西安电子科技大学第十一届文献综述大赛

片上网络容错路由算法综述

姓　名：　王秀程

学　号：　17010199045

专　业：　通信工程专业

学　历：　本

学　院：　通信工程学院

组　员：　曹思齐

2019 年 3 月

摘要

随着半导体制作工艺地快速发展, 多核的片上系统 (Soc) 在电子设备中得到了越来越广泛的应用，但是其基于总线的架构，也使得片上系统在使用中逐渐面临着越来越大的片上通信压力。而片上网络 (Noc) 则因其具有高带宽、低延迟和良好的灵活性等优点, 可以较为有效地解决片上通信的巨大压力的特性，得到了各国相关领域研究人员的重视。而芯片制作的深亚微米级工艺, 也使得芯片的可靠性已成为制约集成电路发展的关键因素。为了提高片上网络 (Noc) 的容错能力, 基于片上网络 (Noc) 的容错路由算法，在近年来得到了广泛的研究, 国内外研究人员也提出了诸多极具使用价值的结构设计。本文对近五年，特别是近三年来国内外研究人员所提出的算法结构进行了简要的介绍，并就其算法特性做出了一定的总结。此外, 还对片上网络容错路由算法，未来的研究方向提出了一定的展望。

关键词: 片上网络，片上通信，可靠性，容错路由算法

Abstract

With the rapid development of semiconductor manufacturing technology, many-core SoC has been more and more widely used in electronic equipment, at the same time, because of its bus architecture, the using of it is also facing a growing on-chip communication pressure. NoC can solve the huge pressure of on-chip communication because of its advantages such as high bandwidth, low latency and good flexibility. Because the chip making process has reached the deep sub-micron level, the reliability of the chip has become a key factor restricting the development of integrated circuits. In order to improve the fault tolerance of NoC, fault-tolerant routing algorithm based on NoC has been widely studied in recent years, and researchers at home and abroad have also put forward a lot of valuable structural design. This paper gives a brief introduction to the algorithm structure proposed by researchers at home and abroad in recent five years, especially in the past three years, and makes a certain summary of its algorithm characteristics. In addition, some prospects for the future research direction of on-chip network fault-tolerant routing algorithm are also put forward.

Keywords: NoC, On-Chip Communication, Reliability, Fault-Tolerant Routing Algorithms

第 1 章　绪论

1.1　研究背景

随着集成电路制作工艺的快速发展，芯片的集成成度，已达到了前所未有的高度。越来越多的计算核心与加速单元等 IP 核，可以被制造在一张芯片之上 [1]，从而大大提高了片上系统 (SoC) 的处理与运算能力。正因此, 芯片算力已不再是阻碍芯片性能进步的最重要的瓶颈问题 [2]。SoC 的设计理念，也逐渐从“以提高计算性能为核心”，向“以提高通信性能为核心”转变。

而且，传统的基于总线结构的片上系统，因其难以满足“高带宽”、“灵活性”，与“高可靠性”的片上通信需求，正受到国内外研究人员越来越多的质疑。片上网络 (NoC) 架构便是在此背景下应运而生的，并因其出色的片上通信性能而渐渐成为相关领域的研究重点。其中基于片上网络架构的路由算法，更是一大热点研究领域。

1.2　研究意义

当前，受限于研发技术与制作工艺，工业化批量生产的集成电路，几乎无法达到完全理想的使用状态。在芯片的使用过程中，将不可避免得因为温度、电压等因素 [3]，造成芯片物理元件的瞬时或永久的损坏。而芯片基本元件的物理损坏，将不可避免地影响该元件地逻辑电平输出。更为关键的是，随着芯片集成度的提升，芯片中基本元件发生故障的问题已愈发严重。

