# 模块化片上系统中高级可扩展接口的死锁避免

郭振江[①②③] 王焕东[④] 张福新[①②③] 肖俊华*[①②③]

[①](计算机体系结构国家重点实验室 北京 100190)

[②](中国科学院计算技术研究所 北京 100190)

[③](中国科学院大学计算机科学与技术学院 北京 100049)

[④](龙芯中科技术有限公司 北京 100190)

**摘 要**：模块化片上系统(MSoC)包含多个独立的IP组件及多个可能的子网络，这种异构集成的方式往往为片上网络(NoC)引入潜在的死锁。该文基于模块化异构系统MSoC研究了使用高级可扩展接口(AXI)协议的片上网络中3种类型的死锁。MSoC包含多种常见的异构组件，以及由多个独立子网络集成的片上网络，能够充分反映真实芯片的复杂性和不规则性。该文发现除环形通道导致的死锁外，基于AXI的片上网络还涉及双重路径死锁和桥接死锁。该文还提出一种两阶段算法检测片上网络中可能存在的这3种死锁。相比于通用验证方法学(UVM)随机验证，使用该算法可以将检测时长从几个月缩短到几个小时，提高片上网络的可靠性和鲁棒性。

**关键词**：片上网络；模块化片上系统；死锁避免；高级可扩展接口协议

**中图分类号**：TN47; TP302; TP303 **文献标识码**：A **文章编号**：1009-5896(2023)09-3175-09

**DOI**: 10.11999/JEIT221142

# Deadlock Avoidance of Advanced eXtensible Interface Interconnection Networks in Modular System-on-Chips

GUO Zhenjiang[①②③] WANG Huandong[④] ZHANG Fuxin[①②③] XIAO Junhua[①②③]

[①](*State Key Laboratory of Computer Architecture*, *Beijing* 100190, *China*)

[②](*Institute of Computing Technology*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*)

[③](*School of Computer Science and Technology*, *University of Chinese Academy of Sciences*, *Beijing* 100049, *China*)

[④](*Loongson Technology Corporation Limited*, *Beijing* 100190, *China*)

**Abstract**: Modular System-on-Chips (MSoC) contain several distinct IP components with possibly multiple subnetworks, resulting in potential deadlock situations for the Network on Chip (NoC). A MSoC is developed, and three deadlock cases in Advanced eXtensible Interface (AXI)-based network-on-chip are studied. MSoC consists of various common heterogeneous components, and NoC integrated by multiple independent subnetworks. MSoC can fully reflect the complexity and irregularity of real chips. NoC based on AXI is found toface double-path deadlock and bridge deadlock in addition to loop-path deadlock. A two-stage algorithm is proposed to detect those three cases. Compared to Universal Verification Methodology(UVM) random verification, this method can reduce detection time from months to hours, improving the reliability and robustness of the on-chip network.

**Key words**: Network on-Chip(NoC); Modular System on Chip (MSoC); Deadlock avoidance; Advanced eXtensible Interface(AXI) protocol

收稿日期：2022-09-01；改回日期：2023-01-16；网络出版：2023-02-03

*通信作者： 肖俊华 xiaojunhua@ict.ac.cn

基金项目：中科院战略先导项目(XDC05020000)

Foundation Item: The Strategic Priority Research Program of the Chinese Academy of Sciences (XDC05020000)

## 1 引言

模块化片上系统(Modular System on Chip, MSoC)作为一种新兴的芯片设计方法，为当前摩尔定律与算力需求之间、芯片面积与良率之间、设计复杂度与快速迭代之间的矛盾提供了突破性的解决方案[1–5]。目前，单个独立芯片的设计遇到了两方面的挑战。一是深度学习时代的算力需求每3～4月翻1番，但摩尔定律却只能每1.5～2年翻1番[6]。二是增大单一芯片面积会导致芯片良率迅速下降，并受到光刻口径限制，最终撞上“面积墙”。MSoC可以通过封装多个独立设计的芯粒(Die)提供更多功能和算力，同时较高的良率、可重复使用的设计、独立的制程工艺带来成本收益足够抵消封装开销[7]。国内外几乎同时成立产业联盟制定互连标准也促使SoC进一步向模块化发展。

表1整理了近年发布的多款处理器芯片中的IP组件情况。从中可以看出，这些芯片异构集成了不同类型的IP，通常包括多个处理器核、多核GPU、神经网络处理单元(Neural Processor Unit，NPU)以及各类定制化的组件如图像信号处理(Image Signal Processing ISP)单元、密码学加速器Crypto、数据压缩加速器等。这些IP来自不同的设计团队，多核IP内部一般还包含有互连子网络。模块化SoC灵活、复杂的结构给片上网络(Network-on-Chip, NoC)带来了新的挑战。

一方面，现有的无死锁互连研究缺少拓扑灵活性，不适用于模块化SoC。现有的研究大多假设SoC包含多个2D Mesh，并通过有限数量的边界路由器在水平或垂直方向互联[14,15]。而实际芯片往往具有明显的非对称特征，无法嵌入到Mesh当中。此外，实际芯片通常包含多个子网络，且子网络间直接连线较多，但现有的研究没有充分利用这种拓扑信息[16]。

