ICS 31.200
CCS L 55

T/CCIASC

中 国 计 算 机 行 业 协 会 团 体 标 准

T/CCIASC XXXX—XXXX

人工智能芯片 面向芯粒的卡间互联接口技术要求

Artificial intelligence chips - technical requirements of inter-card interface for chiplets

XXXX－XX－XX 发布　　XXXX－XX－XX 实施

中国计算机行业协会　　发 布

目　　次

前　　言

本文件按照GB/T 1.1-2020《标准化工作导则 第1部分 标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利，本文件的发布机构不承担识别专利的责任。

本文件由中国计算机行业协会提出。

本文件由中国计算机行业协会归口。

本文件起草单位：新华三技术有限公司、中国电子技术标准化研究院、中国信息通信研究院、上海壁仞科技股份有限公司、沐曦集成电路（上海）股份有限公司、格通智联技术（上海）有限公司、格创通信（浙江）有限公司、上海天数智芯半导体股份有限公司、海光信息技术有限公司、太初（无锡）电子科技有限公司、北京曦望芯科智能科技有限公司、北京谦合益邦云信息技术有限公司、上海合见工业软件集团有限公司、芯耀辉科技股份有限公司、上海晟联科半导体有限公司、芯潮流（珠海）科技有限公司

本文件主要起草人：朱仕银、刘新民、万晓兰、贾琳琳、刘畅、尹航、李峰、聂一、张乾、邸绍岩、王骏成、雷恺、魏莉、曾敏、李军军、丁同浩、孙志峰、罗彬、赵畅、杨朋霖、郑卫华、付庆平、董剑、于彬、孔宁、司照凯、曹宜宁。

人工智能芯片 面向芯粒的卡间互联接口技术要求

1 范围

本文件规定了面向芯粒的卡间互联接口技术要求，包括接口各层（协议层、链路层、物理层）要求，通信性能要求以及其它要求。

本文件适用于加速器与通信芯粒互联研究、设计、开发、测试等。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 9178 集成电路术语

GB/T 14113 半导体集成电路封装术语

UCIe UCIe规范v2.0（Specification Revision 2.0 ）

AXI 协议规范（ AXI Protocol Specification）

AXI-Stream 协议规范（ AXI-Stream Protocol Specification）

3 术语和定义

GB/T 9178、GB/T 14113和GB/T 46280.1界定的以及下列术语和定义适用于本文件。

3.1

互联 interconnection

在芯粒间的物理连接的基础上,使用通信协议协调调度两端实现信息交互的连接线路。

［来源：GB/T 46280.1-2025 3.6］

4 缩略语

下列缩略语适用于本文件。

APB	Advanced Peripheral Bus	先进外设接口
AP	Advanced Package	先进封装
BER	Bit Error Rate	误码率
ECC	Error Checking and Correcting	错误侦测与纠正
EOB	End Of Burst	突发结束
EOP	End Of Packet	包结束
FDI	Flit-aware Data Interface	Flit感知数据接口

GPU	Graphics Processing Unit	图形处理单元
GPIO	General Purpose Input Output	通用输入输出
IGPH	Input from GPU Header	协议层从GPU接收报文头
MCM	Multi-Chip Module	多芯片组件
OGPH	Output to GPU Header	协议层向GPU输出报文头
PI	Power Integrity	电源完整性
RDI	Raw Data Interface	原始数据接口
SI	Signal Integrity	信号完整性
SOB	Start Of Burst	突发起始
SOP	Start Of Packet	包起始
SP	Standard Package	标准封装
UCIe	Universal Chiplet Interconnect Express	通用芯粒接口
UMAC	Unpacking/Packing MAC layer	类MAC层打包/解包模块

5 概述

5.1 场景

本文件适用于加速器和通信芯粒合封场景，以GPU互联互通为示例，适用于各种加速器。如图1所示：

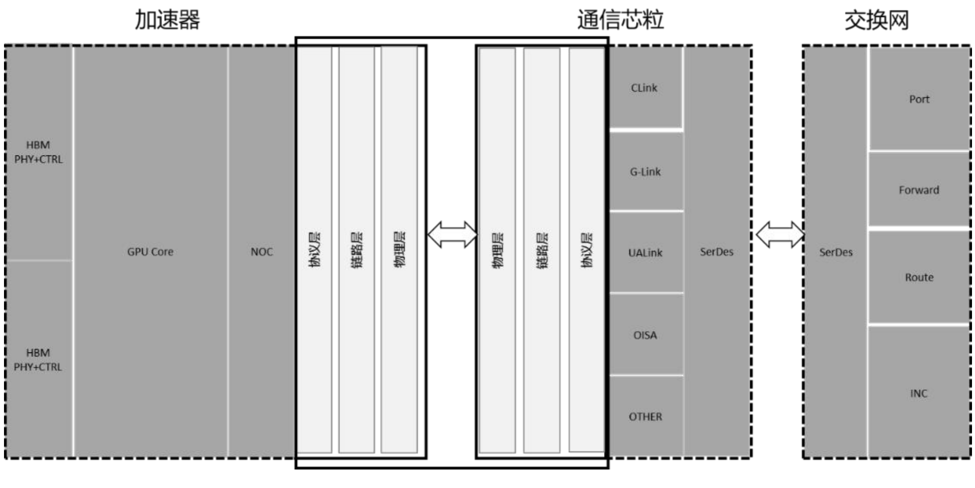


图 1 逻辑功能框架

GPU通过物理层、链路层和协议层与通信芯粒的对应层互联互通，通信芯粒所采用的网络协议不在本文件规定范围内。

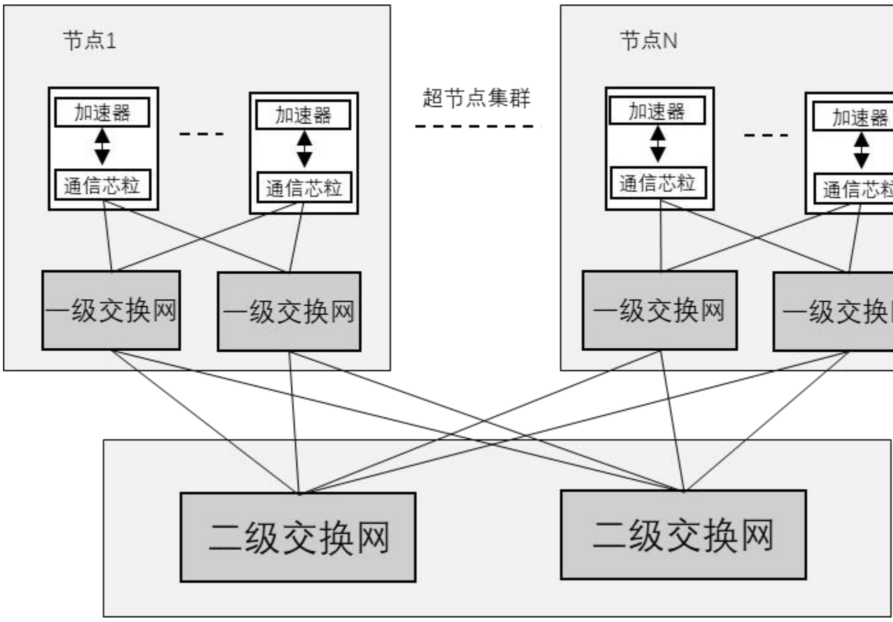


图 2　通信芯粒互联交换网

加速器和交换网互联网络架构如图2所示。

5.2　接口

在一个典型的面向芯粒的卡间互联系统中，加速器是主要的算力芯片，通过合封通信芯粒实现卡间的高速互联；通信芯粒为算力芯片提供高速网络接入能力，实现卡间互联，通过互联交换网完成高效转发，从而实现加速器卡间的互通。

如图2所示：加速器和通信芯粒之间应支持如下接口：

a） 接口1：加速器NOC与协议层之间的接口，其主要作用是完成加速器NOC网络的南向数据传输和协议转换。这类接口通常采用标准总线实现，例如AXI或AXI-Stream等；