NASA 的研究 [4] 指出，随着芯片深亚微米级制作工艺的不断发展，芯片发生故障的

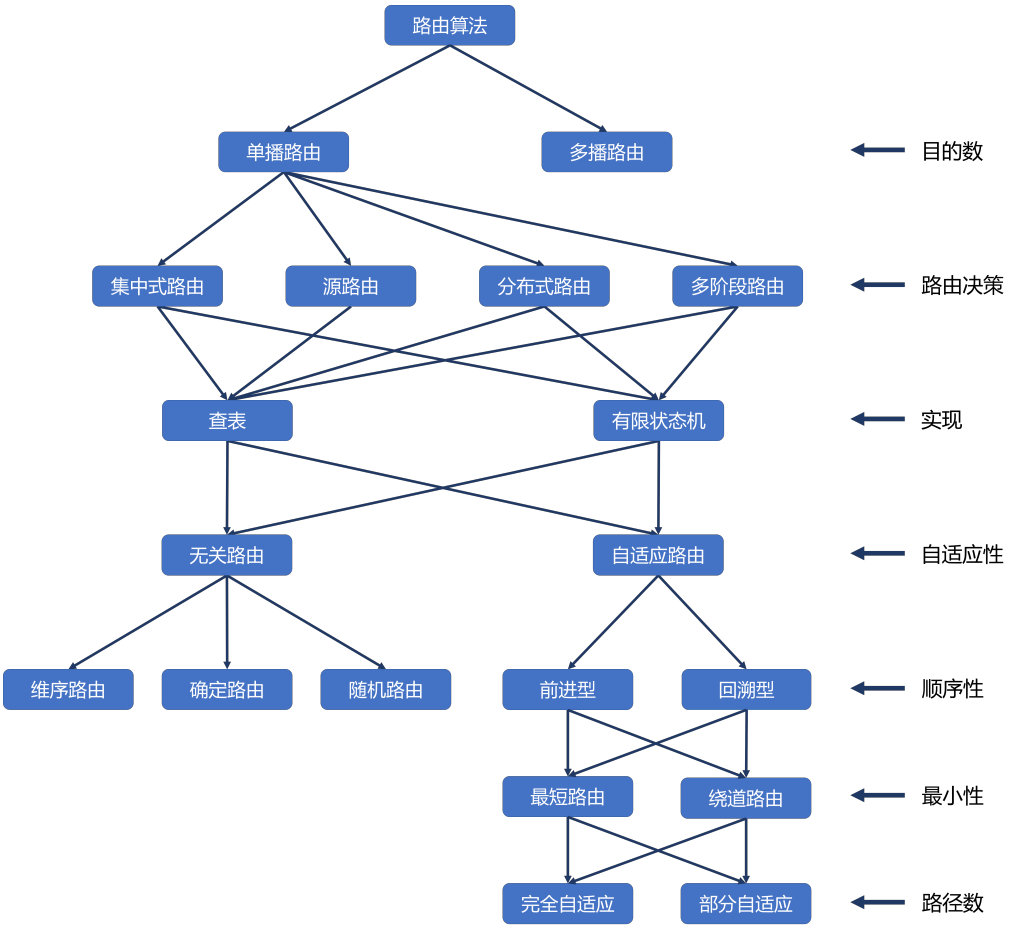

图 1: 容错路由算法的分类

概率也在不断增加。如文献[3]所示，由于制作工艺的不足，在芯片使用的早期阶段，会因为各部件间的相互磨合，使得故障发生率有所下降。而在芯片使用的主要阶段，芯片所产生的故障，主要由元件的随机错误产生，在此阶段信芯片的整体的错误率基本稳定。而在芯片使用的最后阶段，由于自然的氧化与磨损，整体故障率会快速上升[5]。而深亚微米技术的应用，不仅提高了芯片寿命周期中随机错误的发生概率，还会加速芯片的自然老化[6]。因而为了更高效地使用芯片，避免芯片因故障而过早损坏或失去使用价值，对片上网络的容错设计是十分重要的课题。而具有容错能力的片上路由算法，也因具有“容错性能好”、“不需改变物理架构”的特点，使得对其进行研究具有更高的应用价值。

1.3 本文结构

本文第一章为绪论，在绪论部分介绍了片上网络架构的研究背景，以及对其进行容错算法开发的现实意义。第二章则介绍了几种近几年国外研究人员所提出的，具有代表性的容错路由算法。并通过“文字叙述”、“路由图解”、“伪代码”等形式对其工作原理，做出了较为详细的描述。进而对其算法特性进行了简要概括。第三章是对近几年国内研究人员所提出的具有代表性的容错路由算法的介绍，基本结构与第二章相仿。第四章为总结与展望，通过比较归类的方式，将本文中所提到的算法大致分为三类，并对每一类的优势与不足做出了简要的叙述。最后对容错路由算法未来的发展方向，提出了一定的展望。

第 2 章　容错路由算法的国外研究现状

2.1 动态 XY-YX 路由算法

由 Yott Khtchar[7] 等人所提出的动态 XY-YX 算法以确定维序 XY 路由算法为基础，当源节点向目的节点发送信息时，先采用 XY 路由算法，在 XY 路由进行过程中如果遭遇故障节点或故障链路，则将动态调整路由方式。

