西安电子科技大学

硕士学位论文

基于NoC的改进XY自适应路由算法的设计

姓名：贾佳

申请学位级别：硕士

专业：计算机应用技术

指导教师：周端

2011-01

摘要

片上网络借鉴了并行计算机互连网络的思想，在芯片上实现处理单元之间的互连，从而在体系结构上解决了总线结构所造成的一系列问题，逐渐成为当前最为活跃的学术课题之一。其中，路由算法作为片上网络的关键技术之一，是一个非常重要的研究方向。

本文对拓扑结构、交换机制、路由算法等片上网络的关键技术进行分析和研究。针对片上网络异步传输的特点，对片上网络的主要组成部分——网络接口及路由节点分别进行设计，并采用硬件描述语言将其实现。在此基础上，利用所设计的IP 模块构建 Mesh 结构的4X4 片上网络验证平台，在该平台上对改进的 XY 自适应路由算法进行功能验证和性能分析。

论文在深入研究多种自适应路由算法的基础上，针对这些算法存在的实现复杂度大、不具有通用性和数据包绕行传输等问题，提出一种改进 XY 自适应路由算法。该算法采用 XY 双向输出竞争策略，充分地利用路由节点的输出端口，从而降低数据传输延迟，提高网络吞吐量，使片上网络性能得到很大的提升。通过基于单个路由节点及片上网络的验证与性能分析，可以看出本文所提出的改进XY自适应路由算法可以获得较好的网络性能，对于片上网络的应用研究具有很好的指导意义。

关键词：片上网络　　自适应路由算法　　仲裁机制

Abstract

Network on Chip(NoC)uses the idea of parallel computer network for reference, realizing communications through the network on the chip,and solving a series of problems caused by bus structure.Thus,it has become one of the hottest academic topics.As one of the key technologies of NoC,routing algorithm is an important research orientation.

In this paper,the key technologies of NoC such as topology structure,information exchange mechanism and routing algorithm are analyzed and researched.Network Interface(NI)and router,the main components of NoC,are designed and implemented using Hardware Description Language(HDL)according to the characteristic of asynchronous transfer.A 4X4 mesh NoC is achieved based on the designed IP modules, on which the performance verifications of improved XY adaptive routing algorithm are carried out.

In this paper,improved XY adaptive routing algorithm is proposed based on the analysis of problems such as complex realizability,poor universality and circuitous transition existed in some adaptive routing algorithms.In improved XY adaptive routing algorithm,the strategy competing for output ports in both X and Y directions is employed to utilize the output ports of the router suficiently,and to reduce the transmission latency and improve the throughput.Experimental results on the basis of a single router and NoC show that the proposed algorithm is very effective in relieving the communication congestion.Therefore,the algorithm is very helpful to the researches and applications of NoC.

Keywords:NoC Adaptive Routing Arbitration Mechanism

第一章绪论

作为史上持续发展最快的技术之一，集成电路 (IC,Integrated Circuit) 的复杂度在过去50多年的时间里以每年50%以上的速度增长川，而单一芯片上晶体管数目更是随着时间以指数的规律增长。据预测，2010 年单一芯片上晶体管数目将达 到 2 2 亿个21,2015年半导体器件的特征尺寸将达到25nm, 时钟频率将达到10GHz 以上3]。这些要求使得以总线结构为主要特征的片上系统 (SoC,System on Chip）越来越难以满足设计要求。于是， 一种替代总线结构的全新体系结构片上网络 (NoC,Network on Chip) 被提出。良好的可扩展性，灵活的通信机制，较低的系统功耗，这些潜在的优势正吸引了国内外越来越多的高校及研究机构加入 NoC 的研究队伍中来。

1.1 课题背景

1965年， Gordon Moore 提出了计算机第一定律——摩尔定律，该定律指出：集成电路上可容纳的晶体管数目，约每隔18个月便会增加一倍，与此同时，性能也将提升一倍。自1958年第一块集成电路推出以来，在过去50多年的时间里，集成电路得到了迅猛的发展，广泛应用于工业、通信、军事、医疗、交通、遥控等各个领域，可以说集成电路的出现使人们的生产生活方式发生了翻天覆地的改变。

20世纪90年代中期，半导体工艺技术的发展，使得在单一集成电路芯片上实现一个复杂的电子系统成为可能，片上系统的概念应运而生。知识产权 (IP, Intellectual Property)核复用和总线结构是SoC 的两个主要特征。IP 核复用使得所设计的单元能够最大限度的被重复使用，从而缩短了设计时间，降低了设计难度，提高了资源利用率；总线结构保证了系统的通信需求，使得多个 IP 核之间可以方便的互连，提供通信能力，简化系统通信设计。

作为 SoC 的主要特征之一，总线结构由于能够提供高性能的互连而得到了广泛的应用。然而，随着深亚微米超大规模集成电路工艺技术的成熟与发展，出现了一些与总线相关的问题：

(1)可扩展性差。有限的地址资源成为扩大电路规模的瓶颈。虽然总线结构可以有效地连接多个通信节点，但是总线地址资源有限，并不能够随着通信节点的增加而无限扩展。此外，随着电路规模越来越大，总线结构中所采用的集中式仲裁方式使得仲裁过程变得十分复杂。

(2)通信效率低。有限的总线带宽和串行访问机制无法满足大量数据通信的

塞，因而降低了通信效率。

(3)能量消耗大。互连线路的功耗是系统功耗的重要组成部分。随着电路规模越来越大，总线上的容性负载显著增大，从而导致传输数据所需功耗也越来越大。此外，同步模式下庞大的时钟网络也将占据系统功耗的大部分。

(4)全局同步约束困难。总线结构要求在整个总线系统中使用同一时钟频率。随着连线长度的增加和时钟频率的提高，信号传播延时可能会超过系统的时钟周期频率。同时，构建全局同步的时钟树也将耗费大量的芯片面积和功耗，合理控制时钟偏斜也变得非常困难。

针对总线技术所造成的这些问题，研究人员提出了许多方法加以缓解，虽然在一定程度上提高了总线结构的性能，但是并没有能够从本质上解决总线结构所固有的缺陷。1999年，片上网络[5][6][7][8I[91 种新的体系结构被提出， NoC 借鉴了并行计算机互连网络的思想，在芯片上实现节点之间的网络互连，从而在体系结构上彻底解决了总线结构所造成的一系列问题。

NOC 具有良好的可扩展性和并行通信能力；基于分组交换方式进行片上通信，可以采用全局异步局部同步 (GALS,Global Asynchronous Local Synchronous) 的通信机制，很好的解决了单一时钟全局同步约束困难的问题；将长的互连线变成交换开关之间互相连接的短连线，有利的控制了系统功耗。由此可以看出， NoC 所具有的这些特点是总线结构所无法比拟的。

1.2 国内外研究现状

2000年3月，法国第六大学的Pierre Guerrier 等人提出了一种可升级、可编程的集成网络，借鉴并行计算中分组交换的思想，实现系统芯片内部的数据通信，并在此基础上建立了周期精确的路由器性能模型110]。2000 年末，瑞典皇家技术学院首次提出了“Network on a chip” 的概念，该概念指出片上网络即是在一块芯片上实现的、由交叉开关网络互连的计算资源、存储资源以及输入输出资源。各资源节点之间通过交叉开关网络将带有路由信息的数据包由源节点发送到目的节点，从而实现资源节点之间的通信。“Network on a chip” 的概念对片上网络理论的提出起到了推动的作用。2001年6月，斯坦福大学 William J.Dally 等人在大规模多处理器和分布式计算网络的基础上提出了使用片上互连网络代替总结结构的思想，片上网络的概念应运而生[1]。

随着技术的发展，越来越多的研究机构认识到了 NoC 所蕴含的巨大潜力，纷纷加入其中并推动其发展，由此，片上网络成为了当前最为活跃的学术课题之一。目前，全世界范围内有超过30个研究单位从事片上网络的研究工作[12],其中具有

荷兰飞利浦研究实验室、意法半导体以及VTT 技术研究中心等工业研究实验室M。巴西，希腊，芬兰，日本，韩国，澳大利亚，以色列，德国等国家也都有相关的机构参与片上网络的研究工作。此外， NoC 也吸引了众多国际会议的目光，国际固态电路会议、设计自动化会议、国际计算机辅助设计会议等国际会议中每年都有相关的论文发表，ACM/IEEE 片上网络国际研讨会举办了完全以NoC 为研究及讨论主题的会议。随着研究工作的深入，研究人员也不再仅仅满足于理论方面的研究，而是更进一步的试图将NoC 运用到实际应用中。欧洲于2008年开始实施一个全新的合作研究项目，该项目希望能够借助NoC 结构的优点开发出一个设计全局异步局部同步架构芯片的流程[13],

2005年起，我国的一些高校及研究机构陆续开展了NoC 相关的研究工作。其中，清华大学、西安电子科技大学、浙江大学、电子科技大学、西安邮电大学、合肥工业大学、上海大学等为我国 NoC 理论研究工作做出了一定的贡献13114115116IL7118119120。然而，从总体上来看，我国在NoC 研究领域尚处于起步阶段，与国外先进水平相比至少落后3~5年。因而，加强对NoC 的深入研究刻不容缓。

1.3 论文结构及内容安排

本文以适用于片上网络自适应路由算法的分析、设计及验证过程为线索进行组织和安排，主要研究内容可以分为三个部分。首先，对改进XY 自适应路由算法的提出和实现进行介绍，这部分的内容将在第三章给出详细说明。其次，由于所提出的路由算法是基于片上网络的，因而需要提供一个片上网络平台来对所提出路由算法的性能进行验证。作为片上网络的重要组成部分，网络接口及路由节点的设计和实现将分别在第四章和第五章进行细致描述。最后，利用所设计网络接口及路由节点模块搭建片上网络对改进 XY 自适应路由算法进行基于单个路由节点和4X4 Mesh 结构片上网络的验证和性能分析，这部分的内容将在第六章具体给出。全文共分为七个章节，各章安排如下：

第一章主要介绍课题研究的背景以及目前国内外有关于片上网络的研究现状，并说明论文结构及内容安排；

第二章首先阐述片上网络与大规模并行计算机网络之间的差异，然后对拓扑结构、交换机制、路由算法、死锁的产生与避免等几个片上网络的关键技术进行详细的描述。

第三章首先说明改进XY 自适应路由算法提出的原因，然后对改进XY 自适应路由算法的实现以及采用该算法后对仲裁机制所需要做出的相应调整进行系统的描述。

所采用的两种关键技术——异步FIFO 和源节点路由进行细致的说明。

第五章首先给出路由节点的系统架构，然后分模块对输入端口、交叉开关以及输出端口的实现过程进行具体的介绍。

第六章在在基于前三章的基础上完成片上网络验证平台的构建工作，并在该平台上对改进XY 自适应路由算法进行验证及性能分析。

第七章对本次论文工作做出总结，分析设计中仍然存在的不足，对接下来的研究工作提出意见和建议。

第二章片上网络的关键技术

作为一个由大规模并行计算机网络衍生而来的概念，片上网络与大规模并行计算机网络存在着许多相似的地方，但是两者之间依然存在着许多不同的地方。本章首先阐述片上网络与大规模并行计算机网络之间的差异性，然后分别对拓扑结构、交换机制、路由算法、死锁的产生与避免等几个与片上网络研究相关的概念进行详细的描述。

2.1 片上网络与大规模并行计算机网络

自1999年片上网络的概念被提出以来，各方面的研究工作都在如火如荼的进行着。片上网络代替总线结构，应用于单一芯片内实现处理单元之间的互连。片上网络的概念来源于大规模并行计算机网络，因而两者之间有许多相似的地方，特别是在路由算法方面，几乎所有适用于大规模并行计算机网络中的路由算法都可以在片上网络中实现。但是两者间也存在许多不同的地方，主要表现在：