b） 接口2：加速器和通信芯粒间物理层互联接口，主要完成物理层互联互通，本文件要求该接口兼容UCIe2.0规范。

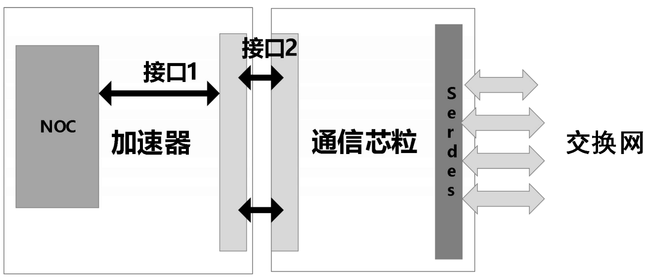


图 3　接口类型

5.3　拓扑

组网拓扑符合以下要求：

a） 通信芯粒应支持与交换网互联；

b) 应支持一级交换网络，如图4所示拓扑；

c) 宜支持二级交换网络，如图5所示拓扑。

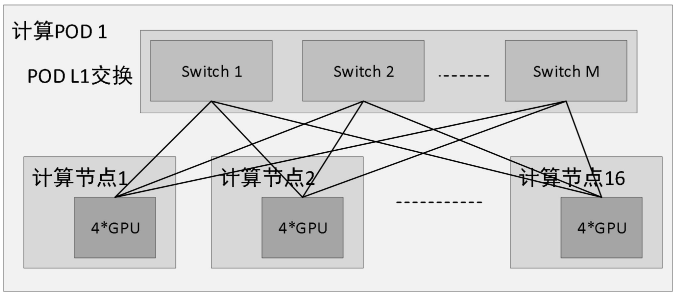


图 4 一级交换网互联拓扑

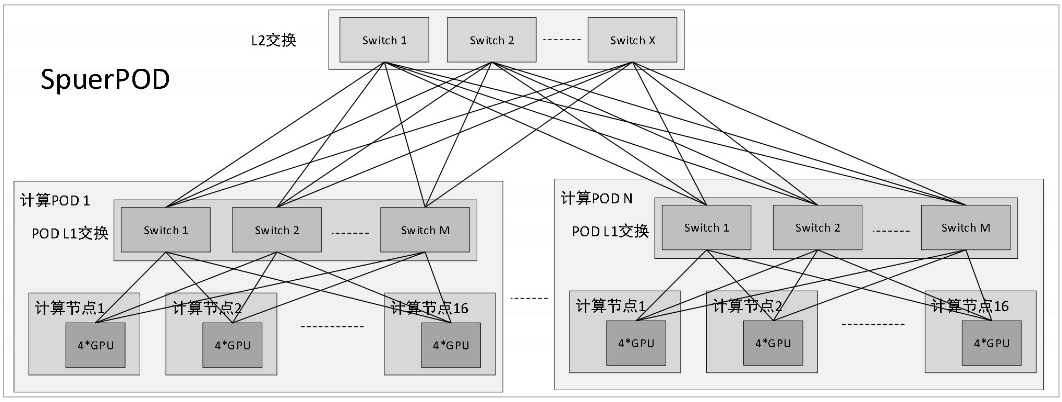


图 5 二级交换网互联拓扑

6 总体要求

6.1 架构

本文件所规定的技术要求应遵循图6的架构设计，加速器与通信芯粒合封，通信芯粒提供接入交换网的能力，通过通信芯粒和交换网组建的网络，完成加速器卡间互联互通。主要包含如下技术功能定义：

a) 协议层功能

定义互联互通的协议层接口，应遵循开放总线标准AXI协议规范和AXI-Stream协议规范；

b) 链路层功能

定义链路层特性，FDI工作频率，CRC校验和重传，链路协商和状态管理，帧格式处理等,应兼容UCIe规范2.0；

c) 物理层功能

定义物理层封装技术，先进封装，标准封装的特性要求，应兼容UCIe规范v2.0；

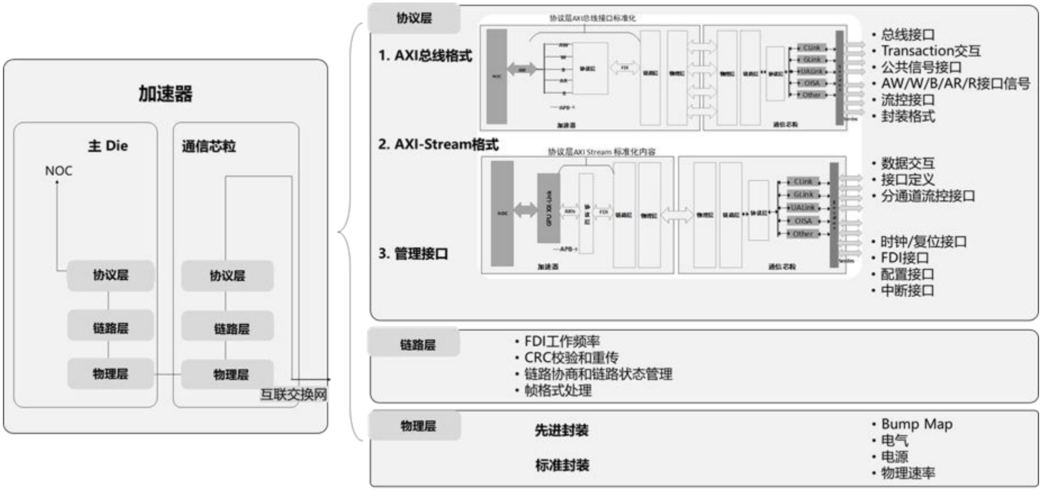


图 6　总体架构

6.2　通信语义

通信语义应支持：

a）同步内存语义：提供基于统一地址编址的内存空间,支持加速器指令级访问统一编址内存空间，如：Load/Store等指令；

b）异步内存语义：支持加速器卡间的直接内存拷贝功能，将大批量数据从一个加速器拷贝到另一个加速器的显存,宜采用AXI-Stream协议规范完成异步内存拷贝。

7　接口各层要求

7.1　协议层要求

协议层应遵循AXI协议规范和AXI-Stream协议规范，即图3接口1的要求：

a)　AXI模式,应遵循高级微控制器总线架构 AXI 协议规范节定义内容，通过AXI实现GPU和通信芯粒之间协议层的互通，协议完成总线事务和通信芯粒可识别数据格式转换，保证GPU从NOC到总线事务，通过定义的协议打包发送到通信芯粒，通信芯粒解析路由信息，选择网络技术，完成网络转发到目的GPU。

b)　AXI-Stream模式，应遵循高级微控制器总线架构 AXI-Stream 协议规范所定义内容，GPU卡基于当前现有的软件和互联协议，可以直接承载在AXI-Stream模式上，AXI-Stream数据直接转到通信芯粒路由转发。

7.1.1　AXI 总线格式

AXI总线互通，如图7所示GPU侧五个独立事务通道，通道的事务信息封装转换成网络报文，与通信芯粒互通处理。

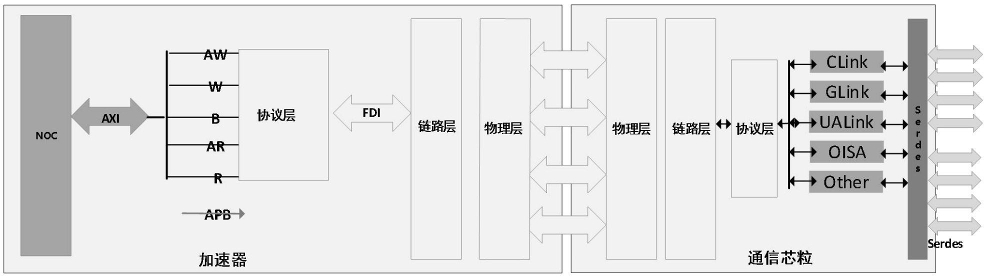


图 7　AXI 总线结构

7.1.1.1　AXI 总线接口

协议层提供AXI总线接口模式，遵循标准AXI协议，定义了五个独立的事务通道，分别是写地址通道（AW）、写数据通道（W）、写响应通道（B）、读地址通道（AR）、读数据通道（R），每个事务通道主动发出事务时作为主接口，接收事务是作为从接口。

GPU的AXI总线直接对接协议层提供的AXI接口，完成GPU侧事务转换到网络报文，透传到通信芯粒完成该笔事务跨GPU的转发。协议层提供的逻辑模块本文件中简称UMAC。

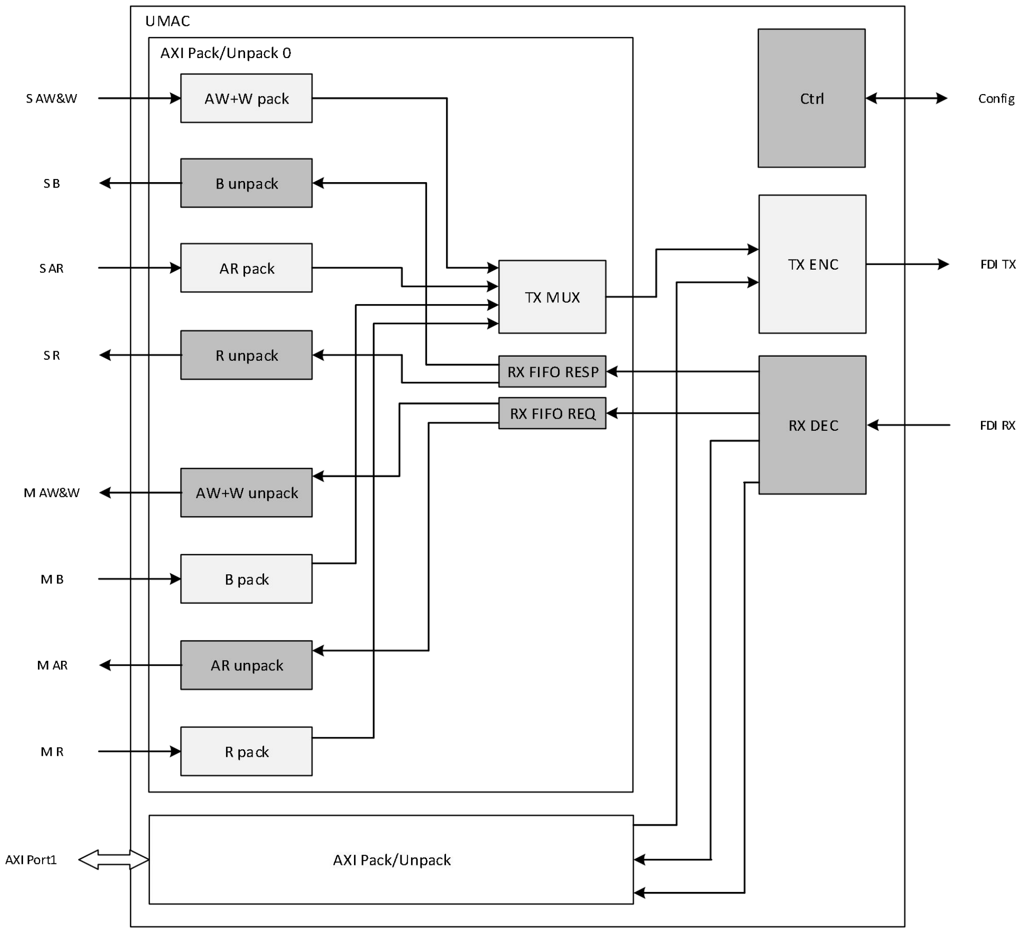


图 8　UMAC 功能图示

图8展示每两组AXI端口对接一路FDI端口的框图示意,实现四个AXI端口数据传输到通信芯粒需要两组图示逻辑。

7.1.1.2 事务交互

交互内容应遵循如下要求:

发送方向:输入事务->打包成为内部使用格式->填入Flit->由FDI发送。

接收方向:FDI输入数据->跨时钟域->从Flit中解出内部使用格式->恢复成原始事务输出。

7.1.1.3 通用报文封装格式

通信芯粒在初始化时,应配置本GPU ID信息,用于通信芯粒网络通信标识,如:发送时本GPU ID作为源GPU ID信息,接收端通信芯粒使用网络传输的源GPU ID,用来提供给本地GPU识别和记录源GPU ID信息。

在GPU到通信芯粒方向上,发送侧UMAC模块将目的信息目的GPU ID和目的PORT ID,打包到IPGH报文头,该笔事务包含的其他控制信息打包到相应类型的AXI报文头上,如果有数据要传输,则以Payload形式附加在AXI报文头之后传递。打包好的报文如下图9所示。通信芯粒通过IGPH的目的GPU ID和目的PORT ID,配合使用本地配置的源GPU ID组装网络转发报文头信息到目的通信芯粒。

在通信芯粒到GPU方向上,网络报文头会携带目的GPU ID和源GPU ID信息,在经过通信芯粒向GPU传递时,使用OGPH 报文头,包含源信息源GPU ID和源PORT ID,接收侧UMAC模块将报文解出恢复成原来的事务信息。总体格式入图9所示:

IGPH/ OGPH	AXI Header	Payload/Rsvd

图 9 AXI 报文封装格式

各通道的AXI报文头封包格式见下面7.1.1.4至7.1.1.7的章节所述。

7.1.1.4 AXI AW+W 通道

通信芯粒侧发起AW请求主接口侧信号定义见表1,接收AW请求从接口侧信号定义见表2,用于和GPU侧AW通道互通使用。

表 1 AW 主接口

接口信号	接口位宽	接口方向	功能描述
AXI_M_AWVALID_0-3	1	O	AWVALID to NOC
AXI_M_AWREADY_0-3	1	I	AWREADY from NOC
AXI_M_AWID_0-3	ID_WIDTH	O	AWID to NOC
AXI_M_AWLEN_0-3	6	O	AWLEN to NOC
AXI_M_AWUSER_0-3	USER_REQ_WIDTH	O	AWUSER to NOC

			1bit: indicates need to pack WSTRB
AXI_M_AWCACHE_0-3	4	O	AWCACHE to NOC
AXI_M_AWADDR_0-3	ADDR_WIDTH	O	AWADDR to NOC only 51 bits valid for ADDR,other 13bits optional:10bits: Source GPU ID 3bits: Source PORT ID
AXI_M_AWLOCK_0-3	1	O	AWLOCK to NOC

表 2　AW 从接口

接口信号	接口位宽	接口方向	功能描述
AXI_S_AWVALID_0-3	1	I	AWVALID from NOC
AXI_S_AWREADY_0-3	1	O	AWREADY to NOC
AXI_S_AWID_0-3	ID_WIDTH	I	AWID from NOC
AXI_S_AWLEN_0-3	6	I	AWLEN from NOC
AXI_S_AWUSER_0-3	USER_REQ_WIDTH	I	AWUSER from NOC 1bit: indicates need to pack WSTRB
AXI_S_AWCACHE_0-3	4	I	AWCACHE from NOC
AXI_S_AWADDR_0-3	ADDR_WIDTH	I	AWADDR from NOC 10bits: Destination GPU ID 3bits: Destination PORT ID
AXI_S_AWLOCK_0-3	1	I	AWLOCK from NOC

通信芯粒侧W接口信号,主接口侧信号定义见表3,从接口侧信号定义表4, 用于和GPU侧W通道互通使用。

表 3　W 主接口

接口信号	接口位宽	接口方向	功能描述
AXI_M_WVALID_0-3	1	O	WVALID to NOC
AXI_M_WREADY_0-3	1	I	WREADY from NOC
AXI_M_WSTRB_0-3	DATA_WIDTH / 8	O	WSTRB to NOC
AXI_M_WLAST_0-3	1	O	WLAST to NOC
AXI_M_WPOISON_0-3	1	O	Reserved, not used in AXI mode.
AXI_M_WDATA_0-3	DATA_WIDTH	O	WDATA to NOC

表 4　W 从接口

接口信号	接口位宽	接口方向	功能描述
AXI_S_WVALID_0-3	1	I	WVALID from NOC
AXI_S_WREADY_0-3	1	O	WREADY to NOC
AXI_S_WSTRB_0-3	DATA_WIDTH / 8	I	WSTRB from NOC
AXI_S_WLAST_0-3	1	I	WLAST from NOC
AXI_S_WPOISON_0-3	1	I	Reserved, not used in AXI mode.
AXI_S_WDATA_0-3	DATA_WIDTH	I	WDATA from NOC

对于有效数据足够长的情况，无需padding，将第一拍低位和最后一拍高位的无效Byte挤除后，WDATA首尾相接作为Payload。打包好的报文如图10所示：

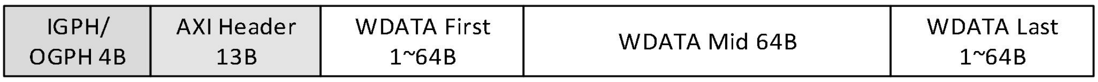


图 10　AW+W(无空洞)报文示意

对于有效数据不够长的情况，会将AXI报文头上的PADDING置1，将包补足到60Byte，打包好的报文如图11。由于Unaligned transfer的存在，有效的WDATA可能来自一拍也能来自两拍。

图 11　AW+W(打包)报文示意

Aligned transfer指AWADDR[5:0]为全0，起始数据对齐总线边界，且全部需要传输的有效数据连续，无需打包WSTRB的传输，如图12。

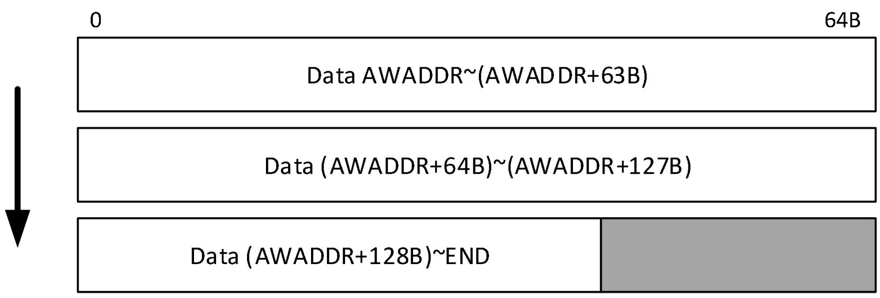


图 12　对齐传输示意

Unaligned transfer指AWADDR[5:0]不为全0，起始数据不对齐总线边界，但全部需要传输的有效数据连续，无需打包WSTRB的传输。对端通过AWADDR将数据恢复到WDATA上的原有位置，如图13。