2.1.1 工作原理

(1) 将源节点的 X，Y 坐标与目的节点的 X，Y 坐标相比较，利用 XY 路由算法将信息传递至目的节点。

(2) 当路由方向为任一方向 (X 或 Y) 且在该方向上的下一节点出现故障时，则将该故障链路标记为 ND，并将路由方向切换为另一方向。

(3) 当前路由方向为 Y 方向，且当前节点与目的节点处在同一列时，若遭遇故障链路，则将路由算法切换为 YX 算法，同时将数据包沿 X 方向传播一个节点。

(4) 当前路由方向为 X 方向，且当前节点与目的节点处在同一行时，若遭遇故障链路，处理方式与 (3) 类似。

(5) 若不论沿 X 方向还是 Y 方向传播，都遭遇故障链路，使得数据包无法进一步向目的节点靠近，则将数据包回溯一步，并调整为 YX 路由算法。

2.1.2 算法伪代码

2.1.3 算法特性

(1) **算法简单，开销低**：该算法主体采用 XY 与 YX 路由的传播方式，使得在 NoC 上应用该算法时无需对静态路由算法做出过多修改。

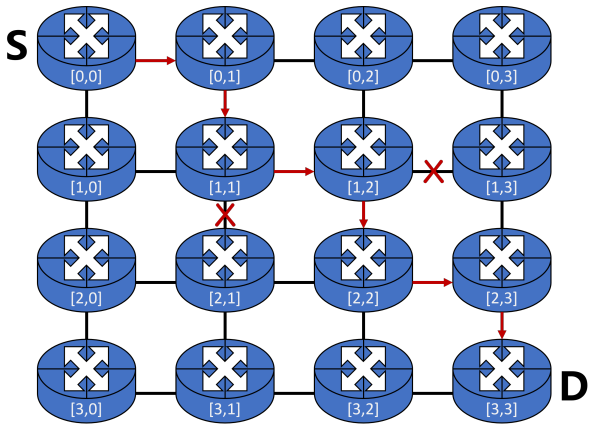

图 2: 动态 XY-YX 路由算法示意图 a

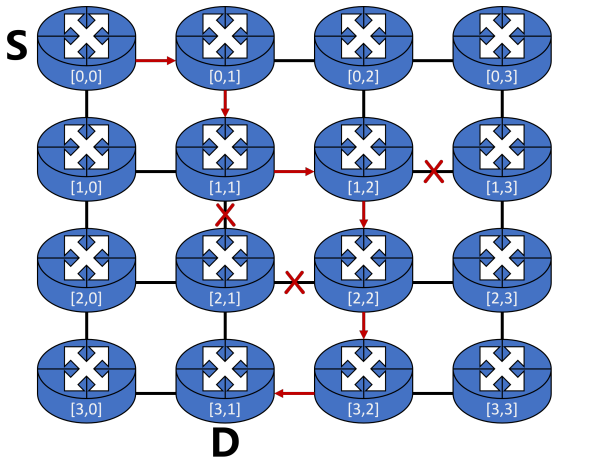

图 3: 动态 XY-YX 路由算法示意图 b

(2) **无死锁，活锁**：通过 XY 与 YX 之间的切换，使得算法不会出现死锁现象，特别是当 X，Y 同时故障时，因允许数据回溯，使算法的容错能力得到进一步提升。

Algorithm 1 动态 XY-YX 路由算法

RouteStrategy = Initial

while $CurrNode! = Destiny$ **do**
  if $RouteStrategy == Initial$ **then**
    RouteStrategy = XY
  end if

  if $Next == ND$ **then**
    if $CurrDir == YANDCurrY == DestinyY$ **then**
      RouteOneStep(X)
      RouteStrategy = YX
    end if
    if $CurrDir == XANDCurrX == DestinyX$ **then**
      RouteOneStep(Y)
      RouteStrategy = XY
    end if
    if $RouteStrategy == XY$ **then**
      RouteStrategy == YX
    else
      RouteStrategy == XY
    end if
    if $X == ND$**AND**$Y == ND$ **then**
      ReturnTo(LastState)
      RouteStrategy = YX
    end if
  end if
end while

2.2 绕道路由算法

对于 NoC 中的任一节点，其直接相连的节点只有“东、西、南、北”四个方向的节点。绕道路由算法的关键，是把数据从故障节点的一个直接相邻节点按照一定的规范 (如顺时针或逆时针) 传递至另一直接邻接节点，以避开故障链路 [8]。

2.2.1 工作原理

(1) 将源节点的 X，Y 坐标与目的节点的 X，Y 坐标相比较，利用 XY 路由算法将信息传递至目的节点。

(2) 当路由方向为任一方向时，若该方向的下一节点出现故障，则比较当前节点与目的节点的坐标值，并将 X 与 Y 维度上靠近目

的节点的方向标记为正方向。

(3) 若当前节点与目的节点在同一行，此处以目的节点在当前节点的东方为例，则将数据按照顺时针方向，依次经过故障节点的西南、南、东南、东向邻接节点从而绕开故障节点，继续路由。

(4) 若当前节点与故障节点位于同一列，其旁路选择方案与 (3) 类似。

(5) 若当前节点与目的节点既不在同一行，也不在同一列，则按接近目的节点的方向，按顺时针或逆时针方向前往另一维度上的直接相邻节点，并按 XY 算法进行路由，直至信号到达目的节点或遭遇下一个故障链路。

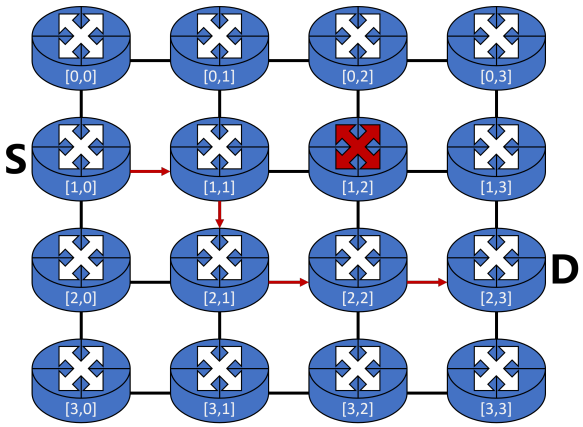

图 4: 绕道路由算法示意图

2.2.2 算法特性

(1) **旁路的选用**：区别于普通的基于 XY 和 YX 算法切换的自适应算法，本算法创造性地提出了“旁路 (Bypass)”的概念，使得该算法在遭遇故障链路时，只需按照顺时针或逆时针方向绕过故障节点，在保持可靠性的同时，兼顾了简洁性。