(1)硬件资源的使用不同。在大规模并行计算机网络中，通常使用专门的芯片和复杂的路由算法对数据信息进行路由、交换、仲裁等处理，硬件资源的使用不受限制；而在片上网络中，由于芯片面积的限制，大都使用简单的逻辑单元和路由算法，硬件资源的使用受到限制。

(2)数据位宽不同。在大规模并行计算机网络中，节点之间的连线在芯片外部，信号干扰较大，而且受到芯片封装的约束，　一般使用串行或较小的数据位宽；而在片上网络中，节点之间的连线在芯片内部，信号干扰较小，而且不受芯片封装的约束，可以使用较大的数据位宽。

(3)时钟方案不同。在大规模并行计算机网络中，节点之间的工作是完全异步，网络中节点之间的通信一般是使用握手的方式进行数据的同步；而在片上网络中，节点之间的工作一般是采用 GALS　的方式，片上网络中节点之间时钟域的转换使用异步缓存(FIFO,First　In　First　Out)单元或时钟暂停模块的方式来实现的。

(4)网络协议不同。在大规模并行计算机网络中，存在有标准的网络传输协议，并且在整个大规模并行计算机网络系统中有专门的协议处理机来处理复杂的网络协议以保证数据包在网络传输的过程中能够完整、无损、顺序的由源节点传送到目的节点；而在片上网络中，到目前为止尚没有一个统一标准的网络传输协议，并且在片上网络中也没有设置专门的协议处理机，所有的网络协议都由硬件来进行处理，因而要求片上网络的网络协议不能太复杂，以避免耗费过多的芯片

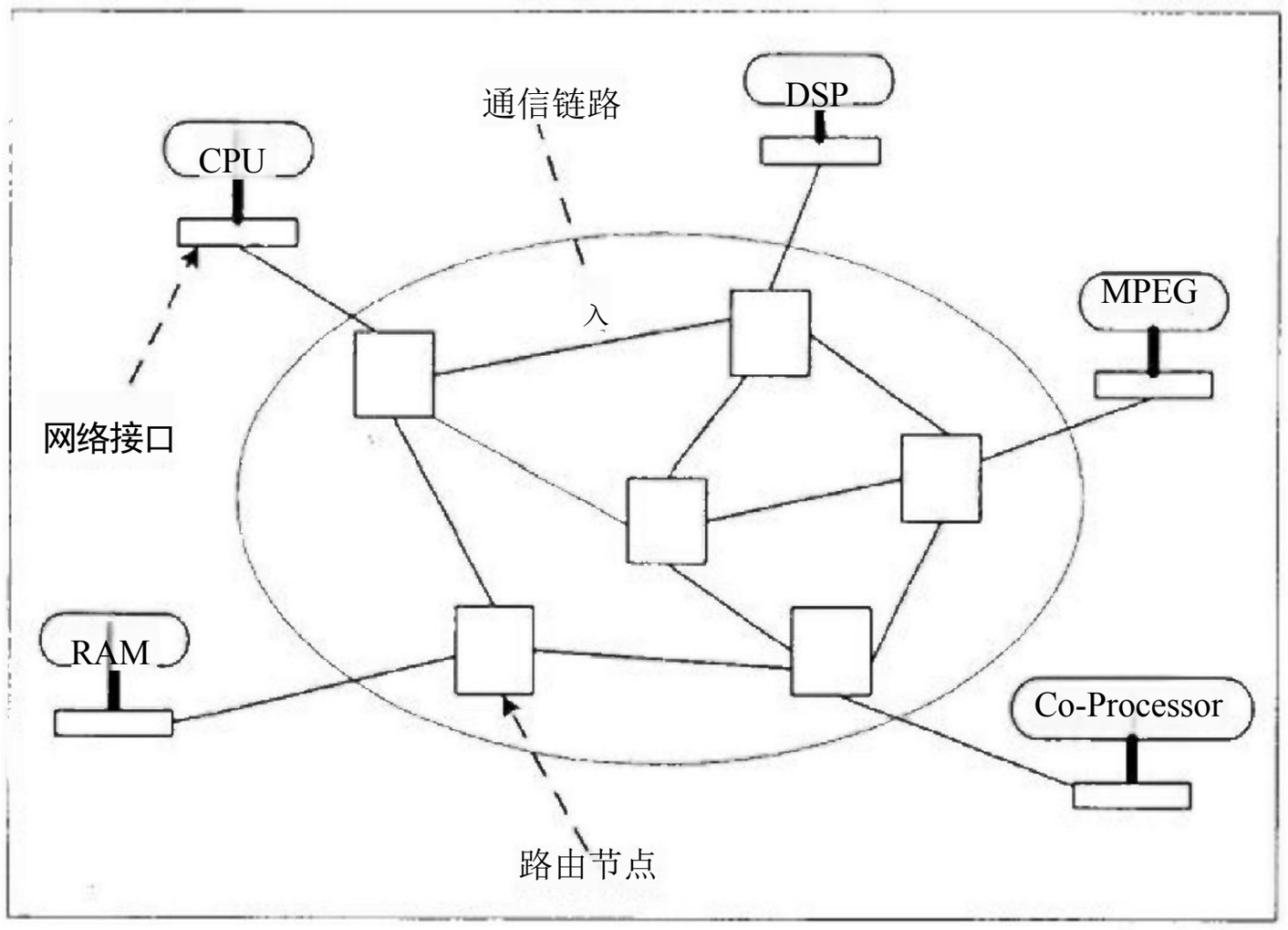


图2.1片上网络结构

片上网络与大规模并行计算机网络之间所存在的上述差异是由片上网络自身所蕴含的特点所决定的。图2.1所示为片上网络的基本结构示意图，由图中可以看出，片上网络一般由网络接口 (NI,Network Interface)、路由节点 (Router) 以及通信链路 (Link) 三部分所组成S 。 网络接口作为处理单元与片上网络的桥梁，实现处理单元与片上网络之间的连接，其中处理单元可以是通用处理器 (CPU)、 数字信号处理器 (DSP)、 存储资源(如 RAM)、 协处理器 (Co-Processor)、 音频视频处理单元(如 MPEG) 以及其他一些具有一定功能的IP 核等；路由节点按照一定的拓扑结构进行连接，从而实现数据包在整个片上网络中的传输；通信链路实现路由节点与网络接口、网络接口与处理单元以及相邻路由节点之间的连接。

2.2 拓扑结构

拓扑结构体现了片上网络中路由节点的分布和连接。拓扑结构的选择直接决定了系统性能的优劣及芯片面积的大小。衡量拓扑结构的标准除了该拓扑结构对路由成本和系统性能的影响之外，还需要考虑通信模式的嵌入属性。 一般来说，包括以下几个特征参数[20]:

- 网络直径：片上网络中连接任意两个节点所需要的最大节点个数。
- 节点连接度：片上网络中与每个节点直接相邻的节点个数。
- 网络带宽：片上网络中在单位时间内能实现的最大通信流量。
- 通信延时：数据包在片上网络传输的过程中从源节点到达目的节点所需要的传输时间。

网 终 天 此 量 ， 单位时间内通过片上网络数据包的个数

络拓扑结构和不规则型网络拓扑结构两类。

2.2.1 规则型网络拓扑结构

在规则型网络拓扑结构中，路由节点具有相同的功能，规模及面积，因而在整个片上网络中，路由节点的分布和布局具有一定的规则性。常见的规则型网络拓扑包括二维网格 (Mesh) 拓扑结构、二维环绕网格 (Torus) 拓扑结构、八角形 (Octagon) 拓扑结构、蝶形 (Butterfly) 拓扑结构、胖树 (Fat Tree) 拓扑结构等[21]。

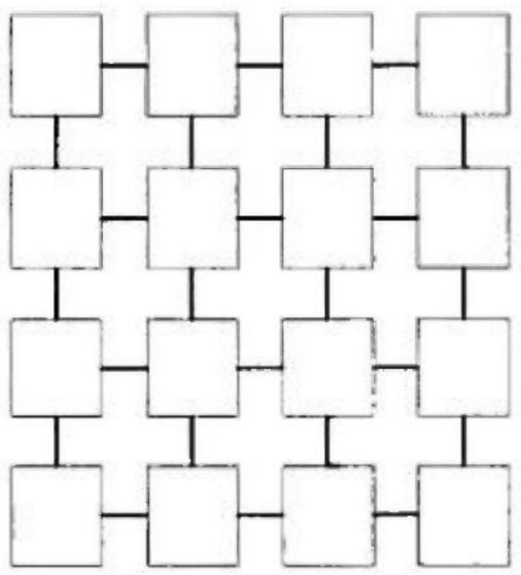


图2.2 二维网格 (Mesh) 拓扑结构

二维网格拓扑结构。如图2.2所示，Mesh 拓扑结构是片上网络研究最早也是最常见的一种拓扑结构。在 Mesh 拓扑结构中，除边缘节点与周围两个或三个相邻节点相连之外，其余中心节点都与周围四个相邻节点相连。Mesh 拓扑结构十分简单，具有良好的规则性和可扩展性，易于在芯片内部实现，因而在目前大部分有关于片上网络的研究工作中，都广泛的采用Mesh 拓扑结构。当然， Mesh 拓扑结构也存在着不足之处。相对于其他直接型网络拓扑而言，Mesh 拓扑结构的网络直径较大。当片上网络规模需要扩大时，系统功耗会随着网络直径的增加而变得越来越大。

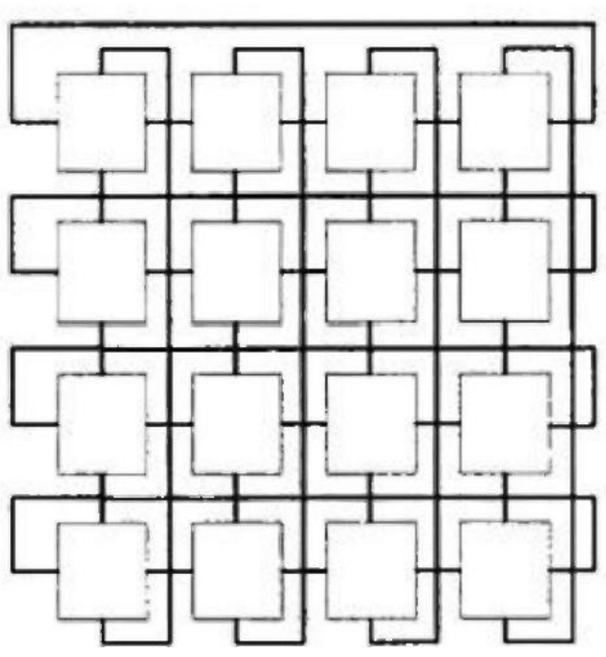


图2.3 二维环绕网格 (Torus) 拓扑结构

二维环绕网格拓扑结构[22]。如图2.3所示，Torus 拓扑结构可以看作是对Mesh

与物理位置上相邻的节点相连之外，还与其逻辑位置上相邻的节点相连，从而使得在整个片上网络中的任一节点都与“相邻”的四个节点实现连接。Torus 拓扑结构与 Mesh 拓扑结构相比，加强了边缘节点的通信能力，从而有效的提高了可用带宽。然而，额外增加的连线同时也为后期布线工作增添了难度。此外，在 Torus 拓扑结构中发生死锁的可能性也相应的有所提高