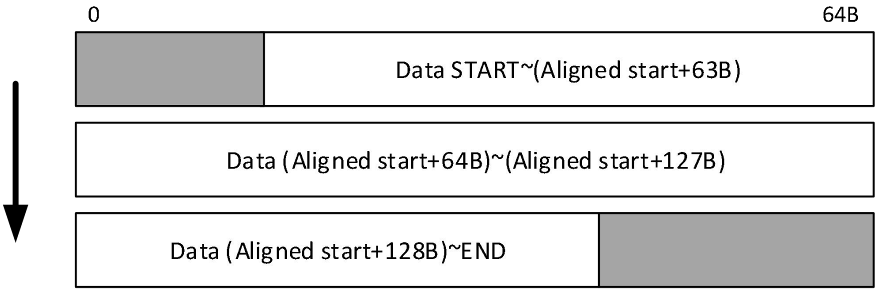


图 13　非对齐传输示意

若需要传输的有效数据地址不连续，存在空洞，则需要透传WDATA+WSTRB供对端使用，如图14和图15所示。

图 14　AW+W(带空洞)报文示意

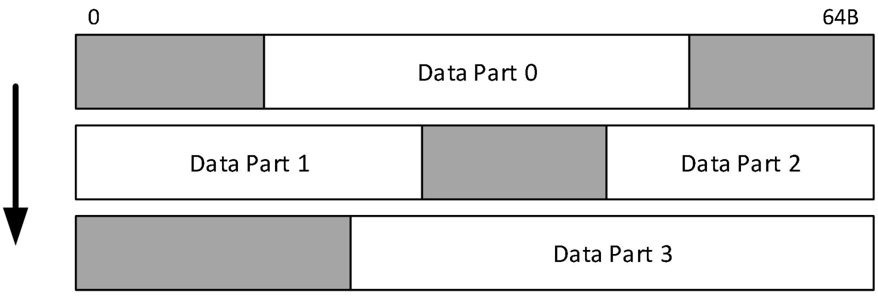


图 15　空洞传输示意

7.1.1.5　AXI B 通道

通信芯粒侧B通道应答主接口侧信号定义见表5,B通道从接口侧信号定义见表6，用于和GPU侧B通道接口互通使用。

表 5　B 主接口

接口信号	接口位宽	接口方向	功能描述
AXI_M_BVALID_0-3	1	I	BVALID from NOC
AXI_M_BREADY_0-3	1	O	BREADY to NOC
AXI_M_BID_0-3	ID_WIDTH	I	BID from NOC 10bits: Destination GPU ID 3bits: Destination PORT id
AXI_M_BRESP_0-3	2	I	BRESP from NOC

AXI_M_BUSER_0-3	USER_RESP_WIDTH	I	BUSER from NOC

表 6　B 从接口

接口信号	接口位宽	接口方向	功能描述
AXI_S_BVALID_0-3	1	O	BVALID to NOC
AXI_S_BREADY_0-3	1	I	BREADY from NOC
AXI_S_BID_0-3	ID_WIDTH	O	BID to NOC 10bits: Destination GPU ID 3bits:Destination PORT id
AXI_S_BRESP_0-3	2	O	BRESP to NOC
AXI_S_BUSER_0-3	USER_RESP_WIDTH	O	BUSER to NOC

B通道没有payload，封装好的报文全长60Bytes。

7.1.1.6　AXI AR 通道

通信芯粒侧发起AR请求主接口侧信号定义见表7,接收AR请求从接口侧信号定义见表8，用于和GPU侧AR通道互通使用。

表 7　AR 主接口

接口信号	接口位宽	接口方向	功能描述
AXI_M_ARVALID_0-3	1	O	ARVALID to NOC
AXI_M_ARREADY_0-3	1	I	ARREADY from NOC
AXI_M_ARID_0-3	ID_WIDTH	O	ARID to NOC
AXI_M_ARLEN_0-3	6	O	ARLEN to NOC
AXI_M_ARUSER_0-3	USER_REQ_WIDTH	O	ARUSER to NOC
AXI_M_ARCACHE_0-3	4	O	ARCACHE to NOC
AXI_M_ARADDR_0-3	ADDR_WIDTH	O	ARADDR to NOC only 51 bits valid for ADDR,other 13bits optional: 10bits: Source GPU ID 3bits: Source PORT ID
AXI_M_ARLOCK	1	O	ARLOCK to NOC

表 8　AR 从接口

接口信号	接口位宽	接口方向	功能描述
AXI_S_ARVALID_0-3	1	I	ARVALID from NOC
AXI_S_ARREADY_0-3	1	O	ARREADY to NOC
AXI_S_ARID_0-3	ID_WIDTH	I	ARID from NOC
AXI_S_ARLEN_0-3	6	I	ARLEN from NOC
AXI_S_ARUSER_0-3	USER_REQ_WIDTH	I	ARUSER from NOC
AXI_S_ARCACHE_0-3	4	I	ARCACHE from NOC
AXI_S_ARADDR_0-3	ADDR_WIDTH	I	ARADDR from NOC 10bits: Destination GPU ID 3bits: Destination PORT ID
AXI_S_ARLOCK_0-3	1	I	ARLOCK from NOC

AR通道没有payload，封装好的报文全长60Bytes。

7.1.1.7 AXI R 通道

通信芯粒侧发起R通道应答主接口侧信号定义见表9，通道应答从接口侧信号定义见表10，用于和GPU侧R通道互通使用。

表 9　R 主接口

接口信号	接口位宽	接口方向	功能描述
AXI_M_RVALID_0-3	1	I	RVALID from NOC
AXI_M_RREADY_0-3	1	O	RREADY to NOC
AXI_M_RLAST_0-3	1	I	RLAST from NOC
AXI_M_RDATA_0-3	DATA_WIDTH	I	RDATA from NOC
AXI_M_RUSER_0-3	USER_REQ_WIDTH	I	RUSER from NOC Keep same value in one transaction
AXI_M_RRESP_0-3	2	I	RRESP from NOC Keep same value in one transaction
AXI_M_RID_0-3	ID_WIDTH	I	RID from NOC Keep same value in one transaction 10bits: Destination GPU ID 3bits: Destination PORT ID

表 10　R 从接口

接口信号	接口位宽	接口方向	功能描述
AXI_S_RVALID_0-3	1	O	RVALID to NOC
AXI_S_RREADY_0-3	1	I	RREADY from NOC
AXI_S_RLAST_0-3	1	O	RLAST to NOC
AXI_S_RDATA_0-3	DATA_WIDTH	O	RDATA to NOC
AXI_S_RUSER_0-3	USER_REQ_WIDTH	O	RUSER to NOC Keep same value in one transaction
AXI_S_RRESP_0-3	2	O	RRESP from NOC Keep same value in one transaction
AXI_S_RID_0-3	ID_WIDTH	O	RID to NOC Keep same value in one transaction 10bits: Destination GPU ID 3bits: Destination PORT ID

R通道封装好的报文如图16所示：

IGPH/ OGPH 4B	AXI Header 8B	RDATA First 64B	RDATA Last 64B

图 16　R 报文示意

7.1.1.8　流控接口

表11定义了加速器和通信芯粒之间传递PFC流控信号的接口，用于通信芯粒传递PFC信号使用。

表 11　流控反压接口

接口信号	接口位宽	接口方向	功能描述
gpu2iodie_eth_pfc_0-3	8	I	GPU 传递给 IO 的以太网接口 PFC
iodie2gpu_eth_pfc_0-3	8	O	IO 传递给 GPU 的以太网接口 PFC

7.1.2　AXI-Stream 格式

AXI-Stream模式互通，如图17所示，GPU按照AXI-Stream格式定义完成转换，保留GPU当前私有互联协议。

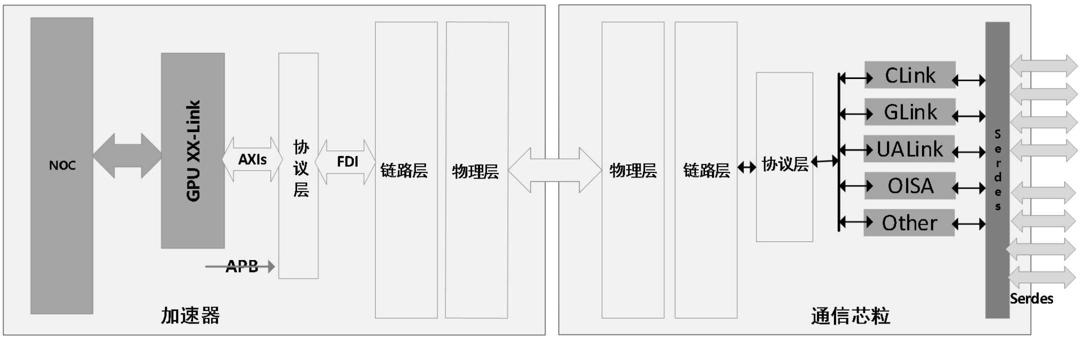


图 17　AXI-Stream 模式

7.1.2.1　AXI-Stream 模式

GPU侧仍然使用当前私有互联协议报文格式，协议层提供AXI-Stream模式。源GPU采用AXI-Stream定义格式，提供标准网络转发信息即可，协议层会把打包后的报文透传给通信芯粒进行网络转发。当对端AXI-Stream接口收到报文后，按照定义的格式透传给目的GPU，目的GPU使用私有互联协议完成最终接收处理。采用AXI-Stream格式，能够支持GPU直接使用私有互联协议。