另一方面，现有的无死锁理论大多基于虫洞路由，但高级可扩展接口(Advanced eXtensible Interface, AXI)协议与虫洞路由不完全相同。第1点不同在于，AXI协议允许同一个数据包的写请求和写数据之间存在一定的时间间隔。这种特性将引入新的死锁类型。第2点不同在于，AXI协议有5个通道。虫洞路由使用通道依赖图(Channel Dependency Graph, CDG)判断NoC是否存在死锁，但AXI网络的CDG非常复杂，并且构建过程容易出错。

因此，当前迫切需要一种针对模块化SoC的NoC无死锁设计方法学。这种方法需要满足模块化SoC架构灵活的特点，需要充分利用拓扑信息，需要简单便捷的快速建构，能够以较少的时间和人力成本检测当前结构中潜在死锁，并有对应的解决方案。

本文主要创新点包含以下3个方面：

(1) 构建了一个模拟系统模块化片上系统MSoC，本系统包含多种异构组件和多个子网络，用于研究实际系统可能遇到的死锁类型；

(2) 研究了基于AXI的NoC中的3种死锁，并分别从拓扑和路由的角度提出了解决方案；

(3) 提出自动化的死锁定位算法“先拓扑后路由”(First Topology Last Routing, FTLR)，大幅降低了硅前死锁检测的难度。将本方法应用于真实模块化SoC的构建中，成功发现了多处潜在死锁，大幅缩短了验证时间。

本文包含以下部分：第1节介绍本文的研究背景和主要贡献；第2节介绍互连网络死锁避免的相关工作，以及它们在模块化SoC上的扩展；第3节介绍本文构建的模块化SoC系统结构和实验方法；第4节介绍3种基于AXI的互连网络的死锁，提出了对应的解决方案；第5节介绍自动化死锁定位方法和实验结果；第6节总结。

## 2 相关工作

针对模块化SoC片上网络的无死锁方案大多从单个NoC的无死锁机制扩展而来，主要的理论依据是Dally理论：片上网络无死锁的充分必要条件是其通道依赖图中无环。下面介绍4种避免死锁的方法，以及如何扩展到模块化SoC。

基于日期线(Dateline)[17]的方法。该方法将跨过Dateline的数据包切换到额外的虚通道(Virtual Channel, VC)中，环形依赖被转换为两倍长度的线

表 1 近年发布的多款处理器芯片的IP组件情况

| 芯片 | CPU | GPU | NPU | 其他IP | 接口 |
|---|---|---|---|---|---|
| Tesla FSD[8] | A72×3 | G71 | NNA×2 | ISP | |
| Apple M1[9] | P Core, E Core | GPU | NPU | ISP | |
| Microsoft MT3620[10] | Cortex-A | Cortex-M | – | Crypto | DDR |
| Intel Sapphire Rapids[11,12] | Cores Mesh ×4 | – | – | DSA, QAT | DDR, PCIE |
| IBM Telum[13] | Core ×8 | – | – | Crypto, Compression | DDR, DCM |

性依赖，从而避免死锁。VC可以提高NoC的路径多样性，但会增加CDG复杂度，导致建模和分析困难。

基于转弯限制(Turn Restriction, TR)[18]的方法。该方法主要用于2D Mesh，限制数据包的转弯方向，避免形成环路。该方法可以与负载均衡、网络容错、功耗控制、近似计算[19–22]等技术结合，提供鲁棒的片上通信服务。但该方法灵活性较差，不适用于不规则结构的网络。

基于流量控制(Flow Control, FC)[23]的方法。该方法在链路中只有1个空闲缓冲区时禁止新数据包的注入，而拥有空闲缓冲区的通道不依赖于其他通道，从而避免死锁。流量控制需要额外的控制逻辑，增加了复杂度。

基于偏转路由(Deflection Routing, DR)[22,24]的方法。该方法消除了NoC中的缓冲区，因此通道之间也不存在依赖关系，从而避免死锁。该方法将收到的数据包转发到输出端口，即使是非生产路径，导致高负载时性能下降等问题。

上述4种方案主要解决单个NoC的无死锁问题。由于无死锁的不可组合性，上述方案需要一定的修改才能扩展到模块化SoC中。针对3D Mesh，已经提出了电梯优先(Elevator-First, EF)[24]方法。该方法在每个2D Mesh平面上使用维度优先路由，而在平面间使用额外的虚通道保证无死锁。针对2.5D集成的芯片，模块化转弯限制(Modular Turn Restriction, MTR)[15]将每个Die的外部抽象成一个虚拟节点，并施加转弯限制；远程控制(Remote Control, RC)[16]通过FC机制避免边界缓冲区被跨Die请求拥塞，从而消除通道间的依赖。文献[25]扩展了DR方法，设计专门的交换模块处理跨Die的请求。

然而，这些方法与如今的模块化SoC有一些差别。

拓扑方面，由于良率和散热等因素的影响，目前只有高带宽存储(High Bandwidth Memory, HBM)设备仍在使用3D集成技术[26]，模块化SoC更倾向于与2.5D集成。另外，异构IP通常仅提供唯一的访问接口，不存在多个边界路由器。最后，模块化SoC结构灵活，很少使用Mesh结构[8,10,24,25]。因此需要一种针对实际模块化SoC拓扑结构的无死锁机制。

路由方面，大多数IP使用AXI协议。上述方法都基于虫洞路由假设，而AXI协议与虫洞路由存在一些差别。其次，AXI协议具有5个通道，这极大增加了构建CDG的复杂度。因此，研究基于AXI协议的无死锁NoC具有实际意义。

