同时针对规模A_{PEA} = 16 的路由型CGRA，设置数据包的大小F为 3，T_{MTU}为 1 个时间周期，采用以度为 4 为映射起点的行放置算法进行路由映射延时的量化评估测试。网格映射和路由映射总延时T_s对比结果如图 5-5 所示，相比较不考虑容错路由的行放置映射，考虑路由时延的路由映射，在总时延方面平均多消耗了 25.1%，具体如表 5-2 所示。实验结果表明，相比较于网格映射，路由映射的传输时延较高，在传输效率方面的表现相对较低，这是由于路由算法和较长的路径长度导致的，但路由型CGRA具有灵活的通信网络结构和路由算法，能够提供更可靠的数据传输。同时，实验还表明了基于路由型CGRA映射的延时量化方法在实践中是可行的，并且能

够有效地帮助优化映射和调度。

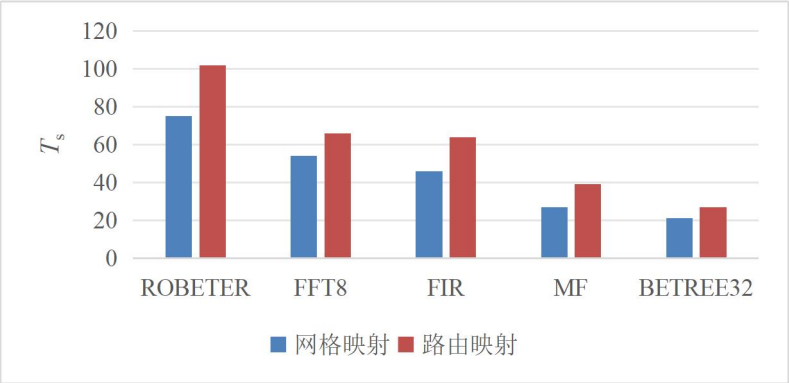


图 5-5　A_{PEA}=16 时　网格映射和路由映射时延(cycles)对比

表 5-2　A_{PEA}=16　时　网格映射和路由映射时延(cycles)统计

映射基准	T_s		
	网格映射	路由映射	△%
ROBERT	75	102	+26.4
FFT8	54	66	+18.2
FIR	46	64	+28.1
MF	27	39	+30.8
BETREE32	21	27	+22.2
平均△%	—	—	+25.1

5.4 本章小结

针对路由型 CGRA，本章节通过计算单元、路由传输等因素，提出了一种路由映射的时延量化方法，并且通过一组基准程序对网格映射和路由映射分别进行延时的量化对比。根据路由型 CGRA 的延时量化分析得出，路由型 CGRA 在传输可靠性方面具备优势，但在传输效率方面仍存在挑战；同时表明了基于路由型 CGRA 映射的延时量化方法在实践中具有可行性。

第 6 章 总结和展望

6.1 工作总结

随着可重构多核处理器研究的展开，粗粒度可重构计算系统得到广泛的应用研究。本工作基于二维路由型 CGRA 提出了一种存在多故障节点的容错路由算法，以解决网络通信等问题；并且针对二维网格型 CGRA，提出了一种流水成本的量化评估方法及映射方法，给出了流水映射成本的时延分析；同时针对二维路由型 CGRA，分析处理单元、数据路由传输等要素，给出路由映射的总时延评估方法。具体工作内容如下：

(1) 提出了一种基于路由型 CGRA 的无虚通道容错路由算法，算法针对数据包在绕行路径上的绕行距离问题做出改进。依赖于内建自测试获取故障区域的位置信息，通过结合故障模型和绕行算法，使用了较低复杂度的绕行策略，相比较于 FTRA 算法，所提算法改善了故障模型周围的负载不均衡现象，且提高了网络性能。