UMAC功能图示如图18（仅展示前两组Stream port对应的部分，后两组Stream port对应部分完全相同）：

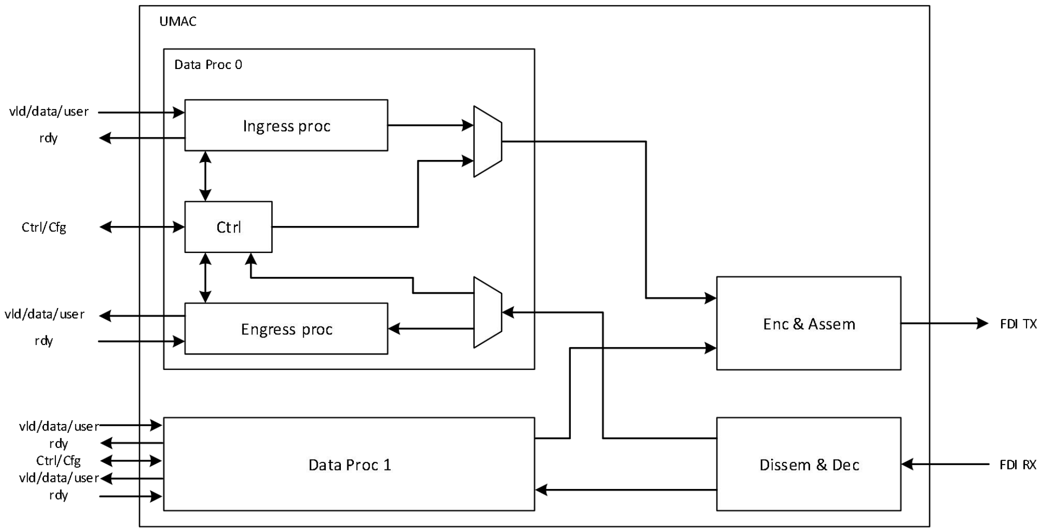


图 18　UMAC 功能图示

UMAC功能：

a)　UMAC一端连接AXI Stream接口，AXI Stream接口均为双向通信；另一端连接链路层FDI接口，FDI发送/接收的基于物理层；
b)　发送方向，从AXI Stream接口获取数据，将其切分打包成为Flit格式传输到FDI接口上；
c)　接收方向，从FDI接口中获取数据，将其解包恢复成原数据从AXI Stream端口发出；
d)　发送方向设置缓存，用于吸收突发输入和响应对端反压，当上行缓存近满时，反压前级；
e)　接收方向设置缓存，用于吸收突发和响应后级反压。当缓存积累时，向对端发起反压；

f） 发送方向缓存出口处可配置流量整形，用于减小对端缓存的压力；
g） 提供对端寄存器访问通路，软件可以通过操作本端寄存器间接读写对端寄存器；
h） FDI前的发送到接收环回功能；
i） 跨die透传分通道反压信号。

7.1.2.2 AXI-Stream 数据交互

交互内容应遵循下面要求：

发送方向：输入数据->切分成为内部使用格式->填入Flit->由FDI发送。

接收方向：FDI输入数据->跨时钟域->从Flit中解出内部使用格式->拼接回原有包格式输出。

AXI Stream模式应提供报文路由信息，包括目的GPU ID、目的PORT ID、报文类型（请求/响应），用于路由转发与负载均衡。

7.1.2.3 AXI-Stream 接口

每组AXI Stream接口每拍只能传输一个包的数据，要求EOP之外的所有拍数据均为满64B，EOP当拍通过SIZE字段给出数据的有效Byte数，表12提供AXI-Stream接口信号详细定义。

表 12 AXI Stream 接口

接口信号	接口位宽	接口方向	功能描述
utx_tvalid_0-3	1	I	AXI Stream 0 tvalid for TX
utx_tdata_0-3	DATA_WIDTH	I	AXI Stream 0 tdata for TX
utx_tuser_0-3	USER_ WIDTH	I	AXI Stream 0 tuser for TX 1bit: SOP, start of packet 1bit: EOP, end of packet 1bit: ERR, has error in data. 6bit: SIZE, (valid byte count-1) in tdata, used only EOP=1 10bits: GPUID, endpoint GPU ID for this packet to be transmitted to, used only SOP=1 1bit: TYPE, this packet is resp(0)/req(1), used only SOP=1
utx_tready_0-3	1	O	AXI Stream 0 tready for TX
urx_tvalid_0-3	1	O	AXI Stream 0 tvalid for RX
urx_tdata_0-3	DATA_WIDTH	O	AXI Stream 0 tdata for RX
urx_tuser_0-3	USER_ WIDTH	O	AXI Stream 0 tuser for RX

			1bit: SOP, start of packet 1bit: EOP, end of packet 1bit: ERR, has error in data. 6bit: SIZE, (valid byte count-1) in tdata, regardless of resp/req, used only EOP=1 10bits: GPUID, startpoint GPU ID for this packet was transmitted from, used only SOP=1
urx_tready_0-3	1	I	AXI Stream 0 tready for RX.

7.1.2.4　分通道流控接口

加速器和通信芯粒之间应支持分通道流控，对应请求通道和响应通道分别流控，每个通道支持4个ready信号用于支持最大4路AXI-Stream接口的流控传递，具体接口信号定义见表13：

表 13　分通道流控接口

接口信号	接口位宽	接口方向	7.1.2.5　功能描述
iodie2gpu_req_rdy_0-3	1	O	通信芯粒到 GPU 的 Request 反压，低有效，当反压拉起时，对应通道的数据需要在 AXI Stream 端口上立即停发
iodie2gpu_resp_rdy_0-3	1	O	通信芯粒到 GPU 的 Response 反压，低有效，当反压拉起时，对应通道的数据需要在 AXI Stream 端口上立即停发
gpu2iodie_req_rdy_0-3	1	I	GPU到通信芯粒的Request反压，低有效； 当反压拉起时，需要路径延时，对应通道的数据才会在AXI Stream端口上停发
gpu2iodie_resp_rdy_0-3	1	I	GPU到通信芯粒的Response反压，低有效； 当反压拉起时，需要路径延时，对应通道的数据才会在AXI Stream端口上停发

7.1.3　事务层公共接口

7.1.3.1　时钟/复位接口

表14定义了时钟/复位接口信号：

表 14　时钟复位接口

接口信号	接口位宽	接口方向	功能描述

clk	1	I	主时钟
fdi_lclk	1	I	FDI 工作时钟，1GHz 或 1.5GHz
rst_n	1	I	异步复位，需要在clk时钟域同步释放

所有的AXI、AXI Stream接口均工作在clk时钟域。

7.1.3.2 FDI 接口

FDI 0/1接口分别工作在fdi_lclk_0/1时钟域，1GHz或1.5GHz。表15定义了FDI接口信号：

表 15 FDI 接口

接口信号	接口位宽	接口方向	功能描述
umac_lp_valid_0/1	1	O	FDI 发送数据有效
umac_lp_data_0/1	1024	O	FDI 发送数据
umac_lp_irdy_0/1	1	O	FDI 有数据待发送状态，与 lp_valid 值完全一致
umac_pl_trdy_0/1	1	I	FDI 发送 ready
umac_pl_valid_0/1	1	I	FDI 接收数据有效
umac_pl_data_0/1	1024	I	FDI 接收数据
umac_pl_flit_cancel_0/1	1	I	取消上一拍数据

7.1.3.3 配置接口（APB）

该APB接口工作在clk时钟域。读写对端寄存器需要操作本端的寄存器序列间接完成。其中pready信号会在读写操作时均会拉低，待其恢复为高时方可进行下一次操作。表16定义了配置接口信号：

表 16 配置接口

接口信号	接口位宽	接口方向	功能描述
cfg_penable	1	I	APB penable
cfg_psel	1	I	APB psel
cfg_pwrite	1	I	APB pwrite
cfg_paddr	16	I	APB paddr
cfg_pwdata	32	I	APB pwdata
cfg_prdata	32	O	APB prdata
cfg_pready	1	O	APB pready

7.1.3.4 中断接口

本模块发生异常时会根据类型，拉起致命中断或者非致命中断线并保持高电平。表17定义了中断接口信号：

表 17 中断接口

接口信号	接口位宽	接口方向	功能描述
cr_intr	1	O	致命中断，这类中断需要停流复位处理。
ncr_intr	1	O	非致命中断，仅作为状态提示。

7.1.4 数据传输

应采用跨die可识别的网络帧格式，用于网络的数据路由转发。

7.1.4.1 通信芯粒协议层报文头

通信芯粒在初始化时，需要GPU主芯片配置通信芯粒本端GPU ID信息，用于通信芯粒网络处理使用，比如发送时本端GPU ID用于生成源GPU ID信息使用，目的端通信芯粒接收到网络传输的源GPU ID，用来生成OGPH头中源GPU ID域段信息。

发往通信芯粒的每个包均携带IGPH报文头，从通信芯粒发来的每个包均携带OGPH报文头。每个包长度不得小于60B。

通信协议层报文头IGPH/OGPH需要按照网络序（大端序）字节序格式，保证在通信芯粒和交换网的数据传输一致性。

表18所示IGPH报文头需要携带信息：单播/组播指示，流分类(区分请求/响应)，目的GPU ID和目的PORT ID信息提供通信芯粒进行网络路由转发使用。

表19所示OGPH头需要携带信息：单播/组播指示，流分类(区分请求/响应)，源GPU ID和源PORT ID信息。

表 18 IGPH 格式

字段		位宽	位号	说明
TYPE		2	31:30	00：单播报文 01：组播报文
TYPE = 00	RSV	13	29:19	保留位。
	Traffic Class	3	18:16	报文优先级，1 为响应，0 为请求
	RSV	2	15:14	保留位
	DST_GPU_ID	11	13:3	目的 GPU ID
	DST_PORT_ID	3	2:0	目的端口

TYPE = 01	RSV	11	29:19	保留位。
	Traffic Class	3	18:16	报文优先级
	RSV	1	15:15	保留位
	MULTICAST_ID	15	14:0	组播组 ID

表 19　OGPH 格式

字段		位宽	位号	说明
TYPE		2	31:30	00：单播报文 01：组播报文
TYPE = 00	RSV	13	29:19	保留位。
	Traffic Class	3	18:16	报文优先级，1 为响应，0 为请求
	RSV	2	15:14	保留位
	SRC_GPU_ID	11	13:3	源 GPU ID
	SRC_PORT_ID	3	2:0	源端口
TYPE = 01	RSV	16	29:14	保留位。
	SRC_GPU_ID	11	13:3	源 GPU ID
	SRC_PORT_ID	3	2:0	源端口