表2比较了上述4种机制和本文所提出方法的适用拓扑、是否需要CDG、是否支持模块化SoC以及实现开销。本文所提方法适用于模块化SoC，面向灵活易扩展的Crossbar或者点对点结构的NoC，并且也不需要构建复杂的CDG。

基于死锁恢复的方法[27–31]允许死锁的发生，同时检测死锁并实施恢复。该方法需要对模块内部做侵入性的修改，这违反了模块化SoC的设计原则。并且根据本文实际观察到的死锁来看，没有必要为其设计复杂又昂贵的检测和处理机制。通过简单的硬件或软件修改避免死锁的出现是最有效的措施。

## 3　系统结构和实验方法

本节首先介绍实验系统MSoC的结构。该系统符合当前微处理器的主流架构，并且基于AXI协议构建其互连网络。本节还将介绍针对互连网络的实验配置和测试方法。

### 3.1　系统结构

本文构建的模块化SoC，称为MSoC，如图1所示。MSoC包含1个CPU Die、1个GPU Die和1个IO Die，这种结构能够准确概括当前主流微处理器的基本结构。

MSoC集成的3个Die由不同的团队独立设计。其中CPU Die包含2个处理器核(Core)、2个最后一级高速缓存(Last-Level Cache, LLC)、1个内存控制器接口(Double Data Rate , DDR)。CPU Die包含两级互连网络NoC-0和NoC-1。 IO Die包含高速IO接口如 高速串行计算机扩展总线标准(Peripheral Component Interconnect Express, PCIE)接口(图1中有4个)以及低速IO接口如USB和GMAC等。高速IO接口连接到NoC-2上，低速IO接口连接

表 2　死锁避免机制比较

| 方法 | 适用拓扑 | 是否需要CDG | 模块化扩展 | 实现开销 |
|---|---|---|---|---|
| Dateline | 环形 | 是 | EF[24] | 额外的虚通道 |
| TR | Mesh | 否 | MTR[15] | 限制路由方向 |
| FC | Ring, Mesh | 是 | RC[16] | 额外的流控机制 |
| DR | Ring, Mesh | 否 | 文献[25] | 额外的跨Die交换模块 |
| 本文 | Crossbar | 否 | 支持 | 几乎无 |

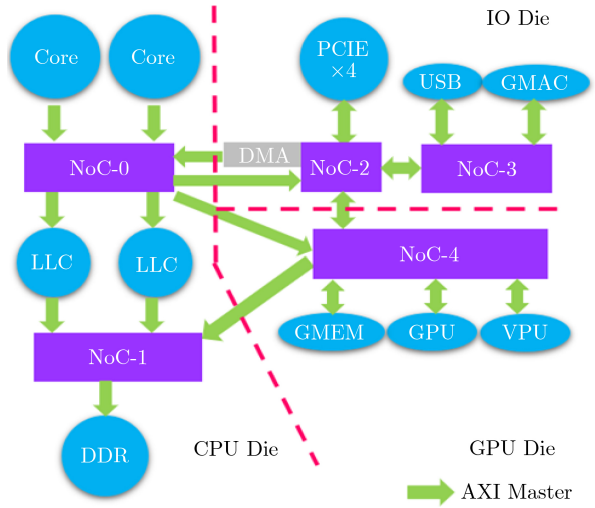



图 1 MSoC芯片架构图

到NoC-3上。GPU Die包含1个GPU、1个视频处理单元(Video Processing Unit, VPU)和显存(GPU MEMory, GMEM)。

下面介绍MSoC中的Die-to-Die互连。CPU Die通过NoC-0访问IO Die和GPU Die，访问GPU Die的通道是用于访问GMEM的快速通道。IO Die通过NoC-2访问CPU Die，此时为保持Cache一致性，需要经过DMA模块。IO Die不能质检访问GPU Die，只能转发CPU Die访问GPU Die的请求。GPU Die一方面通过直连通路快速访问DDR，另一方面复用IO Die的DMA模块进行一致性访问。

MSoC还具有Chip-to-Chip互连的能力。两片MSoC可以通过PCIE进行跨片的内存访问。

图1中所有IP核均通过AXI协议连接到NoC上，其中箭头的方向即代表AXI Master到Slave的方向。两个Core只有Master接口没有Slave接口，而DDR刚好相反，只有Slave接口没有Master接口。其他设备均包含Master接口和Slave接口。每个Master端口有唯一的ID号，用于标明数据包的发送者。

MSoC中的NoC均为Crossbar结构，并通过计算地址窗口命中的方式确定请求的转发方向。互连网络为提供标准的AXI接口用于对接IP核，包含AW, W, B, AR, R 5个通道。AXI协议是安谋(Advanced RISC Machines, ARM)公司提出的高级微控制器总线协议(Advanced Microcontroller Bus Architecture, AMBA)中的高速片内总线。相比于虫洞路由将请求头(Header)信息附在数据前发送，AXI使用独立的通道发送请求和数据，即AW通道负责携带路由信息，而W用于传输数据。每个写事务包含1个AW通道的请求，以及1个或多个W通道的数据。同一个数据包的AW和W不一定紧邻，二者之间可能间隔数个时钟周期。这是AXI协议与虫洞路由协议的重要区别。

为了保持结构尽可能简单，以及保证数据尽可能快地交付，MSoC使用死锁避免机制。由于MSoC的异构 IP核之间几乎都是点对点链接，因此现有的死锁避免方法无法直接使用。因此需要全面检查MSoC中可能出现的死锁，并给出针对性的避免方案。