(2) **解决单故障节点时的高效性**：对于单发的故障节点，该算法在处理上具有明显的高效、高可靠性的特点。但是在对于故障链路连续出现的情况 (如当前节点与故障节点在同一行，在同一列上有连续三个故障点)，则有潜在的活锁可能。

2.3 维度顺序算法

维度顺序算法是一种基于 XY 路由算法的优化方案。当链路中无故障点时，采用 XY 算法；当遭遇故障链路时，则切换传播维度，将数据转向另一个维度，向靠近目的节点的方向传播 [9]。

2.3.1 工作原理

(1) 当数据沿 X 方向路由时，若遭遇故障节点，则切换至 Y 方向路由，并沿 Y 维度向靠近目的节点的方向传播。

(2) 当数据沿 Y 方向路由时，若遭遇故障节点，则切换至 X 方向路由，并沿 X 维度向靠近目的节点的方向传播。

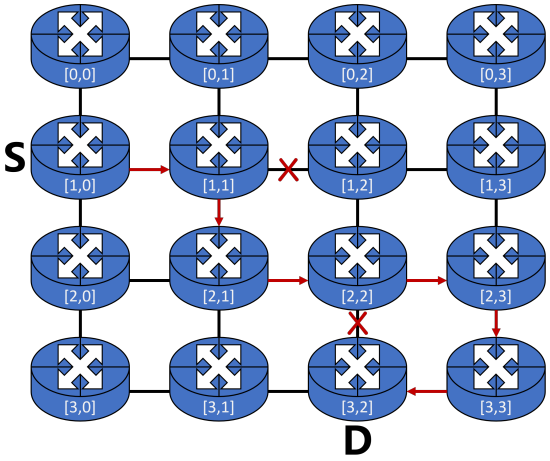

图 5: 维度顺序算法示意图

2.3.2 算法特性

(1) **容错性高**：利用 X、Y 方向的切换，提高了数据的可达性。

(2) **结构简单**：在遭遇故障节点时，只需要切换维度，算法逻辑简单。

2.4 Dn-FTR 算法

Dn-FTR 算法是一种加入了回溯法的基于切换 XY 与 YX 算法的容错算法，并通过对路由失败的规定，避免了死锁等故障的发生 [10]。

2.4.1 工作原理

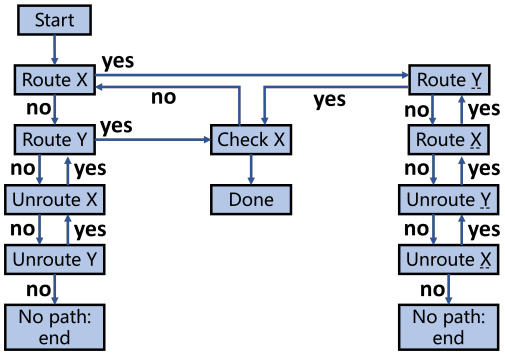

图 6: Dn-FTR 算法流程图

(1) 将源节点的 X,Y 坐标与目的节点的 X,Y 坐标相比较，利用 XY 路由算法将数据传送至目的节点

(2) 当数据包沿 X 维度路由时遭遇故障链路，则将数据沿 Y 方向传播一个节点，若 Y 方向链路可行则切换路由方式为 YX 算法，否则跳转至步骤 (3)。

(3) 将数据沿 X 方向回溯一个节点，若回溯链路可行，则跳转至步骤 (2) 继续路由，否则执行步骤 (4)

(4) 将数据沿 Y 方向回溯一个节点，若回溯链路可行，则跳转至步骤 (3)，若不可行则路由失败，销毁数据包并重新发送。

(5) 若在 Y 维度上路由时遭遇故障链路，处理方式与在 X 维度上相似。

2.4.2 算法特性

(1) **算法结构简单**：相较于前几种，该算法在进行容错路由时，由于在第 (3)、(4) 步的处理中引入了递归的思想，使得算法描述更加精简，算法结构也更为简单。

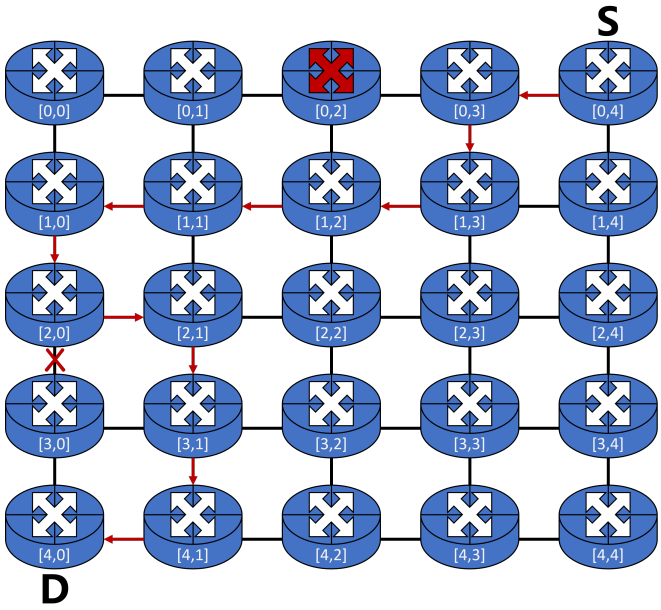

图 7: Dn-FTR 算法示意图

(2) **路由开销略微提升**：区别于前几类算法，该算法在回溯时，需要保存的回溯信息较多，尤其是当算法需要执行第 (4) 步时，需要至少保存 3 个之前节点的路由信息。而同样需要保存回溯信息的动态 XY-YX 算法，则只需保存一步回溯信息。与之相较，该算法的路由成本有略微的提升。