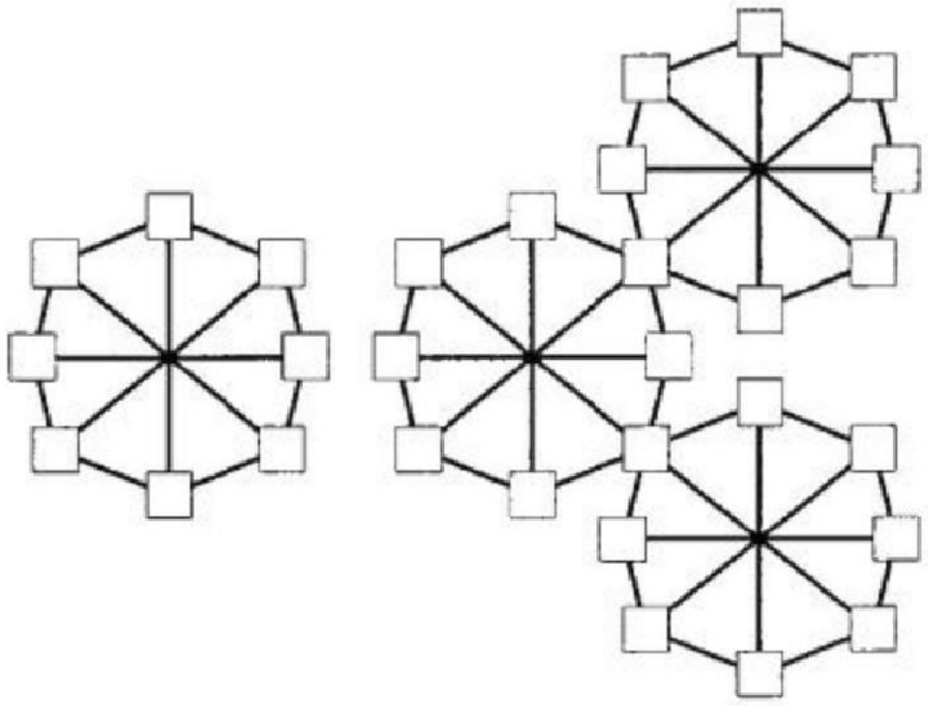
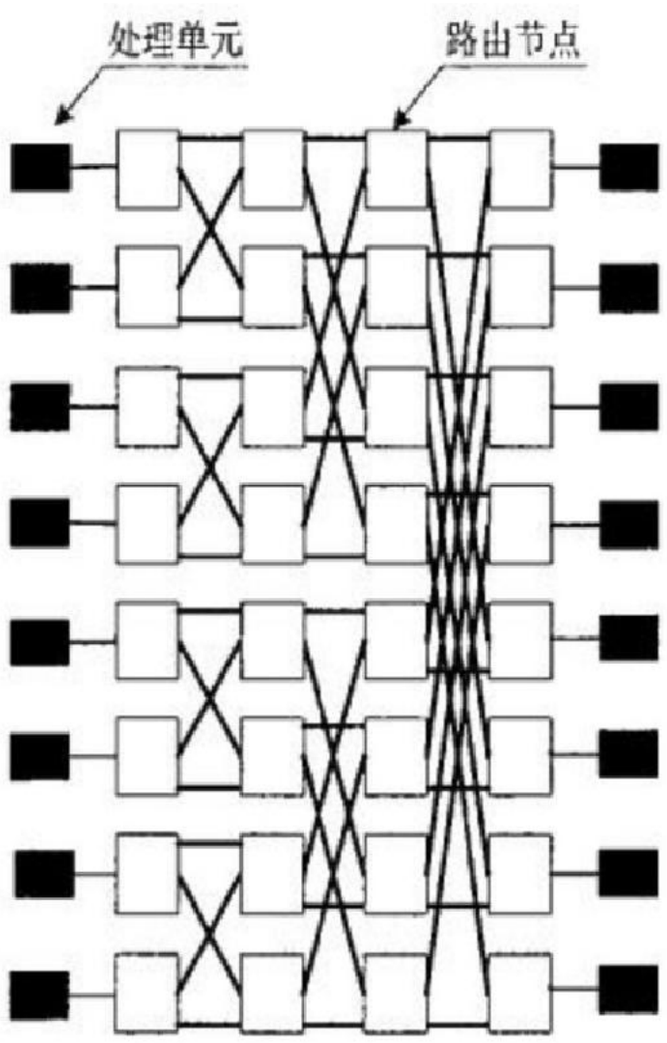


图2.4 八角形 (Octagon) 拓扑结构

八角形拓扑结构1231。如图2.4所示，在 Octagon拓扑结构中，每一个节点都与相邻的两个节点以及对角线节点直接相连，任意两个节点之间的通信最多只需要通过一个节点即可完成。此外，每一个节点又可以单另扩展成为一个基本 Octagon 拓扑结构的网络并且在此基础上进行无限的扩展。然而，采用Octagon 拓扑结构增加了后期布线的复杂性，并且随着片上网络规模的不断扩展，连接两个 Octagon 拓扑结构网络的中间扩展节点很容易变成整个片上网络通信过程中的瓶颈，当该扩展节点失效时，整个片上网络将会被分割为失去联系的两个部分，从而无法实现节点之间的正常通信。



蝶形拓扑结构。如图2.5所示，在Butterfly 拓扑结构中，任意一对源处理单元与目的处理单元之间都存在着一条专门的路径，并且数据包在处理单元之间的传输延迟相同，传输延迟由路由节点所组成开关结构中的中间阶数所决定。专用传输路径的使用，使得不同的处理单元之间数据包的传输相互不会产生干扰，因而有效的提高了整个片上网络的数据带宽。然而，过多的连线不仅为后期布线工作增添了困难，而且大大增加了系统的成本和面积。

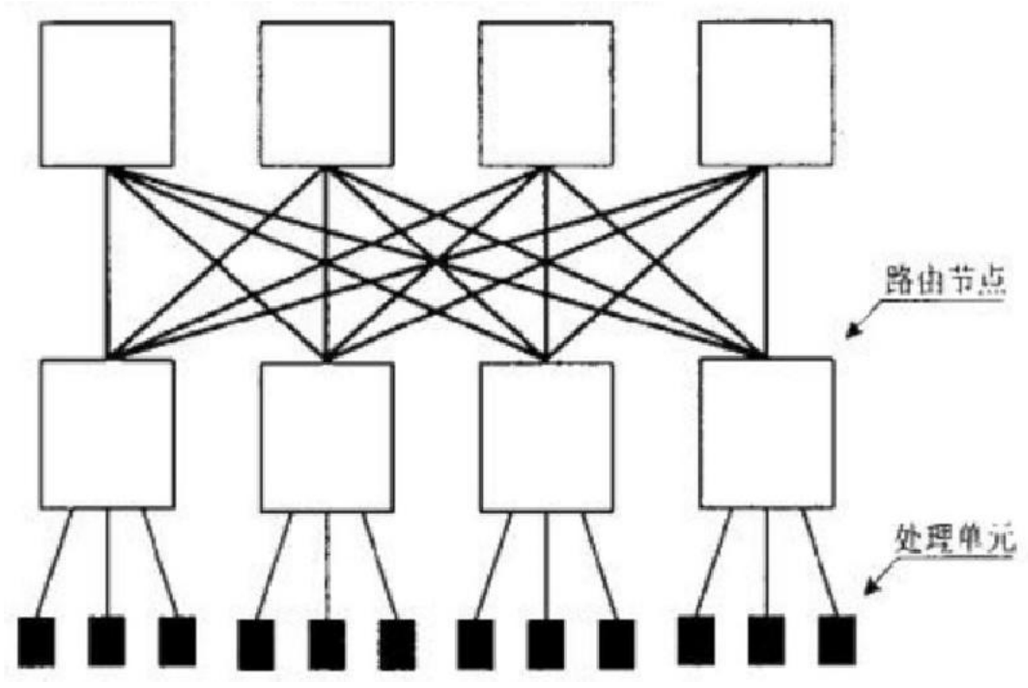


图2.6 胖树 (Fat Tree) 拓扑

胖树拓扑结构。如图2.6所示，在 Fat Tree 拓扑结构中，叶子节点为处理单元，非叶子节点为路由节点，每一个父亲节点在胖树结构中的每一层都被复制四次，通过增加冗余度的方法来提高数据包传输的可靠性，实现片上网络的可扩展性。Fat Tree 拓扑结构所存在的问题就是数据包的交换机制过于复杂，布线的复杂度很大，并且会大大增加了整个芯片的面积。

2.2.2 不规则型网络拓扑结构

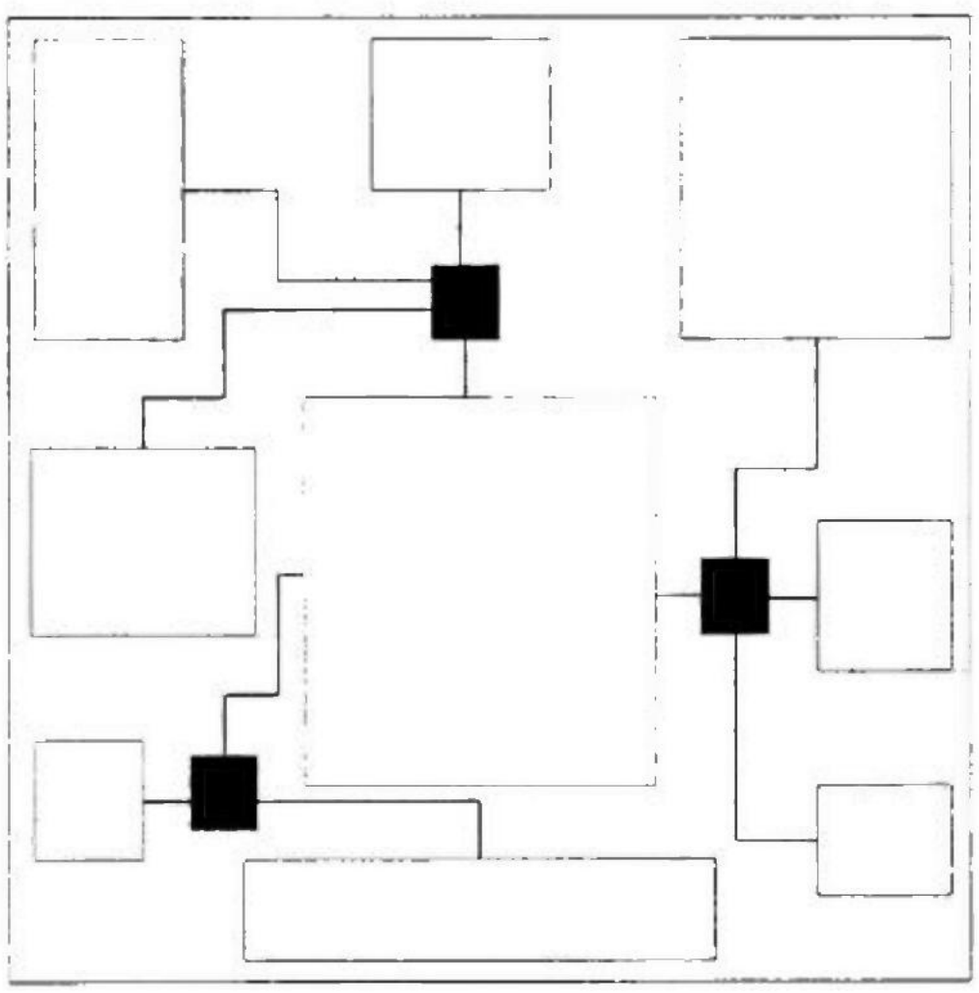


图2.7 不规则型网络拓扑结构

在实际的应用中，由于系统性能要求的不同，可能使得路由节点在设计的过

情况下，无法采用规则型网络拓扑结构对路由节点进行连接，如图2.7 所示，可以针对具体应用，建立专用的不规则型网络拓扑结构24]。

本文的重点主要集中在改进 XY 自适应路由算法的研究，为了不增加片上网络设计的复杂度，减少其他因素对系统性能可能造成的影响，故而采用规则型网络拓扑结构中的 Mesh 拓扑结构。

2.3 交换机制

电路交换和包交换是两种最为常见的交换机制。

所谓电路交换，就是在通信节点之间建立一条专用的物理链路，在整个通信过程中，该物理链路不能为其他节点所用。在电路交换中，专用物理链路分配有固定的带宽，因而使得服务质量得到了保证。然而，由于物理链路的专用性，使得链路的利用率不高，网络吞吐量较低。