### 3.2 实验方法

本文使用寄存器转换级电路(Register Transfer Level, RTL)构建了整个MSoC的片NoC。Core包含私有的一级Cache，包括32 kB的指令Cache和32 KB的数据Cache。私有Cache主要用于模拟Cache一致访问请求。LLC的大小共2 MB，分为两个Bank，采用地址位交错的方式访问。对于接收到的命中的Cache访问，LLC将直接回复数据；而对于非Cache访问和未命中访问，LLC将进一步向DDR请求数据。除此之外不会主动发起对其他设备的访问。LLC使用MSI目录维护Cache一致性。互连网络运行在1 GHz，其他模块通过同步或异步FIFO连接到片NoC上，并采用Round-Robin的方式满足服务质量(QoS)。

Core, GPU和PCIE等模块的AXI Master接口连接一个随机数据包生成器，可以生成全类型全地址空间的访问。GMEM, PCIE和DDR的每个AXI Slave接口解析数据包，并提供随机数据。每个数据包绑定一个计时器，延迟超过0负载延迟的100倍则认为网络发生了死锁。

除了验证单片MSoC无死锁，本文还假设可能通过PCIE接口连接两片MSoC，并研究这种情况下可能出现的新的死锁模式。

MSoC最大限度地还原了真实模块化SoC中主要的异构组件，以及真实模块化SoC构建时可能出现的一些问题。MSoC进行一定程度的简化以降低设计难度、提高仿真效率，但足以模拟所有可能的潜在死锁。在MSoC中，本文发现了3种类型的死锁。下一节介绍这些死锁。

## 4 3种潜在死锁及其避免

本节将介绍发现的3种死锁，即双重路径死锁、环形通道死锁和桥接死锁。对每一种死锁类型，本节分析了死锁发生的拓扑特征和路由特征。最后，本节给出了硬件和软件上的死锁避免方法。

### 4.1 双重路径死锁

在基于虫洞路由的NoC中，逻辑上的环形依赖体现为拓扑上的环形通道。而在基于AXI的互连网

络中，死锁也可能发生在具有双重路径的拓扑结构中。双重路径死锁是本文对Dally理论的重要补充。在两个节点间维持路径多样性是增加带宽、降低平均延迟的常用方法，但在基于AXI的互连网络中反而可能是导致死锁的原因。

#### 4.1.1 写通道双重路径

考虑图2中GPU与DDR之间的通路，如图2(a)所示。GPU与DDR之间存在两条通路，根据请求的Cache属性进行选择。左侧是Cache类型请求的访问通道，需要经过LLC；右侧是Uncache类型请求的专用通道，不经过LLC。图2中$n\times m$代表AXI交叉开关，$n$为交叉开关输入通道的数量，$m$为输出通道的数量。

GPU依次发送3个写请求，前两个是Cache属性的AW1, W1和AW2, W2，后一个是Uncache属性的AW3, W3。AW3通过专用通道先于AW1到达DDR，于是AW1被挡在AW3后面，需要等待W2传输完成。而此时，根据协议，不包含路由信息的W1无法越过AW1，同一通道的W2不能越过W1。于是W3被堵住无法进入右侧的专用通道，即使右侧通道没有其他数据包。因此，形成了AW1→W3→W2→W1→AW1的循环依赖，导致死锁。

从拓扑结构上看，此时存在写通道的双重路径(double write path)。造成这种死锁需要同时满足：

拓扑：两条路径上都包含独立的AW和W通道；

路由：两条路径被同时激活，即Master可以同时发出两种类型的写请求；W包含多个数据块。

#### 4.1.2 读通道双重路径

写通道可能出现双重路径死锁，因为AW通道和W通道分离。那么读通道是否有可能引发类似的死锁呢？由于AR和R通道方向不一致，因此这种情况下不会发生死锁。但如果其中一条路径涉及拆包模块，就可能触发另一种死锁。

如图2(b)所示，GPU向DDR发送两个连续的读请求AR0和AR1，这两个请求通过不同的路径路由。处于某种原因，拆包模块将AR1拆分成两个请求，即AR1-0和AR1-1。DDR按AR1-0, AR0, AR1-1的顺序接收到3个读请求，并依次回复R1-0, R0, R1-1，这将导致死锁，如图2(c)所示。R0在返回GPU时被R1-0阻挡，必须等到R1-1也传回GPU后才能继续传递。但R1-1被R0堵在DDR控制器中无法继续传递，形成了R0→R1-1→R0的循环依赖，导致死锁。需要注意的是，该问题不能通过简单地增加缓冲区来避免，因为死锁发生的原因是缓冲区之间存在依赖关系。

出现双重读路径死锁的条件包括：

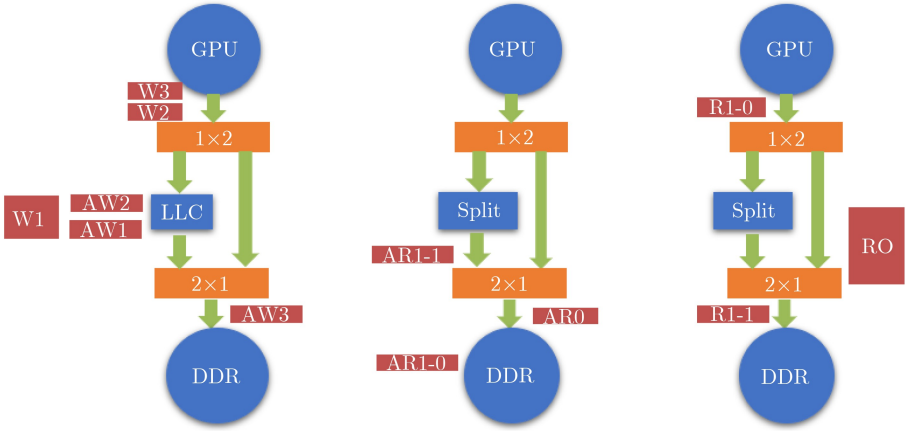
拓扑：请求端与接收端之间存在至少两条读通路(double read path)；其中一条通路存在拆包模块；