7.1.4.2　Flit 定义

Flit格式宜采用Format 6: Latency-Optimized 256B with Optional Bytes Flit Format for Streaming Protocol。在payload空间中，放置端口0和1各一半数据、以及对应的报文头。报文头中承载了数据属性、跨die透传的控制信号等信息。

图19所示中，在250B的payload空间中，放置端口0和1各2个60B数据、1个2B Inf头和1个3B Inf头。

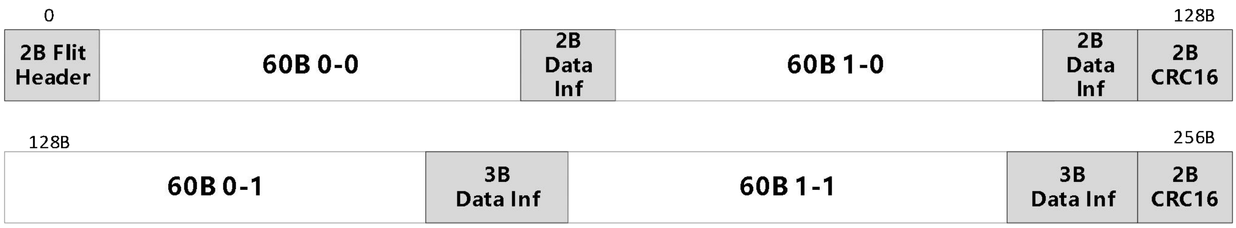


图 19　Flit 格式示意

7.1.4.3　控制帧

控制帧总长度60B（480 bit），用于跨die传输配置等控制信息，全部内容均由ECC保护。

7.1.4.4 NOP Flit 产生

在没有数据流的情况下，需要每隔一个可配置的时间检查跨die透传信号的变化，如果发生变化则产生全部为IDLE粒度的Flit，从而将信号传递过去。

7.1.5 跨 die 初始化

双侧使用的物理层和链路层，以及外围设计可上电后自行建链进入工作状态，无需GPU侧控制模块主动干预。需要有低速的配置接口，实现GPU对通信芯粒的完整访问，当无法自行建链或者建链失败时，通过低速的配置接口，GPU侧的控制模块可干预建链备份配置通路。

在以自动化或者主动配置方式建链完成并进入工作状态后，可以通过下述方式对通信芯粒进行跨die配置，从而完成整个通信芯粒的初始化。

7.1.5.1 跨 die 控制交互

GPU侧控制模块需要提供一组寄存器，用于批量跨die访问对端寄存器。

GPU侧控制处理首先配置好地址信息，写操作还需配置好写数据，随后写指令寄存器指定寄存器数量和操作类型，触发控制访问模块将操作信息传递到对端，由对端按顺序依次执行。全部执行完毕后，对端将反馈信息传回本端。GPU侧控制处理在这个过程中需要连续查询状态寄存器，直到指示完成。

对于读操作，操作执行成功时，可以从数据区中读回数据。

7.1.5.2 端口级反压传递

控制信息宜提供TRDY字段用于端口级反压。在收到TRDY信号为低时，本端UMAC应该立即停止填写新的Flit；在收到TRDY信号重新为高时，继续填写新的Flit并发送。

本端UMAC可以自行在接收侧缓存中数据积累时产生反压并将本端发送的TRDY信号拉低，在解除反压时将本端发送TRDY信号恢复为高。该信号也可由GPU侧提供，此时GPU侧应保证不拉低任何端口上的xREADY信号。

7.1.5.3 中断传递

控制信息宜提供INTR字段用于跨die的电平型中断传递，如不使用该功能，需要保持该bit为0。如果使用该功能，两个die之间需要有额外的备份通道传输对端UMAC与链路层生的致命中断及伴随信息。例如，使用一对GPIO在两个芯粒之间传递上述致命中断，使用I2C等低速总线提供基本的寄存器读写功能以获取中断信息。

7.1.5.4 分通道反压传递

报文头信息中宜提供{BPFC, FPFC}作以下用途：

a) 以太网接口PFC信息传递
b) AXI模式下的分通道反压，其中分别提供{响应反压（作用到B、R通道），请求反压（作用到AW&W、AR通道）}，其余Bit保留作未来规划用途，tie 0。

c) AXI Stream模式下的分通道反压，其中分别提供{响应反压，请求反压}，由GPU侧响应，其余Bit保留作未来规划用途，tie 0。

d) 其他需要分通道反压的用途。

7.1.6 通信芯粒事务层接口

通信芯粒事务层接口应该具有以下功能：

a) 将接收的Flit数据以与主芯片相同的方式解析为原有的数据分片并恢复成为整包。

b) 将交换侧输入的数据与主芯片侧相同的方式填写到Flit上，然后发送。

c) 响应端口级反压。

d) 将通信芯粒侧的分通道反压填写到Flit上。

e) 接受寄存器访问请求帧，发起寄存器访问，并返回结果给主芯片。

f) 在可用的情况下，通过Flit中的INTR字段传递中断。

g) 在可用的情况下，通过Flit中的反压字段传递分通道反压信号。

7.2 链路层要求

通过一种通用的Flit格式，完成不同供应商物理层上承载的Flit格式，不同厂家物理层通过链路层完成互联。

实现链路层功能，包括链路协商、管理等交互。

两侧相同FDI工作频率，边带访问。两侧均应支持本文件的要求，用于互通。

要求两侧FDI工作在512bit@1GHz或512bit@1.5GHz，具体频率可根据PHY的data rate，参考IP供应商的实际实现做调整。

7.2.1 Flit 格式

应采用Latency-Optimized 256B with Optional Bytes Flit Format for Streaming Protocol格式，以得到最佳传输延迟，具体格式如表20和表21所示。

此模式下，两组CRC需要被计算。2字节宽度的CRC校验码和前述一样都需要从128字节的数据中生成。CRC0在Flit的前126字节中生成(包括Flit头、Chunk 0、Chunk 1以及可选字段，其中预留字段在CRC计算中会填充0)，计算出的CRC校验码，将放在的C0B0，C0B1位置(见下图20)。CRC1在Flit的Chunk 2与Chunk 3中生成(可选字段也会加入到CRC计算中)，计算出的CRC校验码，将放在C1B0，C1B1位置。若不需要重传，则链路层需要计算并且填充CRC字段。强烈建议接收方把CRC错误视为无法更正错误。

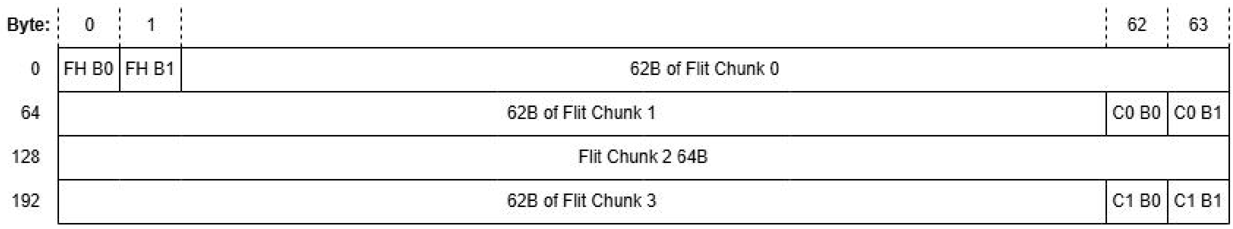


图 20 Flit 格式

表 20 Flit 格式关闭重传

Byte	Bit	Description (Streaming Protocol)
0	[7:6]	Protocol Identifier: 00b:D2D 链路层 NOP Flit Remaining encodings are permitted to be used by Protocol Layer in a vendor defined manner. Protocol Layer must never set this to 00b for Flits sent across FDI.
	[5]	Stack Identifier 0:Stack 0 1:Stack 1
	[4]	Reserved
	[3:0]	Reserved
1	[7:6]	Flit Type: 00b:CXL/PCIe/Streaming Flit/D2D 链路层 NOP Flit 01b:Test Flit 10b:Management Flit 11b:Reserved
	[5:0]	Reserved

表 21　Flit 格式带重传

Byte	Bit	Description (Streaming Protocol)
0	[7:6]	Protocol Identifier: 00b:D2D 链路层 NOP Flit Remaining encodings are permitted to be used by Protocol Layer in a vendor defined manner. Protocol Layer must never set this to 00b for Flits sent across FDI.
	[5]	Stack Identifier 0:Stack 0 1:Stack 1
	[4]	Reserved
	[3:0]	The upper four bits of Sequence number “S”(i.e., S[7:4])
1	[7:6]	Flit Type: 00b:CXL/PCIe/Streaming Flit/D2D 链路层 NOP Flit 01b:Test Flit 10b:Management Flit 11b:Reserved
	[5:4]	Ack or Nak Information: 00b:Explicit Sequence number ”S” of the current Flit is present. 01b:Ack.The sequence number “S” carries the Ack’ed sequence number. 10b:Nak.The sequence number “S” carries 255 if N=1; otherwise, it carries N-1; when N is the Nak’ed sequence number. 11b:reserved
	[3:0]	The lower four bits of Sequence number “S” (i.e., S[3:0]). Sequence number 0 is reserved and if present, it implies no Ack or Nak is sent.