与电路交换不同，包交换在节点通信之前并不建立专用的物理链路，而是将数据和控制信息以数据包的形式封装起来，位于网络中的路由节点读取数据包中的目的节点信息，按照一定的路由算法将数据包传输给下一路由节点，直至最终数据包送达目的节点。目前，片上网络大都采用包交换的交换机制，常见的包交换机制又可以分为存储转发交换、偏转路由交换、虚拟直通交换以及虫孔交换。下面分别对这四种交换机制进行详细的介绍。

2.3.1 存储转发交换

存储转发交换将整个消息分为若干个数据包，每个数据包在传送的过程中，首先存储在路由节点的缓存中，当路由方向确定之后，才传送到下一个路由节点。存储转发交换实现起来比较简单，但是当传送的数据包比较大时，这种交换机制需要在每一个路由节点中设计大量的存储空间，而且数据包在每个中间路由节点由于数据的存储会引入额外的传输延迟。大规模并行计算机网络中普遍采用这种交换机制，但是在片上网络中，由于有限的存储资源，存储转发交换显得不切实际。

2.3.2 偏转路由交换

偏转路由交换也被称为烫手山芋路由交换，在这种交换机制中，当路由节点接收到数据包时，如同捧到了一个“烫手山芋”一样，并不缓存数据包，而是将数据包立即通过空闲的输出端口转发到下一个路由节点，因而，所有的数据包在

缓存当前数据以外，这种交换机制不需要任何多余的缓存空间，从而节省了大量的系统资源，有效的减小了系统芯片的面积。但是由于这种交换机制的极端性，使得死锁产生的可能性较高。此外，数据包并不一定总是按照最短路径进行传输，因而由于数据包在片上网络中的绕行可能造成了过多不必要的传输延迟。

2.3.3 虫孔交换

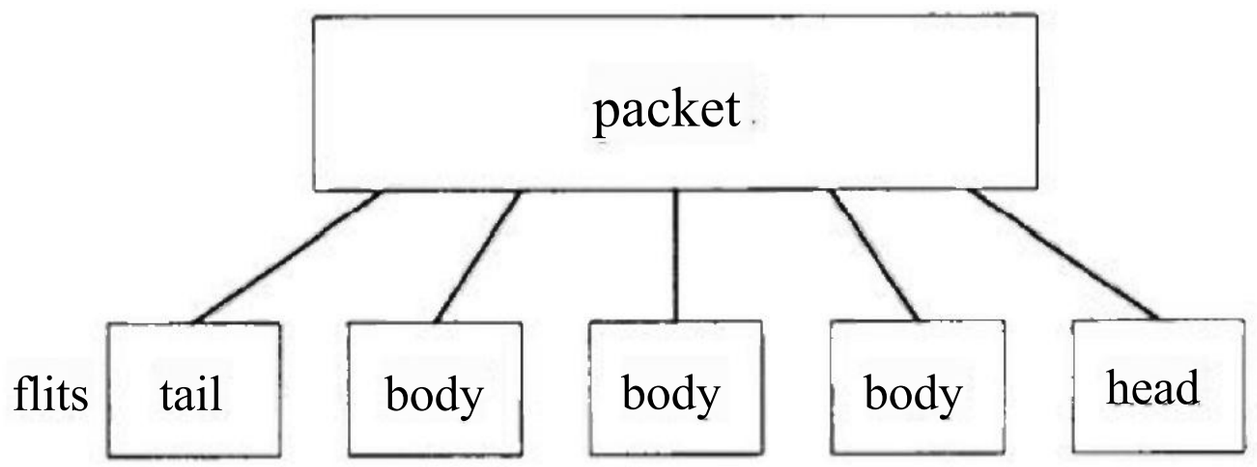


图2.8 数据包划分为数据微片示意图

在虫孔交换中，如图2.8所示，数据包被进一步的分解为若干个被称之为数据微 片 (flit) 的流量控制单位，从而使得数据包传输过程流水化，即路由节点通过读取位于头微片中的路由控制信息进行路径的选择，数据微片沿着头微片的路径以流水的方式在网络中向前“蠕动”,每一个数据微片相当于虫的一个节，数据包以微片为单位顺序地向前“蠕动”。当路由节点发生阻塞时，路由节点只需要缓存当前位于路由节点中的一个或几个微片，不需要缓存整个数据包，因而大大降低了对缓存资源的要求。

与上述两种交换机制相比，虫孔交换具有如下两个优点：

1. 虫孔交换对路由节点缓存资源的要求不高。与存储转发交换需要在路由节点中缓存整个数据包相比，虫孔交换只需要在路由节点中缓存一个或几个数据微片，因而只需要很少的缓存资源，从而有效的减少了片上网络芯片的面积。

2. 网络延迟对路径长度相对的不那么敏感。虫孔交换的网络延迟主要由数据包发送时间和路由节点寻径时间所组成。由于实际应用中数据包的长度较大，数据包的发送时间远大于路由节点的寻径时间，因而延迟主要由数据包发送时间所决定。

然而，虫孔交换也存在着一些问题，例如，当路由节点发生阻塞时，数据微片缓存在当前所处路出节点中，由于缓存资源的有限，可能使得后继数据包的传输也被阻塞，因而进一步造成整个网络拥塞的加剧，从而降低了网络性能。此外，数据包在传输的过程中，头微片所经过的输出端口被“标记”下来，直至整个数据包完全通过该输出端口后才将其完全的释放，在输出端口被“标记”的过程中，不能为其他数据包所使用，因而降低了输出端口的利用率，网络吞吐量较低。

2.3.4 虚拟直通交换

虚拟直通交换是存储转发交换和虫孔交换的一种折中交换方式。在虚拟直通交换中，数据包同样被进一步的划分为数据微片。路由节点接收到数据包头微片后，不必等待整个数据包全部到达，就可以根据数据包头微片中所包含的路由信息立即开始进行输出端口的判断，待头微片接收到来自相应输出端口的响应后，其余的数据微片可以不需要暂存在路由节点，而是直接沿着头微片的传输路径被发送至下一级的路由节点。

与存储转发交换相比，虚拟直通交换将数据包进一步的划分为数据微片，采用流水线的方式对其进行传输，从而有效的减少了数据包在通过中间路由节点时所造成的数据传输延迟。此外，在虚拟直通交换中， 一个数据包在被当前路由节点完全接收之前就可以被传送到下一路由节点，降低了对路由节点缓存资源的要求。

在非阻塞的情况下，虚拟直通交换与虫孔交换是完全一样的，两者之间的差别就在于发生阻塞时对于数据包的处理。如图2.9 (a) 所示，发生阻塞时，虚拟直通交换将整个数据包缓存在发生阻塞的路由节点中；如图2.9 (b) 所示，发生阻塞时，虫孔交换将微片分散的缓存在当前所处路由节点中。虽然与虫孔交换相比，虚拟直通交换可能需要使用更多的缓存资源，但是虫孔交换需要额外的功能模块对所占用路径进行维护和管理，因而实现起来过于复杂。

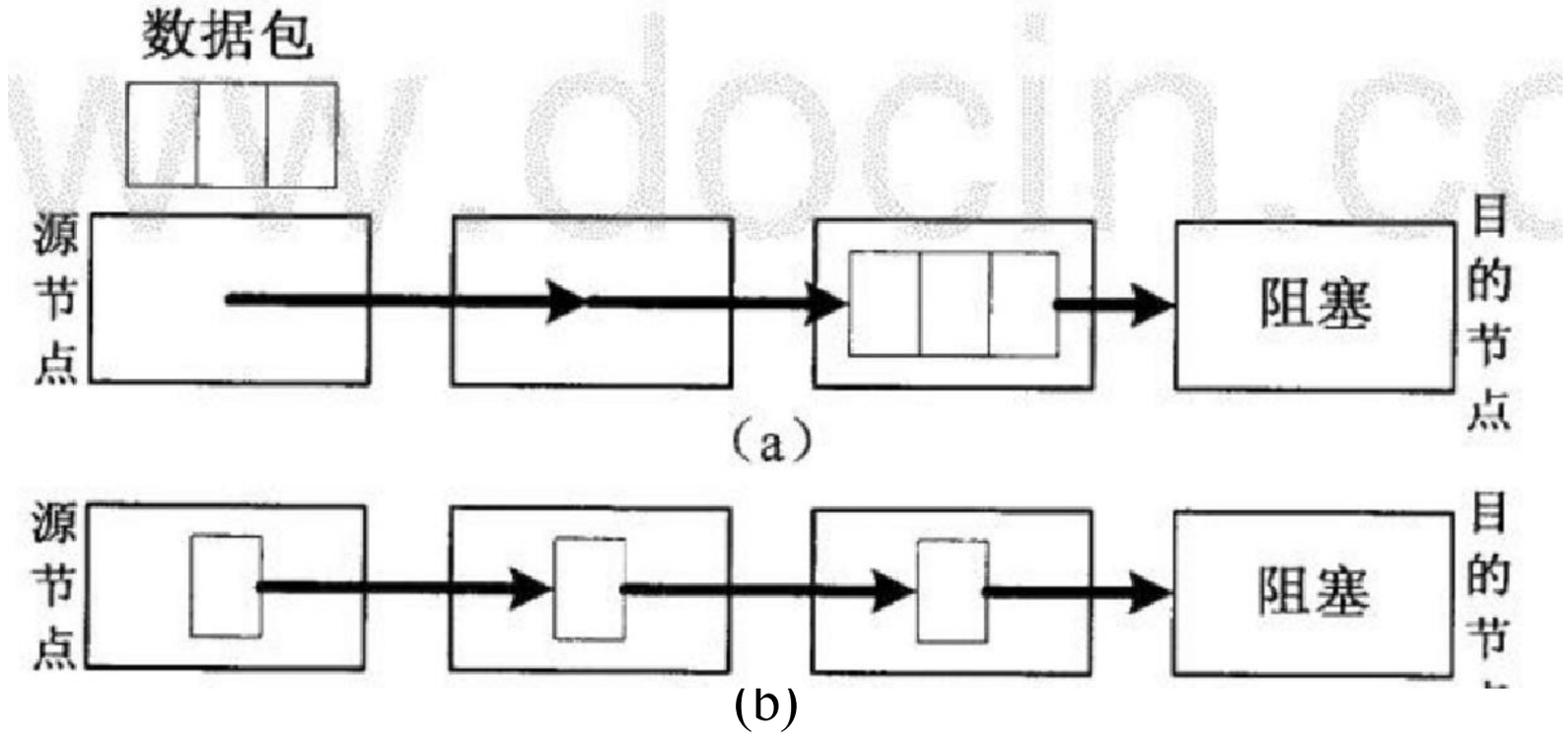


图2.9 虚拟直通交换 (a) 与虫孔交换 (b) 发生阻塞情况时的比较图

通过对上述四种交换机制的叙述和比较，在本文中选择采用一种折中的交换机制——虚拟直通交换。该交换方式一方面继承了存储转发交换的特点，交换机制灵活，实现简单；另一方面，继承了虫孔交换将数据包进一步划分为微片的策略，降低了对路由节点缓存资源的要求。

2.4 路由算法

路由算法决定了数据包在片上网络中传输时所要经历的路径。有效的路出算法可以使得数据包能够在最短的时间内以最短的路径由源节点到达目的节点，对系统性能起到至关重要的作用。

按照是否根据当前网络状况做出路由路径的调整，路由算法可以分为确定路由算法和自适应路由算法1251126]。下面对这两种路由算法进行详细的介绍。

2.4.1 确定路由算法

确定路由算法是一种常见的路由算法，在这种路由算法中，路由路径不会根据当前网络状况的变化做出改变，而只是和源地址与目的地址有关，源地址和目的地址确定了，路由路径也就确定了。在确定路由算法中，维序路由算法因为其简单易实现的路由逻辑而被广泛使用。