路由： Master可以同时发出两种类型的读请求，并且请求可能被拆分。

### 4.2 环形通路死锁

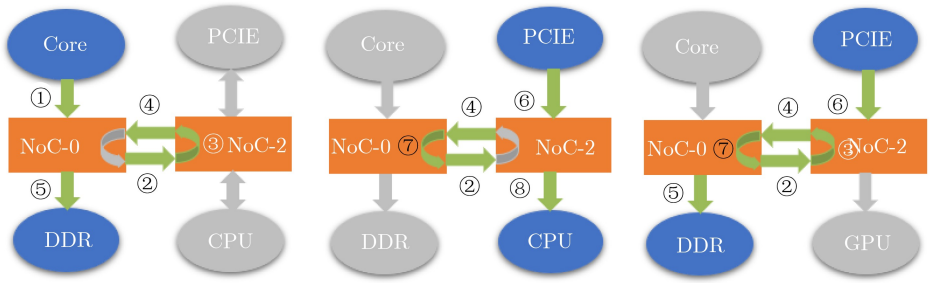
环形通路死锁是符合Dally理论的死锁类型，但实际中很少有互连网络在设计时就存在环形通路。环形通路的引入往往是缺乏全局信息的改动造成的。

考虑下面一种情况。出于某些原因，需要将原本属于GPU的一部分地址空间映射到DDR中。NoC-2作为GPU直接相连的子网络在其中增加了一处回环通道，将对这部分地址的访问请求转发回NoC-0，如图3(a)所示。图3中的数字标号代表了对应的通道。此时的访问路径为①→②→③→④→⑤。而NoC-0中恰好也存在一条回环通道，用于提




(a) 双重写路径　　(b) 双重读请求通道　　(c) 双重读数据通道

图 2 双重路径死锁图示




(a) 半环 (b) 半环 (c) 环

图 3 环形通路死锁图示

供PCIE对GPU的访问，即图3(b)中的⑤→④→⑦→②→⑧。这两种类型的流量如果不加区分地使用同一物理通道，将构成④→⑦→②→③→④的环形依赖，导致死锁。只有一种流量也可能导致死锁，即图3(c)所示的⑥→④→⑦→②→③→④→⑤。

出现环形通路死锁的条件包括：

拓扑：存在环形通路。

路由：环路的各个通道被同时占满。

### 4.3 桥接死锁

模块化SoC面向未来市场开发，其组件在设计时并不完全清楚所有可能的场景。本文使用两片MSoC互连来模拟可能的2.5D封装，并研究可能导致的死锁。本文使用PCIE连接两个芯片，但结论适用于其他非AXI协议。

桥接死锁发生在协议级[32]。Dally理论并不适用于协议级死锁，因为这种类型的死锁往往涉及模块的内部实现细节。AXI协议为请求和响应使用独立的物理通道，但PCIE协议并不区分这两者。当不同类型的数据包使用了同一通道时，就会导致协议级死锁。这类死锁只有在跨片互联时才会暴露出来；其他时候控制器可以正常工作，十分隐秘。

本文的AXI-PCIE转换器如图4所示。PCIE收集需要跨片的AXI数据包，将其转换成PCIE格式的信号传递给接收端。接收端将其还原为AXI数据包，发送到网络上。在当前时刻，由于AW1占据转换器中的缓冲区，B2无法跨片，而只有等到B1返回，AW1才会从缓冲区中离开。在另一侧PCIE中，情况类似。占据缓冲区的AW2阻挡了B1跨片，等到B2返回才能让出缓冲区。这就构成了AW1→B1→AW2→B2→AW1的循环，导致死锁。

桥接死锁发生的拓扑条件和路由条件包括：

拓扑：存在协议转换的通路，并且协议转换模块为请求和响应提供统一的缓冲区。

路由：存在大量而跨片请求。

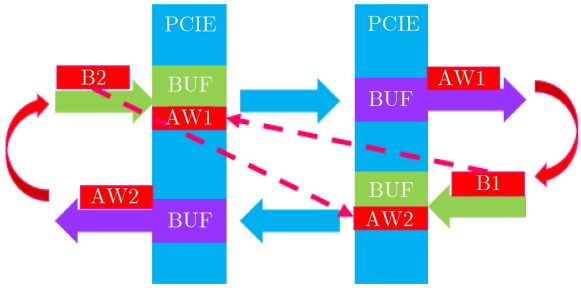



图 4 桥接死锁图示

### 4.4 死锁避免

本节讨论死锁避免的方法。

写通道双重路径死锁可以通过软件的方式避免。软件编程人员需要控制数据包的类型，保证任何时间内都只有Cache的访问或Uncache的访问。两种模式切换时需要保证所有已经发出去的请求都已完成。也可以通过硬件的方式避免，例如控制每个写请求都只包含一个写数据。但这会影响协议的性能。