(2) 针对网格型 CGRA 中流水映射算法在流水执行成本方面的研究问题，本研究提出了一种流水成本的量化评估方法。该方法通过在运算节点层面引入硬件指令流水，综合考虑块间依赖映射和块内依赖映射两个维度的执行成本，从而对流水映射的总时延进行准确的度量和评估。并且基于该评估时延设计了一种低延时的映射方法。并且通过一组基准程序集，测试了该量化方法的可行性及映射方法的有效性。

(3) 针对路由型 CGRA，本研究通过处理单元、路由传输等要素，提出了一种路由映射的时延量化方法，且基于一组基准程序集，进行路由映射总时延的量化统计。通过统计结果和实验分析，得出了以下结论：路由型 CGRA 的数据传输可靠性较强，但数据传输效率较低，同时验证了该量化方法在实践中是可行的。

6.2 研究展望

目前对于路由型 CGRA 的网络通信研究、时延评估工作仍存在许多问题需要解决，本文对网格型 CGRA 的流水成本量化研究同样需要进一步的完善。

(1) 对于路由容错，可以在少量的功耗的基础上，添加部分虚拟通道，在增加传输效率的同时进行容错传输，以提高网络性能。

(2) 可重构流水映射的时延研究，可以根据实际情况进行更具体的建模，在低延时情况下对流水性能进行展开研究。

(3) 对路由传输的进行更加完整具体的建模，通过优化路由传输时延和提高处理单元的运算并行度，可以降低总时延，提高的性能和效率。同时在实际设计中，需要根据具体需求进行详细建模和分析。

参考文献

[1]NOOR D M, SIRAY Q M, MAHA A K. Implementation of 4×4 2D Mesh NoC Architecture using FPGA[C]//Proceedings of 2021 1st Babylon International Conference on Information Technology and Science (BICITS), Babil,2021: 133-137.

[2]MUHAMAD K, ISMAIL F B I. A Survey on Network on Chip Routing Algorithms Criteria[C]//Proceedings of Advances on Smart and Soft Computing, 2021: 455-466.

[3]MANZOOR M, MIR R N, et al. Prime Turn Model and First Last Turn Model: An Adaptive Deadlock Free Routing for Network-on-Chips[J]. Microprocessors and Microsystems, 2022, 89(3): 1-10.

[4]BANSIDHAR J, MANISH K T. A Traffic Intensive Virtual Channels Allocation Scheme in Network-on-Chip[J]. Arabian Journal for Science and Engineering,2023,48(8), 9619-9633.

[5]EBRAHIMI M, DANESHTALAB M. A Light-weight Fault-tolerant Routing Algorithm Tolerating Faulty Links and Routers[J]. Computing Archives for Informatics & Numerical Computation, 2015,97(6): 631-648.

[6]RAMAKRISHNA M, KODATI V K, GRATZ P V. GCA: Global Congestion Awareness for Load Balance in Networks-on-Chip[J]. IEEE Transactions on Parallel and Distributed Systems,2016,27(7): 2022-2034.

[7]ELTARAS T A, FORNACIARI W, ZONI D. Partial Packet Forwarding to Improve Performance in Fully Adaptive Routing for Cache-coherent NoCs[C]//Proceedings of 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Pavia:IEEE cs Press, 2019: 33−40.

[8]ZHANG Y J, FAN W B, HAN Z J, et al. Fault-tolerant Routing Algorithm Based on Disjoint Paths in 3-ary N-cube Networks with Structure Faults[J]. Journal of Supercomputing, 2021, 77(11): 13090-13114.

[9]KUROKAWA Y, FUKUSHI M. Effect of Virtual Channels for a Fault-Tolerant XY Routing Method with the Passage of Faulty Nodes[C]//Proceedings of the 2020 8th International Conference on Information and Education Technology, 2020: 267-272.

[10]ZHANG Z, GREINER A, TAKTAK S. A Reconfigurable Routing Algorithm for a Fault-tolerant 2D-Mesh Network-on-Chip [C]//Proceedings of 2008 45th ACM/IEEE Design Automation Conference. Anaheim, United states: IEEE Computer Society, 2008:441-446.