在维序路由算法中，数据包一次只在一个维度上进行路径路由，当在该维度上到达确定坐标后，按照由低维到高维的顺序再在下一个维度上进行路径路由，直至到达目的地址为止。出于维序路由算法严格按照单调的位数变化顺序在片上网络中进行路径路由，因而是没有死锁的。常见的维序路由算法包括二维 Mesh 拓扑结构下的 XY 路由算法以及超立方体拓扑结构下的 E-cube 路由算法27。

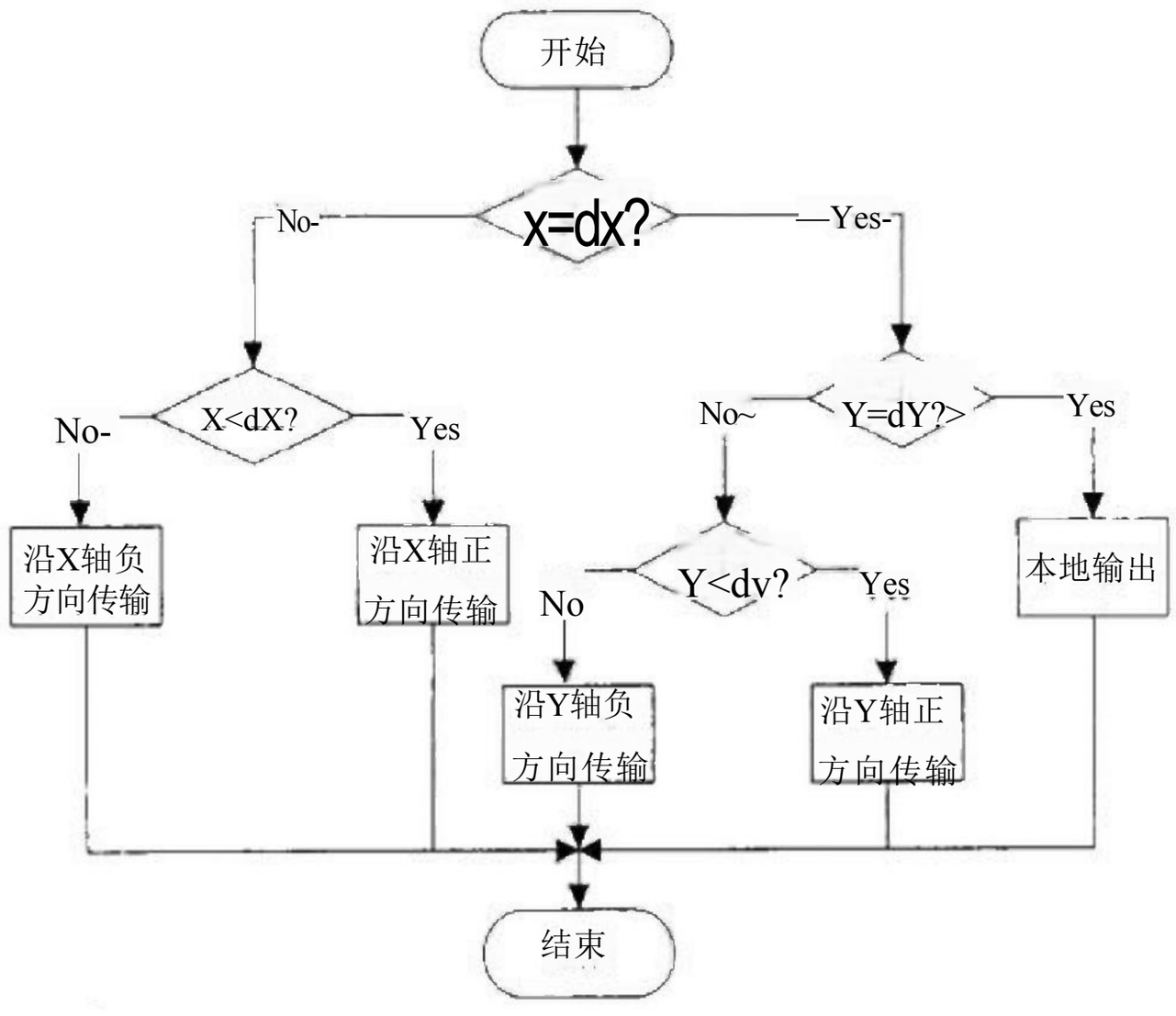


图2.10 XY 确定路由算法

二维 Mesh 拓扑结构下的 XY 确定路由算法是最简单的一种路由算法[28]。该算法具体实现过程如图2.10所示。当路由节点接收到一个数据包时，将该路由节点

时，数据包沿Y 轴正方向传输；当X=dX 且 Y>dY 时，数据包沿Y 轴负方向传输；当 X=dX 且 Y=dY 时，数据包到达目的节点，路由节点将数据包传送给与之相连接的处理单元。

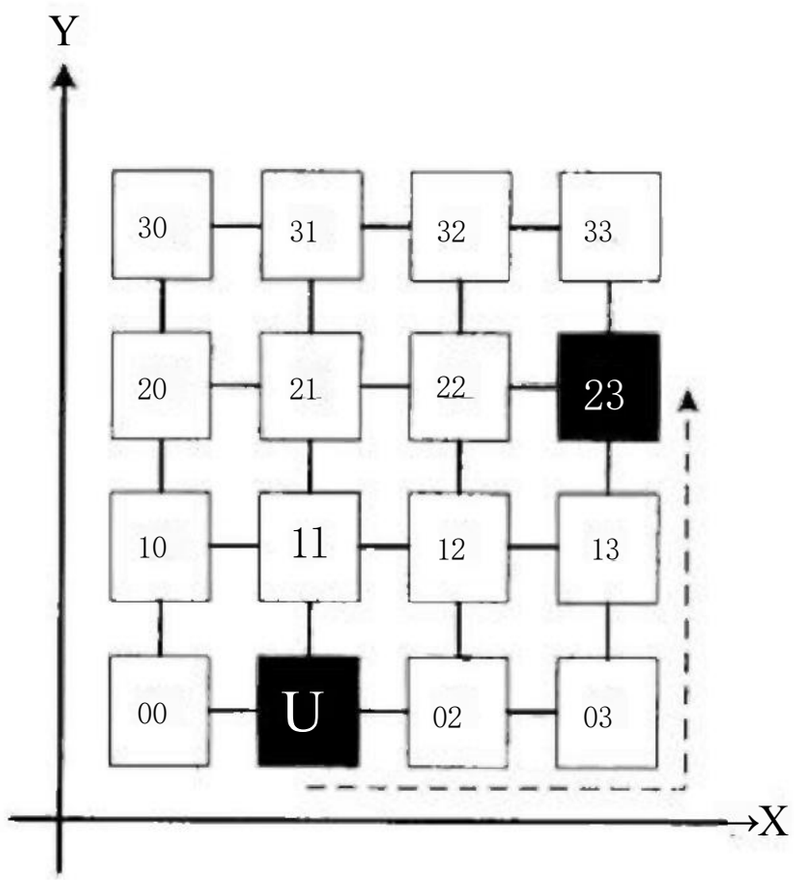


图2.11 XY 确定路由算法应用实例

如图2.11所示，数据包由源节点01发送到目的节点23,按照 XY 确定路由算法，数据包在片上网络的传输路径为：

(01) —>(02)—>(03)—>(13)—>(23)

确定路由算法实现简单，在网络低拥塞情况下能够获得较低的通信延迟，但是由于不能根据当前网络状况对路由路径进行调整，当网络出现拥塞的情况时数据包传输延迟增大，网络吞吐量降低，从而使得整个片上网络的性能迅速降低。

2.4.2 自适应路由算法

在自适应路由算法中，路由路径不仅与源地址和目的地址有关，还与当前网络状况有关。当源地址和目的地址确定时，根据当前网络状况的不同，路由路径可能是不同的。自适应路由算法根据当前网络状况的不同对路由路径做出调整有效的避免了片上网络拥塞的发生，使网络性能一直维持在一个相对稳定的状态。自适应路由算法所存在的主要不足之处就在于当网络处于低拥塞的情况下，复杂的路由逻辑会造成较大的系统开销。

2.4.3 带有一定自适应性的维序路由算法

带有一定自适应性的维序路由算法是对确定路由算法和自适应路由算法的一种折中。在一般的维序路由算法中，数据包每次只能在一个维度上进行路径路由，

当数据包在某一维度进行路径路由的过程中发生阻塞时，可以提前在下一维度进行相应的路径路由，　一旦原维度阻塞解除，再将数据包转回到原维度上继续进行路径路由，直至在该维度上到达恰当的坐标后，再按照由低维到高维的顺序开始在下一维度上的路径路由。带有一定自适应性的维序路由算法在一定程度上缓解了网络拥塞，实现简单，并且能够有效的避免死锁的产生。

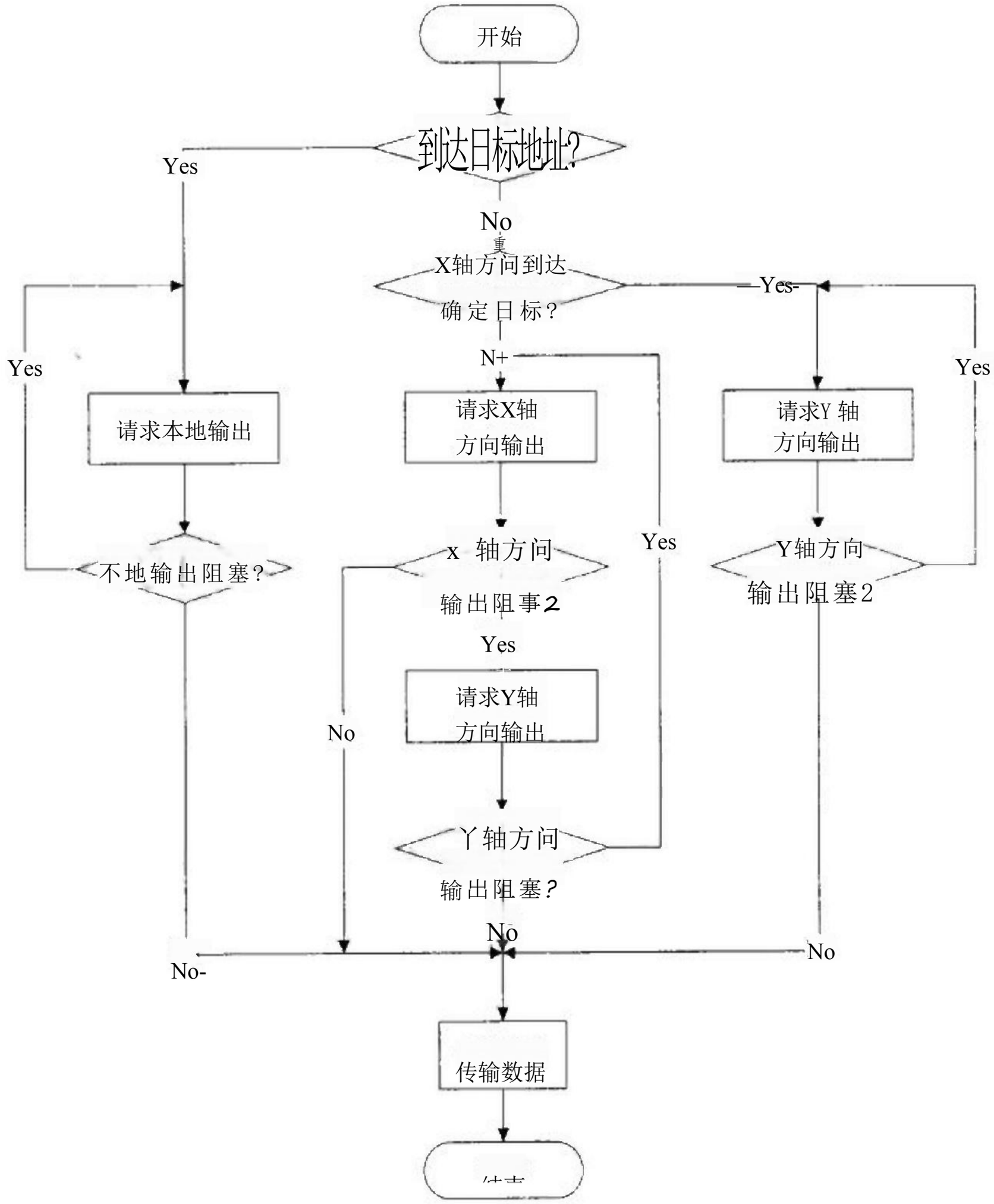


图2．12 简 单XY 自适应路由算法

简单 XY 自适应路由算法是一种常见的带有一定自适应性的维序路由算法，该算法在 XY 确定路由算法的基础上做出了一定的改进，使XY 确定路由算法能够根据网络状况进行路由路径的调整。其具体实现流程如图2.12所示。当数据包到达当前路由节点时，对数据包中所包含的路由信息进行分析以判断数据包的输出方向。当前路由节点即为目的节点时，数据包等待直至通过本地输出端口发送至本地处理单元；当前路由节点不是目的节点时，如果数据包在X 轴方向仍未到达