读通道双重路径死锁可以通过软件或硬件的方式避免。软件上可以将控制请求的长度，用多个短请求代替单个长请求。硬件上可以在模块接入互连网络之前增加拆包模块，提前拆分数据包。

环形通道死锁需要硬件上的解决方案，单纯软件的方案可能过于复杂。各个模块单独设计时保证其内部无死锁，死锁是在模块集成和修改时引入的，这说明需要重新设计集成方案。在上述案例中，解决方法是去掉在NoC-2处的回环设计，在NoC-1中处理相关的地址路由。也可以在环中加入流控机制，避免通道中所有缓冲区被占满。

桥接死锁也需要硬件上的解决方案。PCIE控制器需要为AXI的请求和响应准备不同的缓冲区，并赋予响应更高的优先级。该方法适应与所有涉及跨片连接两个AXI网络的接口。

对于上述死锁，单纯增加缓冲区的数量并不能解决死锁，这是Dally定理已经说明的。问题不在于缓冲区的数量，而在于缓冲区之间的依赖关系。

## 5　死锁定位

基于CDG的死锁定位方法需要同时考虑AXI的5个通道之间的依赖关系，基于UVM随机验证的方法则需要大量的人力和时间。因此迫切需要一种NoC中潜在死锁的定位方法，并要求该方法可以根据设计进度随时反馈。

本节基于上述3类死锁的拓扑特征和路由特征，提出一种轻量级的检测方式，用于快速发现模块化SoC中这3种类型的死锁。

### 5.1　死锁定位算法

本文所提死锁定位算法，名为先拓扑后路由(FTLR)算法。本算法将死锁的定位过程分为两个阶段。第1阶段该算法进行拓扑分析，标识可能造成死锁的数据通路。第2阶段该算法进行路由分析，并与拓扑路由进行匹配。如果匹配成功则说明存在潜在死锁。下面介绍本算法的细节。

第1阶段，FTLR将有向图$\boldsymbol{G}$作为输入，输出中间结果集$\boldsymbol{M}$。$\boldsymbol{G}$中的点是NoC中模块或接口，从模块$A$直接连接到模块$B$的通道表示为从点$A$到点$B$的有向边。中间结果集$\boldsymbol{M}$中记录着从任一Master接口到任一Slave接口的所有数据通路，并且特殊标记其中的拆包模块和桥接模块。在这一阶段，FTLR使用广度优先算法处理图$\boldsymbol{G}$。

第2阶段，FTLR处理路由规则集$\boldsymbol{F}$和中间结果集$\boldsymbol{M}$。$\boldsymbol{F}$通常以表格的形式提供，规定了不同地址空间、不同类型请求的终点设备和访问路径。FTLR在$\boldsymbol{F}$和$\boldsymbol{M}$中进行匹配，检查$\boldsymbol{M}$中路由能否被同时激活，或满足其他死锁发生条件。如果匹配成功，则定位到死锁。FTLR算法的流程整理在算法1中。

图$\boldsymbol{G}$可以很容易地在NoC的设计文档中找到，并且可以随设计改动而迅速调整。FTLR第1阶段的算法复杂度为$O(N_2)$，第2阶段的路由表$\boldsymbol{F}$规模也为$O(N_2)$，其中$N$为图$\boldsymbol{G}$的顶点数。在实际场景中，由于设计阶段复杂性限制和制造阶段成本限制，芯片中的组件数不会很大，$N$通常会被控制在30以内。这一点也可以从表1中得到体现。因此尽管FTLR算法的复杂度为$O(N_2)$，也完全可以在几秒钟的时间内完成，这在以年计算的芯片设计周期中占比微乎其微。

### 5.2　实验

将本文所提检测方法应用于MSoC的互连网络无死锁验证中，并与CDG和UVM的方法做对比。UVM采用全系统随机验证，分批进行4 500轮测试，每轮测试攻击发送125 000数据包，数据包总量超过$5.6\times10^9$次。最终发现双重路径死锁3处、环形通路型死锁1处以及桥接型死锁2处。测试结果整理在表3中。

在死锁覆盖率方面，UVM和FTLR方法可以检测出所有的死锁类型，CDG方法无法覆盖双重通道死锁和桥接死锁。在检测效率方面，CDG和FTLR方法可以同时报告所有死锁，而UVM需要逐个报告，并且需要人工调试波形才能发现死锁。在建模难度方面，CDG方法最为复杂，并且很容易出错；UVM难度一般，只需要RTL代码即可；而FTLR只需要根据MSoC的芯片架构图就可以完成建模，最为简单。在检测时间方面，UVM方法需要长时间的运行测试并且需要人工调试波形，因此可能需要几周的时间；CDG方法需要复杂的建模，因此可能需要几天的时间用于建模；而FTLR方法只需要几个小时就可以完成建模到检测的全过程。本文方法大幅减少了验证时间和人力投入，并且可以随时根据网络结构进行调整，具有一定的灵活性。