第 3 章 容错路由算法的国内研究现状

3.1 T-XY 算法

T-XY 路由算法的关键思想是：当网络中不存在故障时，采用 XY 路由算法，若在 XY 路由的过程中遭遇故障节点，则向离目标节点近的垂直方向转弯路由。当相对于目标节点有 2 个相同距离的垂直节点，则根据不同方向优先选择指定的垂直节点进行路由 [11]。

3.1.1 工作原理

(1) 将源节点的 X,Y 坐标与目的节点的 X,Y 坐标相比较，利用 XY 路由算法将数据传送至目的节点。

(2) 若数据在 X 维度上进行路由时遭遇故障节点，且当前节点与目的节点的 Y 维坐标相等，则将数据沿 Y 方向传播一个节点，然后继续按 XY 算法将数据传送至目的节点。特别的：当数据在向东路由时遭遇故障节点，则优先考虑将数据沿 Y+ 方向传播，若 Y+ 方向链路不可用，再考虑将数据向 Y-方向传播以避开故障节点。

(3) 若数据在 Y 维度上路由时遭遇故障节点，且当前节点与目的节点的 Y 维坐标相等，则将数据沿 X 方向传播一个节点，然后按照 YX 算法将数据传送至目的节点。

(4) 若数据在 X 维向东路由时遇到故障，且目的节点位于当前节点的东北方向则转为 YX 算法；若在 Y 维度向北路由时再次遭遇故障节点，则转换为 XY 算法。

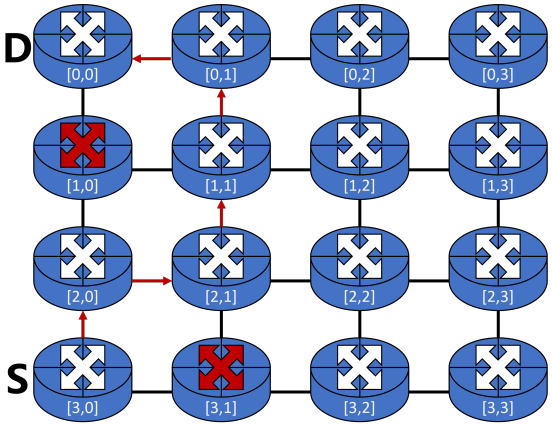

图 8: T-XY 算法示意图 a

3.1.2 算法伪代码

3.1.3 算法特性

(1) **一定的拥塞调节能力**：该算法根据目的节点与当前节点的相对位置的不同，采用

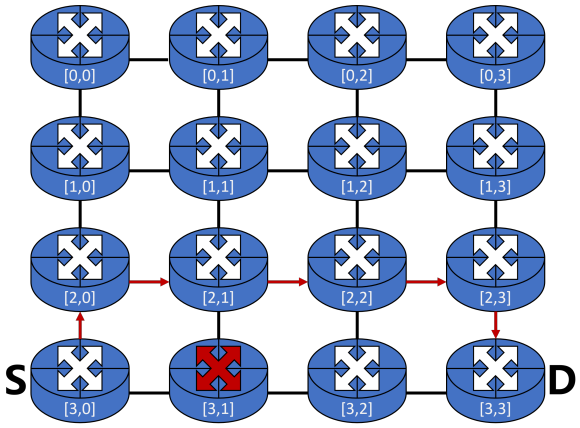

图 9: T-XY 算法示意图 b

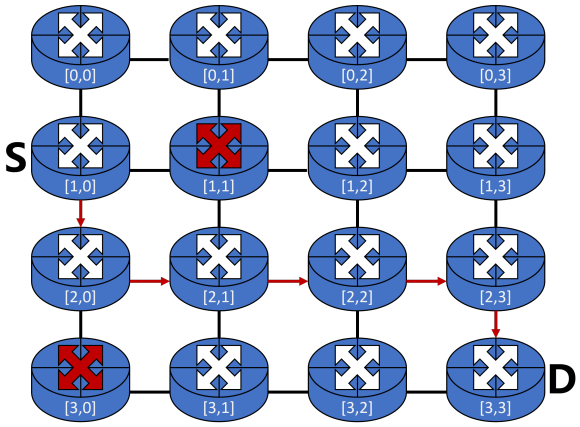

图 10: T-XY 算法示意图 c

不同的拐弯方式，对网络的拥塞状况也具有一定调节作用，同时容错带来的能耗和开销也相对较低。

(2) **较好的时延特性**：除了目的节点与当前节点在同一条直线上时，路由过程具有绕道现象外，在其他故障状态下容错算法所选择的路由路径，仍然为两节点间的最短路径。

3.2 PR 算法

PR 算法本质上是对 XY 路由算法的一种改进，其主要特点是在进行容错工作时，对数据传送方向的优先级做出了明确的规定。相交于同样对传送方向优先级做出规定的 T-XY 算法，PR 算法的另一特点是：对如何确定路由失败并销毁数据包做出了相关定义 [12]。