[11]姚磊,蔡觉平,李赞,等.基于内建自测技术的 Mesh 结构 NoC 无虚通道容错路由算法[J]. 电子学报, 2012, 40(5):983-989.

[12]李娇,徐海鹏,崇云锋,等.一种低延迟的无虚通道NoC容错路由算法[J]. 上海大学学报, 2019, 25(2):189-197.

[13]李娇,郭润龙,蔡升,等.一种转向均衡的 3D NoC 感知容错路由算法[J].上海大学学报(自然科学版),2020,26(05):726-734.
[14]李娇,晁月斌,冉峰,等.基于路径多样性的 3D NoC 流量均衡容错路由算法[J].现代电子技术,2021,44(07):147-152.
[15]谢瑞莲,焦继业,刘有耀.片上网络无虚通道隔离路由算法[J].计算机辅助设计与图形学学报, 2021, 33(5):806-814.
[16]ZHAO H, BAGHERZADEH N, WU J. A General Fault-Tolerant Minimal Routing for Mesh Architectures[J]. IEEE Transactions on Computers, 2017,66(7): 1240-1246.
[17]TAHERKHANI N, AKBAR R, SAFAEI F, et al. A Congestion-aware Routing Algorithm for Mesh-based Platform Networks-on-Chip[J]. Microelectronics Journal, 2021,114: 105145.
[18]GUAN J, CAI J P, WANG Y, et al. Fault-tolerant and Congestion Balanced Routing Algorithm for 2D Mesh NoCs[J]. Journal of Web Engineering, 2020,19(7): 1049-1066.
[19]RAHAMAN M M, GHOSAL P, DAS T S. Latency, Throughput and Power Aware Adaptive NoC Routing on Orthogonal Convex Faulty Region[J]. Journal of Circuits, Systems and Computers,2019, 28(04): 1950055.
[20]BHANU P V, RAHUL G, RAJAT K, et al. Fault-Tolerant Application-Specific Topology-Based NoC and Its Prototype on an FPGA[J]. IEEE Access,2021,9: 76759-76779.
[21]BHANU P V, RAHUL G, RAJAT K, et al. Flexible Spare Core Placement in Torus Topology Based NoCs and Its Validation on an FPGA[J].IEEE Access,2021,9: 45935-45954.
[22]JAGADHEESH S, BHANU P V, SOUMYA J, et al. Reinforcement Learning Based Fault-Tolerant Routing Algorithm for Mesh Based NoC and Its FPGA Implementation[J].IEEE Access,2022,10: 44724-44737.
[23]ZULQAR N, RASHID A, SHERAZ A,et al. A Network Adaptive Fault-Tolerant Routing Algorithm for Demanding Latency and Throughput Applications of Network-on-a-Chip Designs[J]. Electronics,2020,9(07): 1-18.
[24]CHARIF A, COELHO A, ZERGAINOH N E, et al. A Dynamic Sufficient Condition of Deadlock-Freedom for High-Performance Fault-Tolerant Routing in Networks-on-Chips[J]. Emerging Topics in Computing, IEEE Transactions on,2020,8(3): 642-654.
[25]BHANU P V,GOVIL V,JAGADEESH S, et al. Novel Fault-Tolerant Routing Technique for ZMesh Topology based Network-on-Chip Design[C]//Proceedings of the 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2020:1070-1074.
[26]FANG J,ZHANG D,LI X Q.ParRouting: An Efficient Area Partition-Based Congestion-Aware Routing Algorithm for NoCs[J]. Micromachines, 2020, 11(12): 1-18.
[27]ANUGRAH J, VIJAY L, MEENAKSHI T,et al.TRACK: An algorithm for fault-tolerant, dynamic and scalable 2D mesh network-on-chip routing reconfiguration[J], Integration: The VLSI Journal, 2020,72: 92-110.

[28]YASREBI S, REZA A,NIKRAVAN M,et al.A fuzzy integrated congestion-aware routing algorithm for network on chip[J]. Frontiers of Information Technology & Electronic Engineering,2021, 22(5): 741-756.