否则转而向相应Y 轴方向的输出端口发送输出请求，如果得到响应，通过Y 轴方向的输出端口传送至下一级路由节点。如果数据包在X 轴方向已经到达确定坐标，数据包等待直至通过相应 Y 轴方向的输出端口发送至下一级路由节点。简单 XY 自适应路由算法不仅可以有效的避免死锁的发生，而且在一定程度上缓解网络的拥塞情况从而使网络性能保持在一个相对稳定的状态。

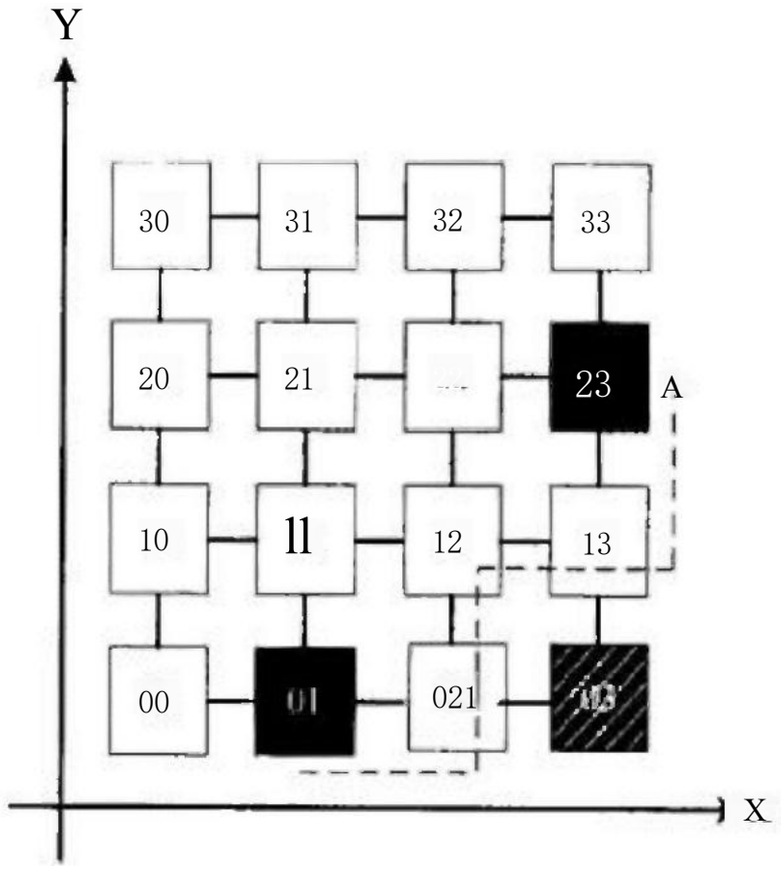


图2.13 简单XY 自适应路由算法应用实例

如图2.13 所示，数据包由源节点01发送到目的节点23,在沿X 轴方向传输的过程中，节点03发生了阻塞，按照简单XY 自适应路由算法，数据包在片上网络的传输路径为：

(01) —>(02)—>(12)—>(13)—>(23)

在片上网络中，由于芯片面积和资源的限制，通常使用简单的确定路由算法或带有一定自适应性的维序路由算法。

2.5 死锁的产生和避免

所谓死锁是指多个数据包在片上网络传输的过程中，由于争夺网络资源而造成的循环等待的现象。如图2.14所示，位于节点1中的数据包A 请求节点2中的缓存资源，但节点2中的缓存资源被数据包B 所占用；与此同时，位于节点2中的数据包B 请求节点1中的缓存资源，但节点1中的缓存资源被数据包A 所占用。数据包A 和数据包B 相互等待彼此所占用的缓存资源，因而造成了对缓存资源的依赖环，从而产生了死锁。

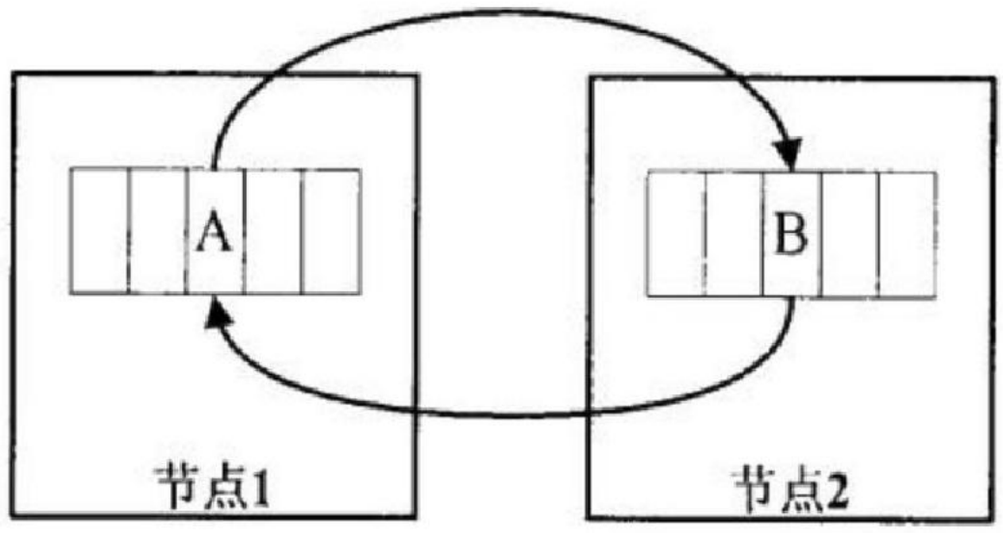


图2.14 死锁

目前，在片上网络中避免死锁的方法主要有两种： 一种是采用维序路由算法另一种是使用虚拟通道291301。维序路由算法在前文中已经有所介绍，这里主要对虚拟通道做以详细介绍。

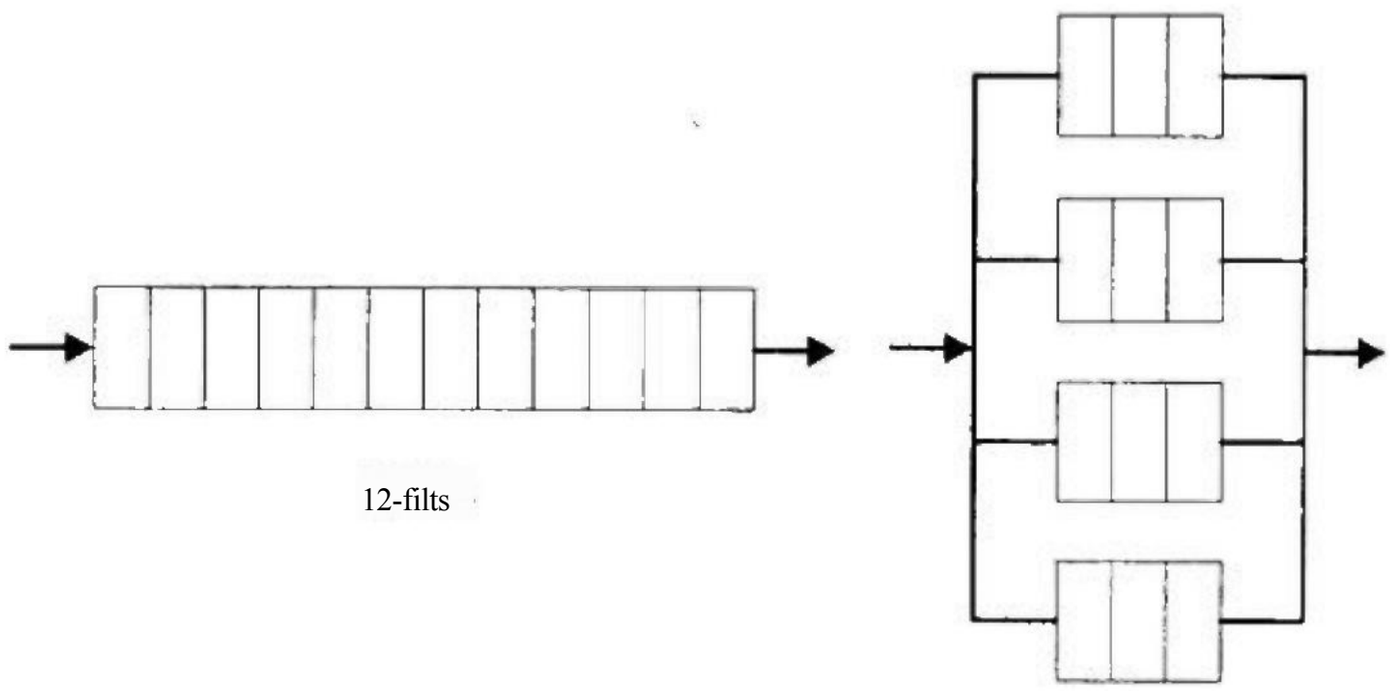


图2.15 虚拟通道

所谓虚拟通道就是指在路由节点中若干个通过时分复用的方式共享一条物理信道的缓存。如图2.15所示， 一个可以容纳12个数据微片的缓存被划分为四个容量为3个数据微片的虚拟通道，并将划分后的四个虚拟通道进行并行的连接，在保证存储容量不变的情况下通过结构的改变来实现四个虚拟通道对原有物理信道的时分复用。