Algorithm 2 T-XY 算法

```
RouteStrategy = Initial

while CurrNode! = Destiny do
    if RouteStrategy == Initial then
        RouteStrategy = XY
    end if

    if X+ == NDANDCurrY == DestinyY then
        if Y+ == Good then
            RouteOneStep(Y+)
            RouteStrategy = XY
        else if Y− == Good then
            RouteOneStep(Y-)
            RouteStrategy = XY
        end if
    else if X- == ND AND CurrY == DestinyY then
        if ifY− == Good then
            RouteOneStep(Y-)
            RouteStrategy = XY
        else if Y+ == Good then
            RouteOneStep(Y+)
            RouteStrategy = XY
        end if
    else if Y+ == ND AND CurrX == DestinyX then
        if ifX+ == Good then
            RouteOneStep(X+)
            RouteStrategy = YX
        else if X− == Good then
            RouteOneStep(X-)
            RouteStrategy = YX
        end if
    else if Y- == ND AND CurrX == DestinyX then
        if ifX− == Good then
            RouteOneStep(X-)
            RouteStrategy = YX
        else if X+ == Good then
            RouteOneStep(X+)
            RouteStrategy = YX
        end if
    else
        if RouteStrategy == XY then
            RouteStrategy = YX
        else
            RouteStrategy = XY
        end if
    end if
end while
```

3.2.1 工作原理

(1) 将源节点的 X,Y 坐标与目的节点的 X,Y 坐标相比较，利用 XY 路由算法将数据传送至目的节点

(2) 若数据包在进行 XY 路由时遭遇故障节点，则比较目标节点于当前节点的坐标值。分别将 X、Y 维度上接近目标节点的方向定义为 X+ 与 Y+ 方向。首先判断 X+ 方向链路是否可行，若可行则将数据沿 X+ 方向路由；若不可行则判断 Y+ 方向链路是否可行，若可行则切换为 YX 路由算法；若不行，则将数据包沿 Y-方向传播一个节点，然后返回步骤（2）开始状态；若仍然不行，则将数据包沿 X-方向传播一个节点，然后返回步骤 (2) 开始状态。如果所有方向的链路都不可行，则输出状态：路由失败，并将数据包丢弃，重新发送。