[29]BISWAJIT B. Maximal Connectivity Test with Channel-Open Faults in On-Chip Communication Networks[J].Journal of Electronic Testing. Theory and Applications, 2020, 36(3): 385-408.

[30]DALLY W J, SEITZ C L. Deadlock-free message routing in multiprocessor interconnection networks[J]. IEEE Transactions on Computers, 1987, 36(5): 547-553.

[31]NAN J, BECKER D U, MICHELOGIANNAKIS G, et al. A detailed and flexible cycle-accurate Network-on-Chip simulator[C]//Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software. Austin, United states: IEEE Computer Society, 2013:86-96.

[32]CHEN N J ,WANG Z,HE R X, et al. Efficient Scheduling Mapping Algorithm for Row Parallel Coarse-Grained Reconfigurable Architecture[J]. Tsinghua Science and Technology, 2021, 26(5): 724−735.

[33]CHEN N J, CHENG F, HAN C H, et al. Loop Subgraph-Level Greedy Mapping Algorithm for Grid Coarse-Grained Reconfigurable Array[J]. Tsinghua Science and Technology, 2023,28(2):330-343.

[34]CHIN S A, ANDERSON J H. An Architecture-Agnostic Integer Linear Programming Approach to CGRA Mapping[C]//Proceedings of 2018 55th Annual Design Automation Conference (DAC),San Francisco:ACM Press, 2018:24-28.

[35]MATTHEW J P W , JASON H A. Generic Connectivity-Based CGRA Mapping via Integer Linear Programming[C]//Proceedings of 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), San Diego:IEEE cs Press, 2019:65-73.

[36]MICHAEL C, MARCELO M,WESTERLEY C, et al. TRAVERSAL: A Fast and Adaptive Graph-Based Placement and Routing for CGRAs[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits & Systems,2021,40(8):1600-1612.

[37]MAHDI H, AVIRAL S, SARMA V. EPIMap: Using Epimorphism to map applications on CGRAs[C]//Proceedings of 2012 Design Automation Conference(DAC), San Francisco, CA, USA,2012:1284-1291.

[38]DAVE S, BALASUBRAMANIAN M, SHRIVASTAVA A. RAMP: Resource-Aware Mapping for CGRAs[C]//Proceedings of 2018 55th Annual Design Automation Conference (DAC), San Francisco: ACM Press, 2018: 1−6.

[39]KOU M Y ,GU J Y ,WEI S J, et al. TAEM: Fast Transfer-Aware Effective Loop Mapping for Heterogeneous Resources on CGRA[C]//Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference.Virtual Event USA,2020:1-6.

[40]BALASUBRAMANIAN M , SHRIVASTAVA A. PathSeeker: A Fast Mapping Algorithm for CGRAs[C]//Proceedings of 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE),Belgium:IEEE cs Press,2022:

268-273.
[41]LIU D J ,YIN S Y ,LUO G J,et al.Data-Flow Graph Mapping Optimization for CGRA With Deep Reinforcement Learning[J].IEEE Transactions on Computer-Aided Design of Integrated Circuits & Systems,2019,38(12): 2271-2283.
[42]陈乃金,江建慧,陈昕,等.一种考虑执行延迟最小化和资源约束的改进层划分算法[J].电子学报,2012,40(05):1055-1066.
[43]陈乃金,江建慧.融合面积估算和多目标优化的硬件任务划分算法[J].通信学报, 2013, 34(02): 40-55.
[44]陈乃金,江建慧.考虑通信成本和硬件碎片利用的簇划分算法[J].计算机辅助设计与图形学学报,2015,27(04):754-763.
[45]陈乃金,江建慧.多叉树数据流图粗粒度可重构单元阵列映射算法[J].计算机辅助设计与图形学学报,2016,28(07):1180-1187.
[46]陈乃金,冯志勇,江建慧.用于二维 RCA 跨层数据传输的旁节点无冗余添加算法[J].通信学报,2015,36(04):39-55.
[47]陈乃金,江建慧.一种粗粒度可重构体系结构多目标优化映射算法[J].电子学报, 2015, 43(11): 2151-2160.
[48]陈乃金,冯志勇,江建慧,等.行并行可重构单元阵列流水映射性能评估[J].同济大学学报(自然科学版),2017,45(08):1218-1226.
[49]陈乃金,冯志勇.不跨层行操作并行 RCA 互连时延性能评估[J].天津大学学报(自然科学与工程技术版),2017,50(04):429-436.
[50]韩承浩.可重构系统映射及容错路由算法研究[D].芜湖:安徽工程大学,2023.
[51]KAN Y R ,WU M , ZHANG R Y,et al. A Multi-grained Reconfigurable Accelerator for Approximate Computing[C]//Proceedings of 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI).Limassol, Cyprus,2020:90-95.
[52]FARAHANI A A D ,BEITOLLAHI H ,FATHI M. A Dynamic General Accelerator for Integer and Fixed-Point Processing[J].IEEE Transactions on Very Large Scale Integration (VLSI) Systems,2020,28(12): 2509-2517.
[53]CASTELLANA V G,MINUTOLI M,TUMEO A,et al. Software defined architectures for data analytics[C]//Proceedings of the 24th Asia and South Pacific Design Automation Conference.Tokyo Japan,2019:1-8.
[54]BAREND H, INPYO B, BERNHARD E. Architectures and algorithms for on-device user customization of CNNs[J]. Integration, the VLSI Journal, 2019,67: 121-133.
[55]NGUYEN Q M , SANCHEZ D. Fifer: Practical Acceleration of Irregular Applications on Reconfigurable Architectures[C]//Proceedings of the 54th Annual IEEE/ACM International Symposium on Microarchitecture, 2021:1064-1077.
[56]CHEN Y G , ZHAO Z Y ,JIANG J F. Reducing Memory Access Conflicts with Loop Transformation and Data Reuse on Coarse-grained Reconfigurable Architecture[C]//Proceedings of 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE),Europe:IEEE cs Press,2021:124-129.
[57]MAHDI H, AVIRAL S, SARMA V. Branch-aware loop mapping on CGRA

s[C] //Proceedings of the 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC).San Francisco, CA, USA, 2014:1−6.

[58]RAJENDRANRADHIKA S H, SHRIVASTAVA A, HAMZEH M. Path selection based acceleration of conditionals in CGRAs[C]// Proceedings of the Automation & Test in Europe Conference & Exhibition , 2015:121-126.

[59]DA S S , MARTIN K , COUSSY P, et al. Efficient mapping of CDFG onto coarse-grained reconfigurable array architectures[C]// Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference. IEEE, 2017: 127-132.

[60]LEE J G, LEE J E. Specializing CGRAs for Light-Weight Convolutional Neural Networks[J].IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2021:1-13.

[61]LU Y N, LIU L B, LIU J, et al. A Reconfigurable Branch Predictor for Spatial Computing Architectures[C]//Proceedings of the 2020 4th International Conference on Digital Signal Processing(ICDSP), Chengdu China, 2020:295-299.

[62]MAHDI H, AVIRAL S, SARMA V. REGIMap: Register-Aware Application Mapping on Coarse-Grained Reconfigurable Architectures (CGRAs)[C] //Proceedings of the DAC: Annual ACM/IEEE Design Automation Conference, 2013,1(50):1-10.

[63]YUAN B F, LIU L B, ZHU J F, et al. Compiling Loops with Branches on Static-Scheduling CGRA[C]//Proceedings of 2021 IEEE 4th International Conference on Computer and Communication Engineering Technology(CCET), 2021:290-294.

[64]YUAN B F, ZHU J F, MAN X C, et al. Dynamic-II Pipeline: Compiling Loops with Irregular Branches on Static-Scheduling CGRA[J].IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,2022,41(9): 2929-2942.