7.2.2 CRC 计算

协议采用CRC计算与重传来确保可靠数据传输，CRC校验的结果放在Flit头信息中。CRC的生成多项式为 $(x+1)*(x^{15}+x+1)=x^{16}+x^{15}+x^{2}+1$，此公式为随机位错误提供了 3 bit 检测的保证：2 bit 检测保护是因为原始多项式 $(x^{15}+x+1)$，另外的 1 bit 检测保护是因为多项式中的 $(x+1)$，通过奇校验来提供1个额外的错误码检测保证。

CRC总是在128字节的数据中计算产生。对于更小的数据段，需要在高位上填充0以达到128字节。

CRC的LFSR计算的初始值为0000h。CRC计算从字节0的第0个bit开始计算，并且每个字节从bit0开始计算到bit7，如图21所示。在图中，C[15]是字节1的bit7，C[14]是字节1的bit6，如此类推。C[7]是字节0的bit7，C[6]是字节0的bit6，如此类推。

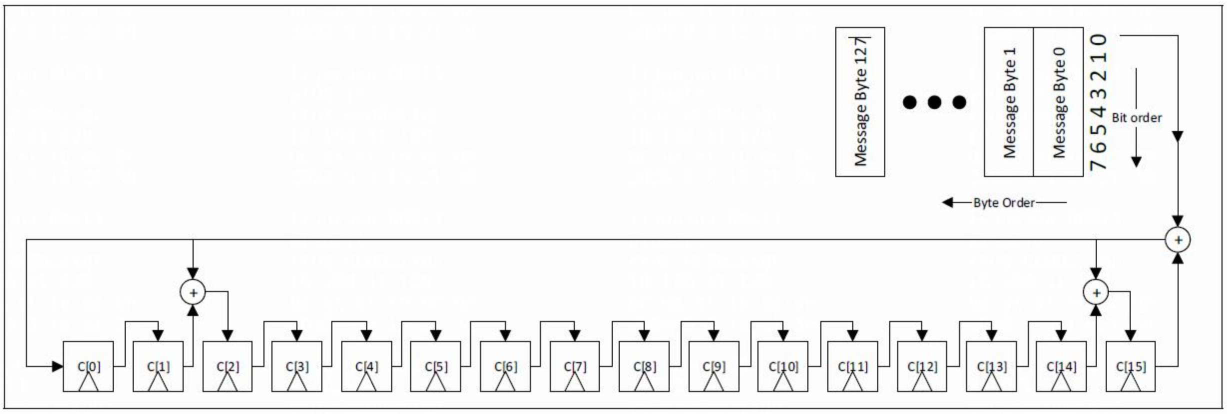


图 21　CRC 计算

CRC校验码生成的verilog代码宜兼容UCIe 2.0协议规范的格式，并且必须在编写代码时使用。此代码是应用于发送端。它使用1024位数据作为输入，输出16位数据作为验证码。在接收端，使用接收到的Flit字节计算CRC校验码，并且在高位上填充适当数量的0以满足128字节。如果收到的CRC与计算得到的CRC不匹配，则此Flit将会被声明为无效且发起重传。

7.2.3 重传机制

7.2.3.1 能力协商

当链路的速率小于8GT/s时，重传机制可选，对于速率大于8GT/s的链路，链路层必须要支持重传功能，除非处于Raw模式下。如果链路层不支持重传，在链路训练的过程中物理层不能对外传播支持8GT/s以上的速率，除非当前协商模式为Raw Mode。

当链路在启动时，在链路重传协商成功后，重传是无法被关闭的，降低链路速率也无法关闭。重传只能在下一次链路启动时进行重协商。

7.2.3.2 Ack/Nak 机制

Ack/Nak机制与PCIe 6.0 Flit Mode 的重传机制类似，采用8bit的Flit序列号,位于Flit头中。另外注意开启重传和不开启重传时，Flit头相关位域的定义是不同的，参考7.2.1章节报文头格式。

可采用Ack/Nak机制：

Ack/Nak机制内部发送端由NTS计数器（Next Transmit Seq Number），CRC生成器，重传缓存,Retry_timer计数,Retry_num计数,Ackd_seq寄存器,Flit CRC校验等元素构成，其中NTS计数器用于产生下一个待发送Flit的序列号。

CRC生成器用于生成一个CRC值，作用于整个Flit和对应的序列号，重传缓存用来备份发送方的每一个Flit（包括序列号和CRC），直到其收到来自接收方的Ack，确认该Flit已经成功的被接受，才会删除这个备份。如果接收方发现Flit存在错误，则会向发送方发送Nak，然后发送方会从重传缓存中取出数据，重新发送该Flit。

Retry_timer计数是一个看门狗计时器，当该计时器溢出，则表明发送端已经发送了一个或多个Flit,但是并未收到来自接收端的应答信号，此时，发送端会将重传缓存中的Flit重新发送，并将看门狗计时器重启。只要发送任何的Flit，该计时器便会启动，当收到应答信号前都会持续地进行，当收到应答信号时，定时器会被立马清零。如果此时重传缓存依然有Flit,定时器会被立即重新启动，如果没有Flit,便不会重新启动。

Retry_num计数是一个计数器用来记录Flit发送失败的次数，当达到阈值时，强制PHY进行Link 重新训练。当接收到接收端的NAK和Retry_timer溢出时，该计数器会被加一，当接收到ACK时，计数器清零。

Ackd_seq寄存器用来存储最近接收到的ACK或者NAK的序列号。当复位或数据链路层关闭时，该寄存器会被初始化为1。

Flit CRC校验是接收端在接收到来自发送端的Flit后，检查Flit CRC,如果有错误，则会将其丢弃，判断为无效的Flit.

接收端由CRC错误校验，NRS计数器（Next Receive Seq Num）,NAK_Scheduled指示，ACK/NAK_Latency_Timer和ACK/NAK生成器构成。

CRC错误校验用来检查接收到的Flit是否存在错误，如果错误，丢弃Flit，并产生一个NAK发送给发送端，让其重新发送该Flit。

NRS计数器的值为已经接收到的Flit的序列号加一。用于检查当前的Flit是不是该收到的Flit。另外，如果NRS计数器和当前的Flit序列号相等，则认为这是一个有效的Flit，但接收端并不会立即发送ACK，需要等到ACK/NAK Latency_timer溢出才发送ACK flit以增加效率。如果当前接收到的Flit序列号小于NRS计数器的值，则认为该Flit已经发送过。这并不是一个错误，直接将此Flit遗弃，但会返回一个上次成功接收到的Flit序列号的Ack给发送端。如果当前接收到的Flit序列号大于NRS计数器的值，表明丢包，会返回NAK，丢弃该Flit。

NAK_Scheduled指示是接收端接收到NAK时的标志位。成功接收到时会被清零。当处于置位状态时，接收端不应产生其他的NAK。

ACK/NAK_Latency_Timer会在接收端成功接收到有效的Flit且并未向发送端返回ACK之前运行。当定时器溢出时，接收端会立即发送ACK给发送端，无论返回ACK或NAK，定时器都会被复位，但只有接收到有效的Flit时，定时器才会被重新启动。

ACK/NAK生成器产生ACK/NAK Flit。

7.2.4 链路状态管理

链路状态进行管理应采用层次化的方式，以实现不同层之间功能划分的明确定义，如图22所示为不同配置的状态机层次结构示例。FDI接口和RDI接口信号以及与对端的边带消息交互遵循统一互联接口协议。

a) 对于PCIe或Streaming协议，链路层LSM暴露在FDI pl_state_sts上。链路层LSM使用{LinkMgmt.链路层*}边带消息与对端协调链路状态，这些消息由链路层发起和接收。

b) RDI SM是提供给上层的抽象的物理层接口状态，RDI SM状态转换使用{LinkMgmt.RDI*}边带消息与远端协调。这些消息由PHY发起和接收。

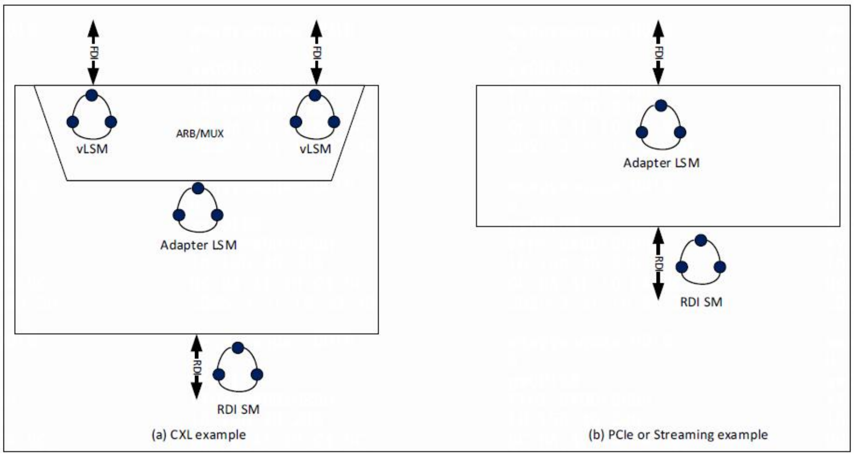


图 22 不同配置的状态机层次结构

对于FDI接口侧和RDI接口侧的链路状态管理接口定义如表22和表23：

表 22 链路状态管理接口(FDI 接口侧)