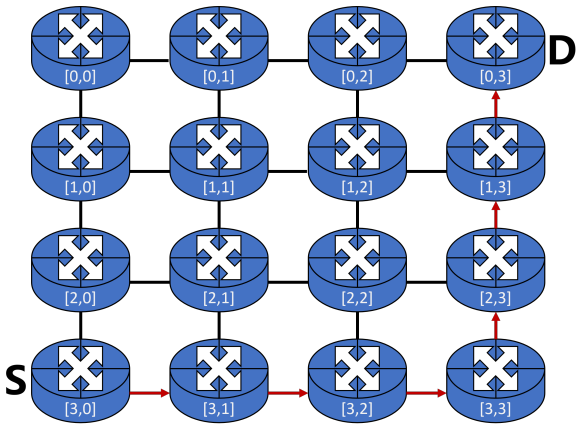

图 11: PR 算法示意图 a

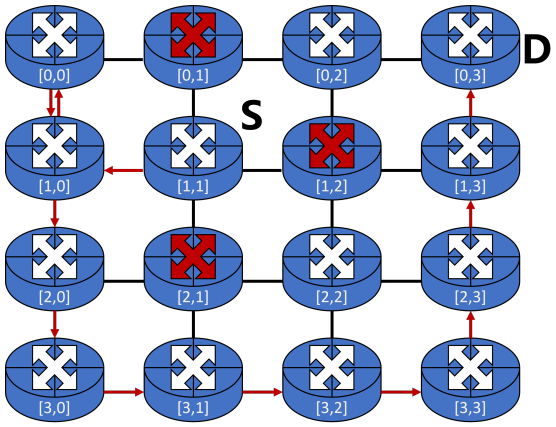

图 12: PR 算法示意图 b

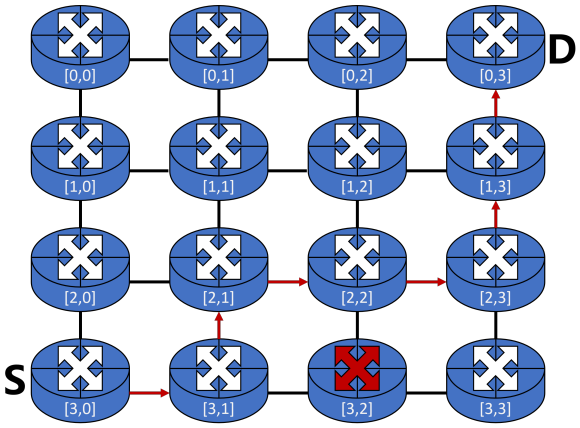

图 13: PR 算法示意图 c

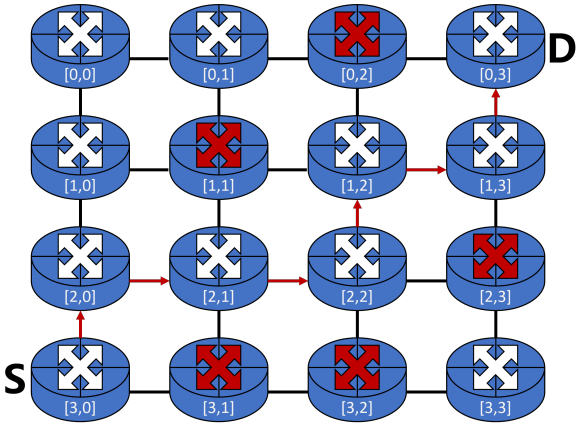

图 14: PR 算法示意图 d

3.2.2 算法伪代码

3.2.3 算法特性

(1) **适应性强**：由于算法在遭遇故障节点时，做出的决策与故障节点的位置及故障节点的数量无关。因此该算法不仅适用于故障节点数量较少的情形，当网络中的故障节点较多时也同样适用。

(2) **数据可达性高**：由于该算法对转向的优先级做出了规定，故可以有效的避免死锁、活锁现象的发生。只要网络中存在从目的节点到源节点的通路，该算法便可找出其中一条作为路由路径。

(3) **网络开销低**：相对于泛洪算法，本算法在兼顾容错性能的同时，网络开销也大大降低。

Algorithm 3 PR 算法

```
1:  while CurrNode! = Destiny do
2:      Route(XY)
3:      if Next == ND then
4:          if DestinyX >= CurrX then
5:              DirX = X+
6:          else
7:              DirX = X−
8:          end if
9:          if DestinyY >= CurrY then
10:             DirY = Y+
11:         else
12:             DirY = Y−
13:         end if
14:
15:         if DirX == Good then
16:             Route(DirX)
17:         else if DirY == Good then
18:             Route(YX)
19:         else if −DirY == Good then
20:             RouteOneStep−DirY
21:         else if −DirX == Good then
22:             RouteOneStep(−DirX)
23:         else
24:             RoutingFailed
25:         end if
26:     end if
27: end while
```

3.3 故障区再利用算法

传统的基于故障区域划分的算法模型，往往禁止数据进入故障区域，并要求数据包沿故障区域边沿传播，以避开故障节点，并通过转弯限制等方式避免死锁。而文献 [13] 所提出的算法，则是对传统算法的进一步改进，其突出特点是：允许数据进入故障区域路由。

3.3.1 工作原理

(1) 按文献 [15] 等经典算法，用矩形框划分出网络的中含有故障节点的区域。

(2) 将源节点的 X,Y 坐标与目的节点的 X,Y 坐标相比较，利用 XY 路由算法将数据传送至目的节点。

(3) 当数据在路由过程中遭遇故障区域，首先判断当前链路是否可行。若不可行，则按文献 [y] 中算法将数据沿故障区域边沿传输。

(4) 若当前链路可行，则继续按 XY 算法进行数据路由，若有一条直接通路存在，使得数据可以离开故障区域，则返回步骤 (2) 继续用 XY 方式路由。

(5) 若在故障区域内部路由时，无法通过简单的 XY 路由方式直接离开故障区域，则允许数据在故障区域中第一次遭遇故障时，将路由方式切换为 YX 方式。若数据能离开故障区域，则返回步骤（2）继续用 XY 方式路由。若在故障区域中第二次遭遇故障链路，则在故障区域中路由失败，数据返回至进入故障区域前的初始节点，并按文献 [y] 中算法将数据沿故障区域边沿传输。

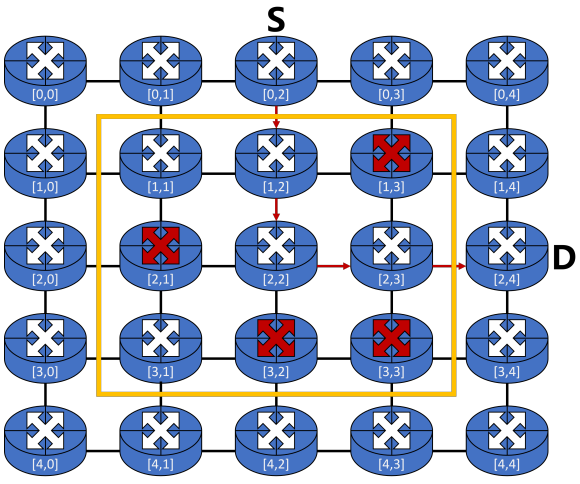

图 15: 故障区再利用算法示意图 d

3.3.2 算法伪代码

3.3.3 算法特性

(1) **时延低**：由于允许数据进入故障链路中路由，故其传播时延低于普通的沿故障区域外延传播的路由算法。

(2) **无死锁**：由于限制了数据在故障区域中的转弯次数，因为避免了死锁的发生。

第 4 章　总结与展望

目前，对于 NoC 片上网络容错路由的研究，仍是学界一较为新型的研究。本文通过对近年来片上网络容错路由算法发展状况的跟踪，通过大量文献资料的阅读，对近年来国内外学者提出的算法进行了分析、总结，并对他们的算法特性做出了一定的比较。综合来看当前所提出的容错算法可大致分为三类：