[65]ZHAO Z Y, SHENG W Y, WANG Q. Towards Higher Performance and Robust Compilation for CGRA Modulo Scheduling[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(9): 2201−2219.

[66]KOJIMA T, OHWADA A, AMANO H. Mapping-Aware Kernel Partitioning Method for CGRAs Assisted by Deep Learning[J]. IEEE Transactions on Parallel and Distributed Systems(TPDS), 2022, 33(05):1213-1230.

[67]YUAN J, ZHANG X M. Low Power Mapping Optimization of Loops for Dual-Vdd CGRAs[C]//Proceedings of 2017 IEEE 12th International Conference on ASIC, China, 2017:682-685.

[68]GANGHEE L, EDIZ C, OLIVER D. Fault Recovery Time Analysis for Coarse-Grained Reconfigurable Architectures[J]. ACM Transactions on Embedded Computing Systems, 2018, 17(2):1-21.

[69]GU J Y, YIN S Y, LIU L B,et al.Stress-aware loops mapping on CGRAs with dynamic multi-map reconfiguration[J].IEEE Transactions on Parallel and Distributed Systems, 2018, 29(9): 2105-2120.

[70]BRANDALERO M, LIGNATI B N, BECK A C S, et al.Proactive Aging

Mitigation in CGRAs through Utilization-Aware Allocation[C]//Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference,Virtual Event USA,2020:1-6.

攻读学位期间参与的科研项目

[1] 安徽省自然科学基金面上项目，粗粒度行并行和网格可重构结构时域划分映射方法和性能评估研究，编号：1808085MF203。

[2] 安徽工程大学研究生教育创新资金资助项目(实践与创新)，可重构映射阵列与容错算法研究，编号：Y412022086。(已结题)

攻读学位期间发表的学术成果

已发表的论文：

[1]胡宇杨,代广珍,陈乃金,等. 2D Mesh 的多故障容错路由算法[J].天津理工大学学报(自然科学版),已录用.

[2]韩承浩,陈乃金,胡宇杨,等.负载均衡的 2D Mesh 单节点故障容错路由算法[J].长春理工大学学报(自然科学版),2023,46(02):128-135.

[3]李抗,陈乃金,韩承浩,胡宇杨,等. 粗粒度可重构浮点处理单元四则运算性能评估[J].天津理工大学学报(自然科学版),已录用.

已申请的专利：

[1]陈乃金,胡宇杨,韩承浩,李抗,等.一种 2D Mesh 的多故障容错路由方法和路由装置[P]. CN：202311409338.8,发明专利已实审.

[2]陈乃金,李抗,韩承浩,胡宇杨,等.一种粗粒度可重构浮点处理单元和网格型浮点 CGRA[P]. CN：202311538711.X,发明专利已实审.

[3]陈乃金,李抗,韩承浩,胡宇杨,等.粗粒度可重构浮点处理单元和网格型浮点 CGRA[P]. CN：202323112234.9,实用新型专利已授权.

致 谢

时光飞瞬，三年研究之旅即将落幕。回首过往三年，收获甚满，有悉心栽培的师长，有挺身相助的朋友，难以尽述。

首先，有幸拜代广珍副教授、陈乃金教授为师，为我指导科研道路，提供珍贵的建议与启迪性的指导。感谢代广珍老师孜孜不倦的教诲，使我受益匪浅。同时感谢陈老师这三年学术生涯的保驾护航，表达最真诚的感激之情。

其次，幸得李抗、韩承浩、朱耀、罗桢、程飞等同窗好友，大家共同面对挑战的经历让我们建立了深厚的团队友谊，感谢他们在我科研及生活上的帮助。

最后，感谢父亲、母亲的教育之恩，姥姥、姥爷的养育之恩。你们给予了我物质上的滋养以及精神上的奉献，是你们的支持与鼓励，使我度过了这二十年的求学之路，感激之情溢于言表，将永远镌刻在我的心中。

尚德教学　唯实惟新