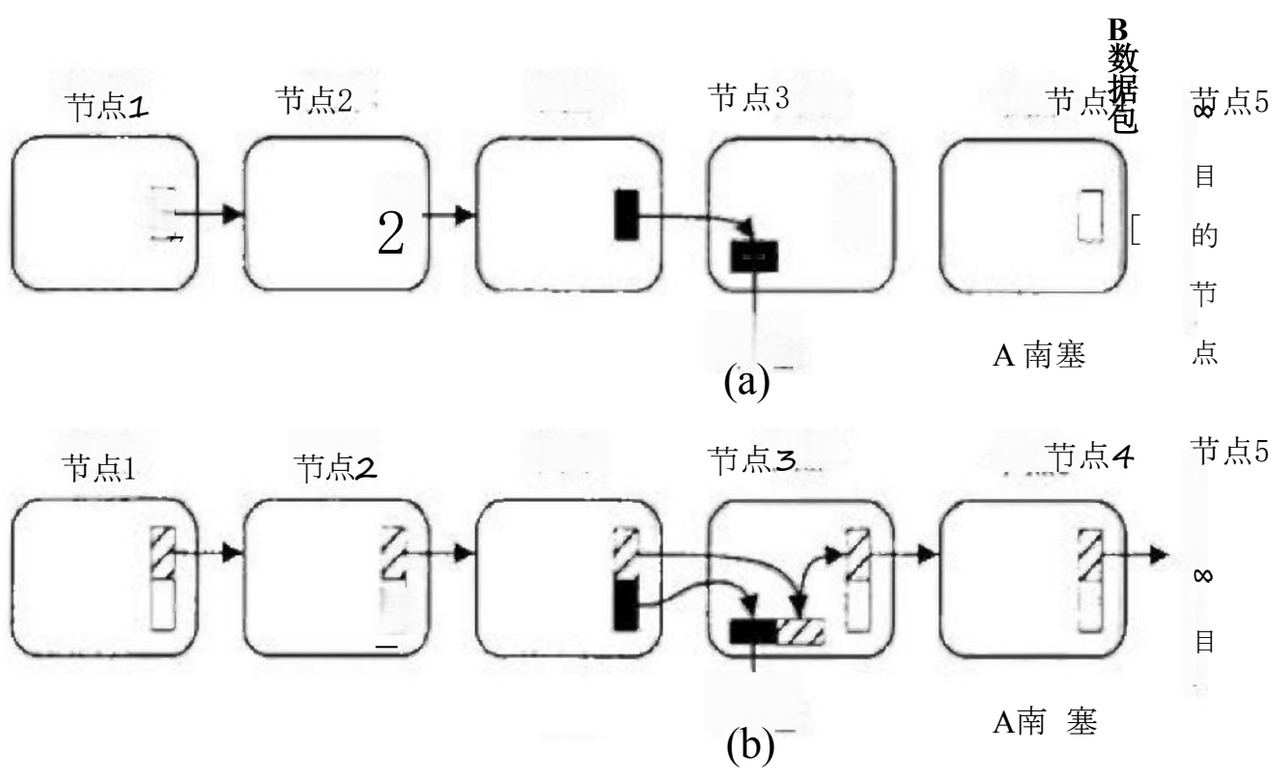


图2.16所示为使用虚拟通道技术前后的比较。在图2.16中数据包A 被路由节点4所阻塞，其随后的数据微片被缓存在当前所处路由节点3中。数据包B 希望通过路出节点3到达目的节点。在图（a） 中，当未引入虚拟通道技术时，数据包B 因为数据包A 占用了数据包B 前行方向路由节点中的缓存资源而被阻塞；在图(b) 中，当引入虚拟通道技术时，数据包B 可以绕过数据包A 所占用的虚拟通道，通过其他虚拟通道最终到达自己目的节点。由此可以看出，虚拟通道技术的引入，不仅提高了数据链路的利用率和网络吞吐量，而且有效的避免了死锁的产生，因而在本文的设计中，采用虚拟通道技术以避免死锁。

2.6 本章小结

本章首先将片上网络及大规模并行计算机网络进行了比较，阐述了两者之间的相同点及不同点。然后对一些常见的拓扑结构、交换机制、路由算法以及死锁避免的方法进行了细致的描述和分析，在通过分析比较的过程中，选择了将二维Mesh 拓扑结构，虚拟直通交换机制以及虚通道技术应用到本文所设计的片上网络中去。

第 三 章 改 进XY 自适应路由算法的设计

路出算法决定了数据包在片上网络传输的过程中所要经历的路径。有效的路由算法能够缩短数据包的传输延迟，增大网络的吞吐量，对系统性能的提高起到至关重要的作用。本章主要讨论改进 XY 自适应路由算法的相关内容，首先阐述改 进XY 自适应路由算法提出的原因，其后描述改进XY 自适应路由算法的实现，最后针对改进 XY 自适应路由算法的特点，对仲裁机制的设计进行详细说明。

3.1 改进 XY 自适应路由算法的提出

出上一章的内容可知，路由算法按照是否根据当前网络状况做出路由路径的调整，可以分为确定路由算法和自适应路由算法两种。

确定路由算法实现简单，当网络数据包注入率较低时，确定路由算法能够很好的发挥效能，数据包传输延迟较低，片上网络可以获得较好的性能。但是随着网络数据包注入率的提高，整个片上网络可能出现拥塞的状况，从而导致数据包传输延迟增大，片上网络的性能迅速降低。因而对于那些通信量需求很大的应用来说，确定路由算法的采用将对系统性能造成制约。

自适应路由算法与确定路出算法不同，能够根据网络实时状况做出路由路径的调整，从而有效避免的网络拥塞情况的发生，使得片上网络的性能一直保持在一个相对稳定的状态。因而对于通信量需求很大的应用来说，采用自适应路由算法更能够使系统发挥更好的效能。

对于当今大部分集成到片上系统中应用来说，系统功能的实现是通过多个模块之间的协同合作来完成的，因而在系统不同的模块之间就需要进行数据的通信，特别是对于音频视频处理的应用来说，更是需要巨大的通信量。如果在模块之间采用合理的路由算法对通信数据进行路径路由，那么对于系统性能的提升将会起到事半功倍的作用。自适应路出算法正好能够满足这样的需求，因而对自适应路由算法的研究是相当有必要的。目前，在全世界范围内已有许多研究人员提出了多种可行的基于片上网络的自适应路由算法。

文献[31]中提出了一种基于路由表方式的自适应路由算法 APSRA, 该算法能够针对具体应用提供无死锁的路由方案。APSRA 算法以特定应用的通信结构图作为输入，通过分析节点之间的通信关系以及节点之间通信关系在整个通信过程中的动态变化情况，生成特定的路由表存储在路由节点之中。当数据包在片上网络传输的过程中，通过读取路由表中的相应信息来判断输出端口的方向。由于在进

算法以应用的通信结构图作为输入，对通信结构图的分析以及路由表的生成的过程相对来说都十分复杂。此外，在路由节点中采用路由表的方式记录路由信息，占用了系统大量的资源及面积，与NoC 精简系统资源及面积的设计目标相违背。

文献[32]中在 XY 确定路由算法的基础上提出了一种自适应路由算法。在模型中，每一个路由节点都是一个对网络状况敏感的个体，整个网络是一个对网络状况敏感的群体。片上网络能够通过对当前的网络状况的分析将通信量合理的分布在整个网络中，并将发生阻碍的路由节点从片上网络中分离开来。在该算法中，路由节点通过接收相邻路由节点所传输过来的配置数据包来进行量化负载值的计算，量化负载值是路由节点当前存储状况的一种量化表现形式。路由节点通过对相邻路由节点量化负载值的分析和比较为数据包选择恰当的输出方向。当片上网络出现拥塞状况时，路由节点通过发送重配置数据包的方式进行路由节点之间的通信，从而使得通信量被合理的分布到整个片上网络当中。该算法在一定程度上缓解了网络拥塞情况发生的可能性，但是量化负载值的计算以及自适应过程的实现所需要发送的配置输出包都为整个片上网络增加了许多额外的通信量，从而加重了片上网络的负担。此外，数据包被分配的静态优先级，不会根据当前网络状况或数据包在路由节点中等待的时间而发生变化，因而在仲裁的过程中可能由于长期无法得到输出响应而出现“饥饿”的状况。

文献[33]中提出了一种适用于Mesh 拓扑结构片上网络的自适应路由算法。根据相邻节点的缓存资源的使用情况，数据包在算法的作用下沿最短路径或非最短路径在片上网络中进行传输。最短路径及非最短路径的选择都是在奇偶偏转路由算法的基础上实现的，因而能够有效的避免片上网络中死锁或活锁的发生。通过该算法，数据通信量被合理的均匀分布在整个片上网络之中，避免出现某一个节点负载过重的情况发生。虽然该算法能够在一定程度上缓解网络拥塞情况的发生，使片上网络获得较好的性能，但是当数据包沿非最短路径前行时，由于绕行所花费的时间可能会大于数据包在路由节点中等待输出端口响应的时间，因而造成更大的传输延迟，反而得不偿失。

通过对上述三个自适应路由算法的分析可以看出，对于一些常见的自适应路由算法来说，主要存在以下几个方面的不足：

1. 算法实现复杂度大。为了能够获得稳定的网络性能，自适应路由算法一般都会采用复杂的手段来实现能够根据网络当前状况对数据包路由路径进行自适应调整的目的。这样不仅提高了整个片上网络实现的复杂度，而且占用了系统大量的资源和面积。

2. 算法不具有通用性。为了实现路由算法的自适应性，可能需要提前对网络

算法不具有通用性。

3. 数据包绕行传输。为了避免长时间的等待，在自适应路由算法的作用下，数据包可能会沿着非最短路径进行传输。这样的策略虽然能够在一定程度上缓解路由节点的负担，但是绕行所花费的时间可能带来更大的传输延迟。

以上述自适应路由算法中所存在的问题为鉴，本文在对简单 XY 自适应路由算法进行分析的基础上做出一定的改进，提出了一种简单可行的自适应路由算法——改进 XY 自适应路由算法。

3.2 改进XY 自适应路由算法的实现

简单 XY 自适应路由算法是一种带有一定自适应性的维序路由算法，该算法在 XY 确定路由算法的基础上做出了改进，从而使得数据包的在网络传输的过程中能够根据网络当前的拥塞状况做出路径的调整，将确定路由算法的简单性和自适应路由算法的灵活性很好的结合在一起。简单 XY 自适应路由算法虽然在一定程度上提高了网络的性能，但是依然存在着不足之处。

当采用简单 XY 自适应路由算法时，多个输入端口可能同时向 X 轴方向的输出端口发送输出请求，这就可能造成在同一时间内 Y 轴方向的输出端口处于无任何输出请求的空闲状态。而一个输出端口在一次传输过程中只能接收来自一个输入端口的输出请求。当一个输入端口获得输出响应时，其余尚未得到响应的输入端口又转而都向 Y 轴方向的输出端口再次发送输出请求，进而可能造成在同一时间 内X 轴方向上的输出端口处于无任何输出请求的空闲状态。即该算法在实现的过程中输入端口在同一时间内仅向一个方向的输出端口发送输出请求，这样单向数据输出竞争策略可能使得另一个方向的输出端口处于空闲的状态，因而造成输出端口没有能够得到充分的利用。

为了提高输出端口利用率，降低数据传输延时，本文对简单 XY 自适应路由算法做出了改进，提出了改进XY 自适应路由算法，该算法采用XY 双向输出竞争策略，即输入端口同时向X 轴 ，Y 轴方向的输出端口发送输出请求，从而在不占用过多系统资源的情况下提高输出端口的利用率；并引入了动态优先级的仲裁机制遏制饥饿情况的发生；数据包始终沿最短路径前行，以避免沿非最短路径前行所造成的额外的传输延迟。