FTLR算法可以有效地发现基于AXI的互连网

**算法1 FTLR算法**

输入：拓扑结构图$\boldsymbol{G}$，路由规则集$\boldsymbol{F}$。

输出：潜在死锁依赖$P$。

变量：中间结果集合$\boldsymbol{M}$。

步骤1　读入$\boldsymbol{G}$，检测$\boldsymbol{G}$中是否存在环。如果存在，则将其加入到$\boldsymbol{M}$中。

步骤2　遍历$\boldsymbol{G}$中所有Master到Slave的数据通路，加入到$\boldsymbol{M}$中，并标识拆包模块和桥接模块。

步骤3　遍历$\boldsymbol{F}$中所有Master到Slave的路由通路，如果与$\boldsymbol{M}$中某一数据通路组合满足死锁发生条件，则输出该路径。

**表 3　互连网络死锁检测方法对比**

| 方法 | 双重通道死锁 | 环形通道死锁 | 桥接死锁 | 检测效率 | 建模难度 | 检测时间 |
|---|---|---|---|---|---|---|
| CDG | × | √ | × | 并行 | 困难 | 几天 |
| UVM | √ | √ | √ | 串行 | 一般 | 几周 |
| FTLR | √ | √ | √ | 并行 | 简单 | 几小时 |

络中潜在的死锁。相比传统的验证方法，基于拓扑特征和路由特征的FTLR算法发现死锁的速度更快、成本更低、结果更全，可以提高互连网络的鲁棒性，并且可以在SoC构建过程中反复执行，随时发现问题。

## 6 结束语

模块化SoC的设计理念正在被越来越多的处理器厂商使用，这给基于AXI的互连网络带来了新的无死锁挑战。首先，AXI协议不同于传统的虫洞路由，需要对原有理论的扩展。其次，多种功能的组件在集成时经常会发生改动和调整，这可能在原本无死锁的互连子网络之间引入新的死锁，为此需要全程的、增量的、灵活的监控方案。最后，为了避免在多芯片跨片互连或2.5D集成时引入协议级死锁，协议转换、拆包模块等特殊组件需要预留足够的资源。

本文基于真实异构SoC构建了死锁分析模型MSoC，系统性地研究了基于AXI的互连网络在模块化集成过程中可能引入的潜在死锁。本文提出了3种死锁，即双重通道死锁、环形通道死锁和桥接死锁，研究了它们的拓扑特征和路由特征，并提出了对应的死锁避免方案。本文进一步构建了自动化的定位工具全面排查和监控互连网络潜在的死锁问题，极大地缩短了验证时间、提高了验证效率。本文扩展了传统的Dally理论，填补了AXI互连无死锁理论的空白，为模块化SoC的设计提供了帮助。

## 参 考 文 献


[1] NAFFZIGER S, BECK N, BURD T, *et al*. Pioneering chiplet technology and design for the AMD EPYC™ and Ryzen™ processor families: Industrial product[C]. 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), Valencia, Spain, 2021: 57–70. doi: 10.1109/ISCA52012.2021.00014.

[2] BECK N, WHITE S, PARASCHOU M, *et al*. 'Zeppelin': An SoC for multichip architectures[C]. 2018 IEEE International Solid - State Circuits Conference - (ISSCC), San Francisco, USA, 2018: 40–42. doi: 10.1109/ISSCC.2018.8310173.

[3] ARUNKUMAR A, BOLOTIN E, CHO B, *et al*. MCM-GPU: Multi-chip-module GPUs for continued performance scalability[C]. The 44th Annual International Symposium on Computer Architecture, Toronto, Canada, 2017: 320–332. doi: 10.1145/3079856.3080231.

[4] VIJAYARAGHAVAN T, ECKERT Y, LOH G H, *et al*. Design and analysis of an APU for exascale computing[C]. 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), Austin, USA, 2017: 85–96. doi: 10.1109/HPCA.2017.42.

[5] MA Xiaohan, WANG Ying, WANG Yujie, *et al*. Survey on chiplets: Interface, interconnect and integration methodology[J]. *CCF Transactions on High Performance Computing*, 2022, 4(1): 43–52. doi: 10.1007/s42514-022-00093-0.

[6] WANG Mengdi, WANG Ying, LIU Cheng, *et al*. Network-on-interposer design for agile neural-network processor chip customization[C]. 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, USA, 2021: 49–54. doi: 10.1109/DAC18074.2021.9586261.

[7] FENG Yinxiao and MA Kaisheng. Chiplet actuary: A quantitative cost model and multi-chiplet architecture exploration[C]. The 59th ACM/IEEE Design Automation Conference, San Francisco, USA, 2022: 121–126. doi: 10.1145/3489517.3530428.

[8] TALPES E, SARMA D D, VENKATARAMANAN G, *et al*. Compute solution for Tesla's full self-driving computer[J]. *IEEE Micro*, 2020, 40(2): 25–35. doi: 10.1109/MM.2020.2975764.

[9] [EB/OL].https://www.anandtech.com/show/17019/apple-announced-m1-pro-m1-max-giant-new-socs-with-allout-performance.

[10] STILES D. The hardware security behind azure sphere[J]. *IEEE Micro*, 2019, 39(2): 20–28. doi: 10.1109/MM.2019.2898633.

[11] BISWAS A. Sapphire rapids[C]. 2021 IEEE Hot Chips 33 Symposium (HCS), Palo Alto, USA, 2021: 1–22. doi: 10.1109/HCS52781.2021.9566865.

[12] ROTEM E, MANDELBLAT Y, BASIN V, *et al*. Alder lake architecture[C]. 2021 IEEE Hot Chips 33 Symposium (HCS), Palo Alto, USA, 2021: 1–23. doi: 10.1109/HCS52781.2021.9567097.