(1) 基于动态自适应的 XY-YX 算法转换的容错路由算法。在本文中代表算法有“动态 XY-YX 算法”、“绕道路由算法”等。

(2) 基于路径转向优先级设置的容错路由算法。在本文中代表算法有“维度顺序算法”、“T-XY 算法”、“PR 算法”等。

(3) 基于故障区域对网络节点进行矩形划分的容错路由算法。本文中以“故障区再利用算法”为例进行了简要论述。

其中第 (1)、(2) 类算法原理相对简洁，原理也更加易于理解。并且由于其在进行容错处理时，路由决策与故障链路的位置与数量无关，其可适用性更好，即使在数据的传输过程中由于物理因素的影响，使故障节点数目突然增多，其依然具有可靠的自适应性。特别是以“PR 算法”为代表的一类容错算法，得益于其算法结构，只要网络中存在源节点到目的节点间的可行通路，其便可选着一条作为

Algorithm 4 故障区再利用算法

```
1: FaultRingSet = GetFaultRing()
2: RouteStrategy = Initial
3:
4: while CurrNode! = Destiny do
5:     if RouteStrategy == Initial then
6:         RouteStrategy = XY
7:     end if
8:
9:     if CurrNode ∈ FaultRing then
10:        EntryPos = CurrPos
11:        if ∃RoutePathmakeCurrNode + Route(RoutePath) == Destiny then
12:            Route(RoutePath)
13:        else if ∃RoutePathmakeCurrNode + Route(RoutePath) ∉ FaultRing then
14:            Route(OutPath)
15:            RouteStrstegy = XY
16:        else
17:            while Next! = ND do
18:                Route()
19:                if ifCurr ∉ FaultRing then
20:                    Break
21:                end if
22:                RouteStrstegy = YX
23:            end while
24:            while Next! = ND do
25:                Route()
26:                if ifCurr ∉ FaultRing then
27:                    Break
28:                end if
29:                ReturnTo(EntryPos)
30:            end while
31:        end if
32:    end if
33: end while
```

路由路径。然而此类算法在容错处理时，往往需要使用虚通道，且在处理单故障节点时有一定的不足。[14]

因而以上两类算法未来的发展方向，应在与如何兼顾容错能力与虚通道的使用数量，以及如何降低由于算法的绕路决策所造成的时延，从而进一步提高算法性能。

而第 (3) 类算法的突出特点是：在容错处理时不需要使用虚通道，相较而言也具有更好的单故障节点处理能力。然而其往往需要划分出大量的故障区域，而且在区域划分时使用矩形区域划分的方式，往往使得较多的正常链路被标记在故障区域中，使其无法发挥功效，造成了一定的资源浪费。所以对于此类算法，在进一步的研究中，应参考类似文献 [13] 中所提出的设计理念，寻找故障区域中的可用通路，降低对正常链路的资源浪费。

参考文献

[1] 王芳莉, 杜慧敏. 片上网络路由算法综述 [J]. 西安邮电学院学报,2011,16(01):72-77.

[2] Wang, J.-S., & Huang, L.-T. Review on Fault-Tolerant NoC Designs.2018.

[3] Bjerregaard, T., & Mahadevan, S.. A survey of research and practices of Network-on-chip. ACM Computing Surveys , 2006 , 38(1), 1–51.

[4] White, M..Scaled CMOS Technology Reliability Users Guide. 2010.

[5] Collet, J. H.. A brief overview of the challenges of the multicore roadmap. Mixed Design of Integrated Circuits & Systems (MIXDES), 2014 Proceedings of the 21st International Conference. 2014 , 22–29.

[6] Marculescu, R., Hu, J., & Ogras, U. Y.. Key research problems in NoC design: a holistic perspective. Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. 2005 ,69–74.

[7] Khichar, J., Choudhary, S., & Mahar, R. Fault tolerant dynamic XY-YX routing algorithm for network on-chip architecture. 2017 International Conference on Intelligent Computing and Control (I2C2). 2017.

[8] Priya, S., Agarwal, S., & Kapoor, H. K. Fault Tolerance in Network on Chip Using Bypass Path Establishing Packets. 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems. 2018, 457–458.

[9] Tatas, K., Sawa, S., & Kyriacou, C. (2014). Low-cost fault-tolerant routing for regular topology NoCs. 21st IEEE International Conference on Electronics, Circuits and Systems . 2014, 566–569.

[10] Sinha, D., Roy, A., Kumar, K. V., Kulkarni, P., & Soumya, J. Dn-FTR: Fault-tolerant routing algorithm for Mesh based network-on-chip. 4th International Conference on Recent Advances in Information Technology . 2018.

[11] 韦良芬, 张佑生, 王勇. 一种高吞吐低延时 NoC 容错路由算法 [J]. 安徽工业大学学报 (自然科学版),2014,31(02):195-198.

[12] 刘家俊, 顾华玺, 王长山.mesh 优先级容错路由 [J]. 计算机工程与应用,2009,45(04):105-107+114.

[13] 郭志海. 片上网络容错路由算法的研究与实现 [D]. 哈尔滨工业大学,2014.

[14] 姚磊. 片上网络无虚通道容错路由技术研究 [D]. 西安电子科技大学,2014.

[15] Chen K H, Chiu G M. Fault-tolerant routing algorithm for meshes without using virtual channels[J]. J. Inf. Sci. Eng., 1998, 14(4): 765-783.