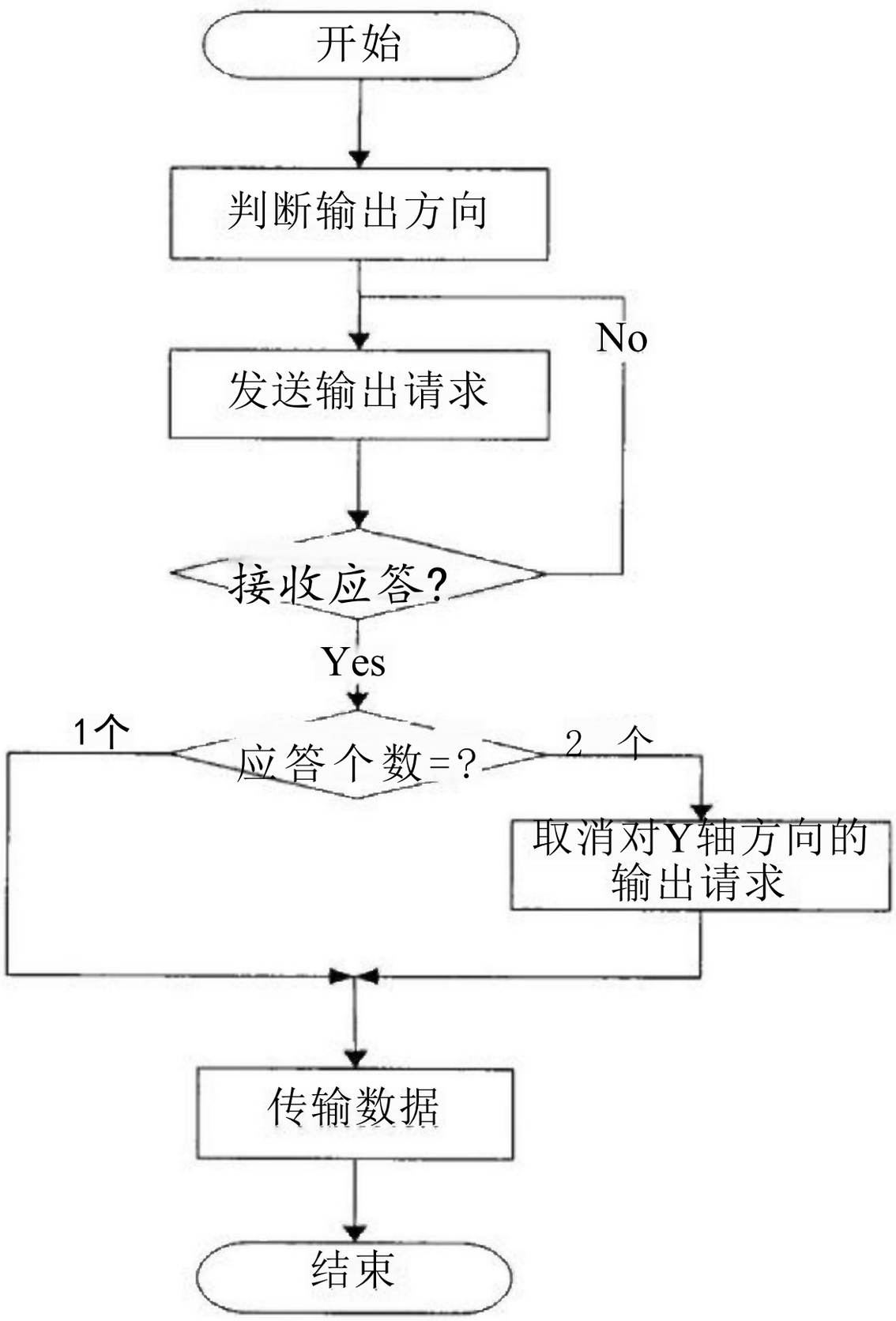


图3.1 改进 XY 自适应路由算法

图3.1 为改进XY 自适应路由算法的流程图，从算法的流程图中可以看出，当数据包到达当前路由节点时，对数据包中所包含的路由信息进行分析以判断数据包的输出方向。当前路由节点即为目的节点时，数据包等待直至通过本地输出端口发送至本地处理单元；当前路由节点不是目的节点时，如果数据包在 X 轴方向仍未到达确定坐标，那么同时向相应X 轴方向的输出端口以及相应Y 轴方向的输出端口发送输出请求，如果仅得到来自X 轴方向输出端口或Y 轴方向输出端口的单一响应，那么沿着获得响应方向的输出端口传送至下一级的路由节点，同时取消向另一个未获得响应方向的输出端口发送的输出请求。如果同时获得来自相应X 轴方向输出端口以及相应Y 轴方向输出端口的响应，那么按照 XY 确定路由算法中所规定的先X 后 Y 的规则，接受来自X 轴方向的响应，放弃来自Y 轴方向的响应，并沿X 轴方向的输出端口传送至下一级的路由节点，同时取消向Y 轴方向输出端口发送的输出请求。**如果数据包在X 轴方向已经到达确定坐标，数据包等待**直至通过相应Y 轴方向的输出端口发送至下一级路由节点。

3.3 仲裁机制的实现

出端口输出请求响应的输入端口可能同时获得来自 Y 轴方向输出端口的输出请求响应，这样，按照改进 XY 自适应路由算法的思想，来自于 Y 轴方向输出端口的输出请求响应应该被舍弃，接受来自于X 轴方向输出端口的输出请求响应，数据包沿 X 轴方向的输出端口传输到下一级路出节点。如果这样的情况连续发生，那么对片上网络性能的提高会产生一定的影响。为了进一步提高输出端口的利用率，减少一个输入端口同时接收来自X 轴 ，Y 轴两个方向输出端口输出请求响应情况的发生，需要对仲裁机制做出相应的调整。

Round Robin 轮询调度算法由于其实现简单，调度公平性好，因而在许多网络交换需要数据包调度的场合中得到了广泛的应用34]。本文就采用 Round Robin 轮询调度算法进行输出请求的仲裁。

对于 n 个输入仲裁请求 R1,R2, …,Rn 来说，初始状态下规定其优先级顺序为： R1>R2>...>Rn。假设在第一次仲裁过程中，某个输入仲裁请求 Rm 得到了响应，那么在下一次仲裁时输入优先级被改变，其顺序为：R(m+1)>R(m+2)>...>Rn>R1>R2>...>Rm, 即将 Rm 及在初始优先级顺序中优先级高于Rm 的所有输入申请 R1,R2,...,Rm 作为一个整体(其内部优先级关系不变),其优先级降至R(m+1),R(m+2),...,Rn 之后。在后续的每次响应后都按照上述的规则在初始化优先级的基础上进行变化9]。

表3.1 各方向输出端口初始化优先级顺序

	0	1	2	3
东	本地	西	北	南
南	北	东	西	本地
西	本地	东	北	南
北	南	东	西	本地
本地	北	南	东	西

如表3.1 所示，X 轴方向， Y 轴方向，本地方向的输出端口分别采用不同的初始化优先级顺序。初始化优先级顺序是以 XY 确定路由算法为出发点，以输入端口向相应输出端口发送请求的概率为依据而确定的。根据 XY 确定路由算法，数据包始终按照先X 轴方向后Y 轴方向的顺序在整个片上网络中进行传输。

对于一个由本地处理单元发送过来的数据包来说，其沿 X 轴方向进行传输的可能性最大，沿Y 轴方向传输的可能性最小。因此，对于X 轴方向的输出端口来说，本地输入端口具有最高优先级，对于 Y 轴方向的输出端口来说，本地输入端口具有最低的优先级。

对于一个X 轴方向发送过来的数据包来说，其沿X 轴方向继续进行传输的可

入端口具有较高的优先级，对于Y 轴方向的输出端口来说，来自于X 轴方向的输入端口也具有较高的优先级，对于本地输出端口来说，来自于X 轴方向的输入端口具有最低的优先级。

对于一个由Y 轴方向发过来的数据包来说，其沿Y 轴方向继续进行传输以及沿本地输出端口传输至本地处理单元的可能性最大，其沿X 轴方向继续进行传输的可能性最低。因此，对于X 轴方向的输出端口来说，来自于Y 轴方向的输出端口具有最低的优先级，对于Y 轴方向的输出端口来说，来自于相反Y 轴方向的输入端口最高的优先级，对于本地输出端口来说，来自于 Y 轴方向的输入端口具有最高的优先级。

通过各输出端口分别采用不同初始化优先级顺序的方法可以使得在X 轴方向输出端口具有较高优先级的输入端口在Y 轴方向具有较低的优先级，这样就降低了同一输入端口同时获得X 轴 ，Y 轴两个方向输出端口输出请求响应的可能性。

由于采用 XY 双向输出竞争的策略，获得 Y 轴方向输出端口输出请求响应的输入端口未必会沿着该 Y 轴方向将数据包传送至下一级路由节点，如果在该方向上仍然采用 Round Robin轮询调度算法进行输入请求的仲裁可能会出现“假输出”的状况，即输入端口虽然获得了输出响应但数据包并未沿该输出端口进行传输。为了避免这种状况的发生，对于Y 轴方向的输出端口来说，需要对 Round Robin 轮询调度算法进行改进。而对于X 轴方向的输出端口以及本地输出端口来说，可以直接按照 Round Robin 轮询调度算法进行输出请求的仲裁。

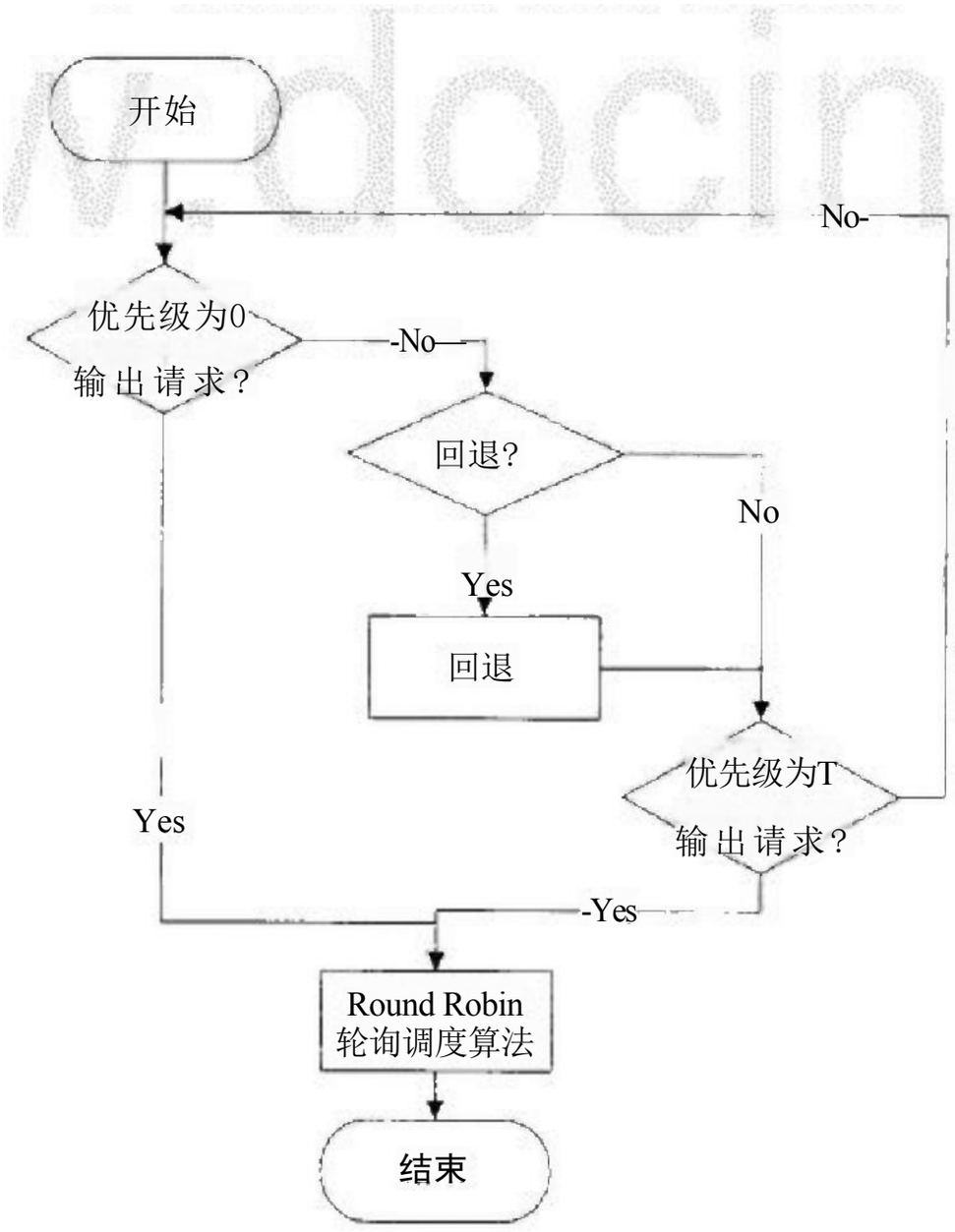


图3.2 Y 轴方向输出端口仲裁机制的实现