[13] JACOBI C. Real-time AI for enterprise workloads: The IBM Telum processor[C]. 2021 IEEE Hot Chips 33 Symposium (HCS), Palo Alto, USA, 2021: 1–22. doi: 10.1109/HCS52781.2021.9567422.

[14] ZHENG Hao, WANG Ke, and LOURI A. Adapt-NoC: A flexible network-on-chip design for heterogeneous manycore architectures[C]. 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Korea, 2021: 723–735. doi: 10.1109/HPCA51647.2021.00066.

[15] MAJUMDER P, KIM S, HUANG J, *et al*. Remote control: A simple deadlock avoidance scheme for modular systems-on-chip[J]. *IEEE Transactions on Computers*, 2021, 70(11): 1928–1941. doi: 10.1109/TC.2020.3029682.

[16] DALLY W J and TOWLES B. Principles and Practices of Interconnection Networks[M]. San Francisco: Morgan Kaufmann Publishers, 2004.

[17] EBRAHIMI M and DANESHTALAB M. EbDa: A new theory on design and verification of deadlock-free interconnection networks[C]. The 44th Annual International Symposium on Computer Architecture, Toronto, Canada, 2017: 703–715. doi: 10.1145/3079856.3080253.

[18] LI Ming, ZENG Qing'an, and JONE W B. DyXY: A proximity congestion-aware deadlock-free dynamic routing method for network on chip[C]. The 43rd Annual Design Automation Conference, San Francisco, USA, 2006: 849–852. doi: 10.1145/1146909.1147125.

[19] FU Binzhang, HAN Yinhe, MA Jun, *et al*. An abacus turn model for time/space-efficient reconfigurable routing[C]. The 38th Annual International Symposium on Computer Architecture, San Jose, USA, 2011: 259–270. doi: 10.1145/2000064.2000096.

[20] SAMIH A, WANG Ren, KRISHNA A, *et al*. Energy-efficient interconnect via router parking[C]. 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA), Shenzhen, China, 2013: 508–519. doi: 10.1109/HPCA.2013.6522345.

[21] WANG Ling, WANG Yadong, and WANG Xiaohang. An approximate multiplane network-on-chip[C]. 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2020: 234–239. doi: 10.23919/DATE48585.2020.9116377.

[22] RAMRAKHYANI A and KRISHNA T. Static bubble: A framework for deadlock-free irregular on-chip topologies[C]. 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), Austin, USA, 2017: 253–264. doi: 10.1109/HPCA.2017.44.

[23] XIA Jing, CHENG Chuanning, ZHOU Xiping, *et al*. Kunpeng 920: The first 7-nm chiplet-based 64-core ARM SoC for cloud services[J]. *IEEE Micro*, 2021, 41(5): 67–75. doi: 10.1109/MM.2021.3085578.

[24] DUBOIS F, SHEIBANYRAD A, PÉTROT F, *et al*. Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3D-NoCs[J]. *IEEE Transactions on Computers*, 2013, 62(3): 609–615. doi: 10.1109/TC.2011.239.

[25] WANG Tianqi, FENG Fan, XIANG Shaolin, *et al*. Application defined on-chip networks for heterogeneous chiplets: An implementation perspective[C]. 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Korea, 2022: 1198–1210. doi: 10.1109/HPCA53966.2022.00091.

[26] HU Xing, STOW D, and XIE Yuan. Die stacking is happening[J]. *IEEE Micro*, 2018, 38(1): 22–28. doi: 10.1109/MM.2018.011441561.

[27] LANKES A, WILD T, HERKERSDORF A, *et al*. Comparison of deadlock recovery and avoidance mechanisms to approach message dependent deadlocks in on-chip networks[C]. 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip, Grenoble, France, 2010: 17–24. doi: 10.1109/NOCS.2010.11.

[28] ANJAN K V and PINKSTON T M. An efficient, fully adaptive deadlock recovery scheme: DISHA[C]. The 22nd Annual International Symposium on Computer Architecture, Santa Margherita Ligure, Italy, 1995: 201–210. doi: 10.1145/223982.224431.

[29] RAMRAKHYANI A, GRATZ P V, and KRISHNA T. Synchronized progress in interconnection networks (SPIN): A new theory for deadlock freedom[C]. 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), Los Angeles, USA, 2018: 699–711. doi: 10.1109/ISCA.2018.00064.

[30] PARASAR M, FARROKHBAKHT H, JERGER N E, *et al*. DRAIN: Deadlock removal for arbitrary irregular networks[C]. 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), San Diego, USA, 2020: 447–460. doi: 10.1109/HPCA47549.2020.00044.

[31] PARASAR M, JERGER N E, GRATZ P V, *et al*. SWAP: Synchronized weaving of adjacent packets for network deadlock resolution[C]. The 52nd Annual IEEE/ACM International Symposium on Microarchitecture, Columbus, USA, 2019: 873–885. doi: 10.1145/3352460.3358255.

[32] JERGER N E, KRISHNA T, and PEH L S. On-Chip Networks[M]. 2nd ed. Cham: Springer, 2017.

郭振江：男，博士生，研究方向为计算机体系结构、片上网络.

王焕东：男，博士，高级工程师，研究方向为计算机体系结构.

张福新：男，博士，正高级工程师，研究方向为计算机体系结构、二进制翻译.

肖俊华：男，博士，高级工程师，研究方向为计算机体系结构、高性能处理器设计、性能分析和评估.

责任编辑：余　蓉