接口信号	接口位宽	接口方向	功能描述
fdi_lp_state_req	4	I	协议层请求链路层状态更改
fdi_lp_linkerror	1	I	协议层到链路层指示发生了链路错误，请求链路层进入 linkerr 状态。
fdi_pl_state_sts	4	O	链路层到协议层接口的状态指示
fdi_pl_inband_pres	1	O	表明 D2D 链路已经完成与对端的参数协商
fdi_pl_rx_active_req	1	O	链路测请求协议层开启其接收器数据通道并准备接受

			数据。
fdi_lp_rx_active_sts	1	I	指示协议层已准备好接收和解析协议数据或 Flits
fdi_pl_protocol	3	O	指示采用的协议。
fdi_pl_protocol_flitfmt	4	O	指示采用的 Flit 格式。
fdi_pl_protocol_vld	1	O	指示已协商好合适的协议和 Flit 格式
fdi_pl_stallreq	1	O	链路层请求协议层中断在 Flits 边界的传输
fdi_lp_stallack	1	I	指示已在 Flits 边界暂停其流水线
fdi_pl_clk_req	1	O	请求从协议层内部移除其时钟门控
fdi_lp_clk_ack	1	I	fdi_pl_clk_req 的响应信号
fdi_lp_wake_req	1	I	协议层作为请求的发起方，请求链路层移除时钟门控。
fdi_pl_wake_ack	1	O	fdi_lp_wake_req 的响应信号
fdi_pl_phyinrecenter	1	O	指示链路正在进行训练或再训练
fdi_pl_phyinl1	1	O	指示物理层处于 L1 电源管理状态
fdi_pl_phyinl2	1	O	指示物理层处于 L2 电源管理状态

表 23　链路状态管理接口(RDI 接口侧)

接口信号	接口位宽	接口方向	功能描述
rdi_lp_state_req	4	O	链路层请求物理层状态更改
rdi_lp_linkerror	1	O	指示链路层到物理层发生了链路错误，要求链路断开。
rdi_pl_state_sts	4	I	物理层到链路层的状态指示。
rdi_pl_inband_pres	1	I	表明 D2D 链路已经完成与对端链路伙伴的参数协商，并准备将 RDI 过渡到 Active 和第三阶段的启动。
rdi_pl_clk_req	1	I	物理层请求从链路层的内部逻辑中移除时钟门控。
rdi_lp_clk_ack	1	O	rdi_pl_clk_req 的响应信号
rdi_lp_wake_req	1	O	链路层作为请求的发起方，请求物理层移除时钟门控。
rdi_pl_wake_ack	1	I	rdi_lp_wake_req 的响应信号
rdi_pl_phyinrecenter	1	I	指示物理层正在训练或再训练。
rdi_pl_stallreq	1	I	物理层请求在 Flit 边界对齐发送器，并且不发送任何新的 Flits 来准备状态切换。
rdi_lp_stallack	1	O	指示已在 Flits 边界暂停其流水线。

在主带建链前，须对链路进行初始化。宜将链路初始化分为4个阶段，如图23所示，链路初始化最后一个阶段（Stage3）便是链路层初始化。

Stage 0 复位，每个Die独立发生的；不同的芯片可能需要不同的时间来完成阶段0。

Stage 1 涉及边带初始化。

Stage 2 涉及主带训练和修复。

Stage 3 涉及适配器之间的参数交换以协商协议和 Flit 格式。

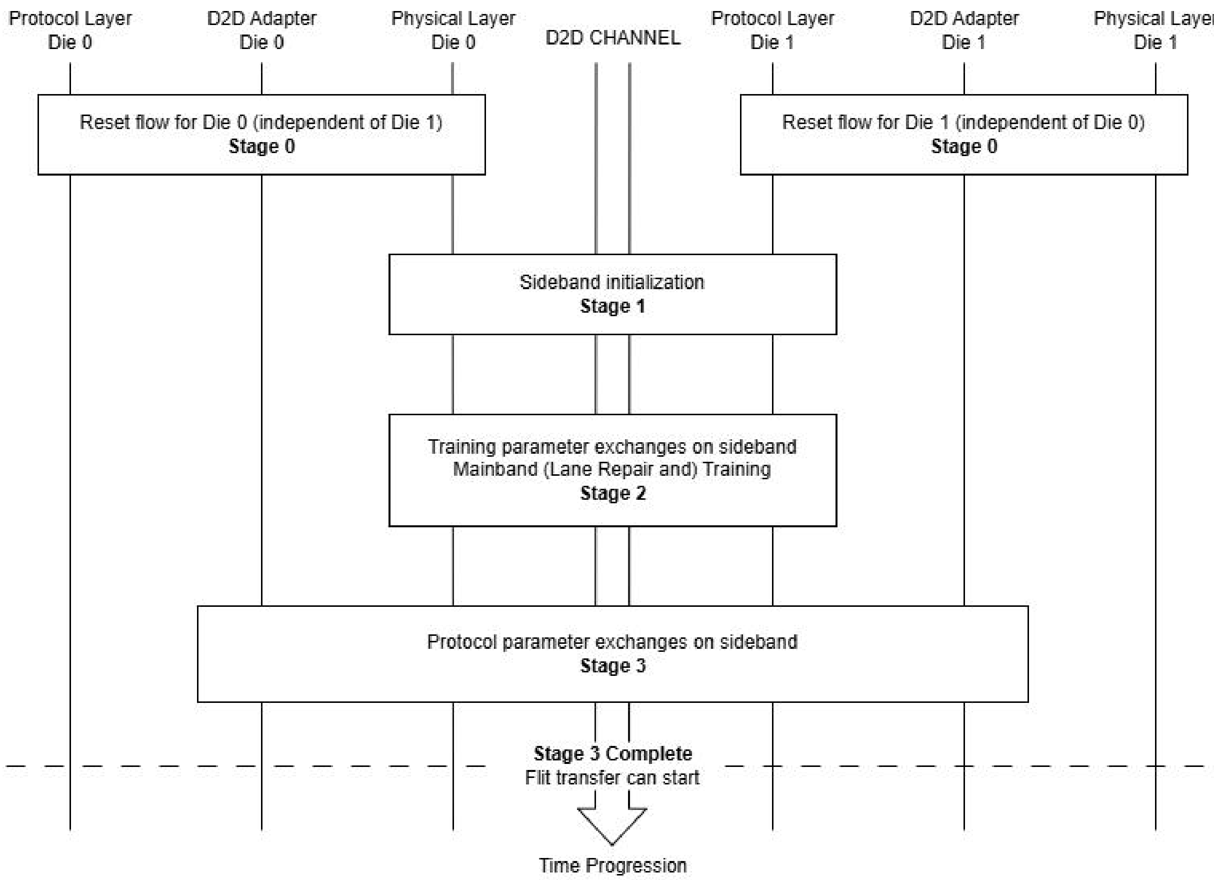


图 23　链路初始化流程

在PHY的初始化和训练过程中，边带和主带宜分开进行初始化和训练，如图24所示，以下PHY初始化描述：

边带初始化(SBINIT),使边带进入正常工作状态,便于后续初始化及训练过程中在链路上传递边带信息;

主带初始化(MBINIT)，两侧模块进行参数交换及协商、链路修复等工作，使主带能够工作在最低速（4GT/s）。

对主带进行训练(MBTRAIN)、电气参数调整、切速等工作，使主带工作在最高传输速率。

主带训练完毕后进行RDI Bring Up，最后PHY LSM进入激活状态，完成链路初始化及训练。

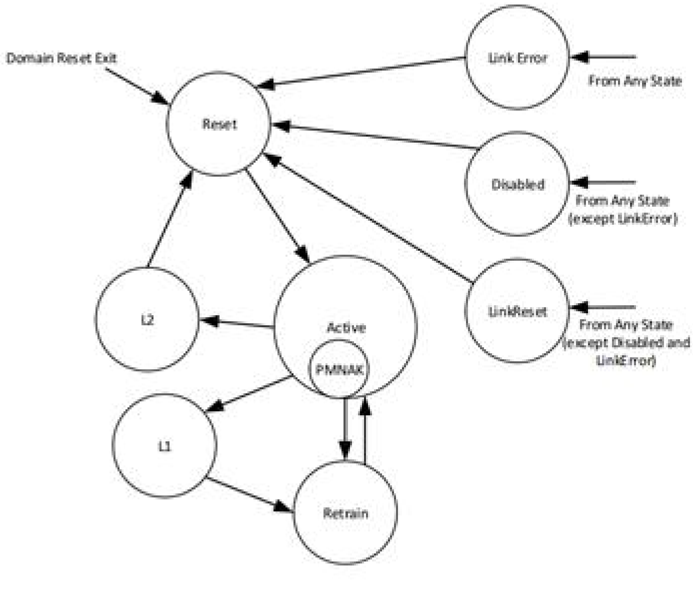


图 24　RDI 接口状态机

当RDI状态机进入活动状态时，链路初始化的阶段2完成。图25为RDI从复位状态转换为激活状态的初始化流程。（当RDI处于激活状态时，PHY将会从Link control寄存器中清除其“Start Link training”bit位）

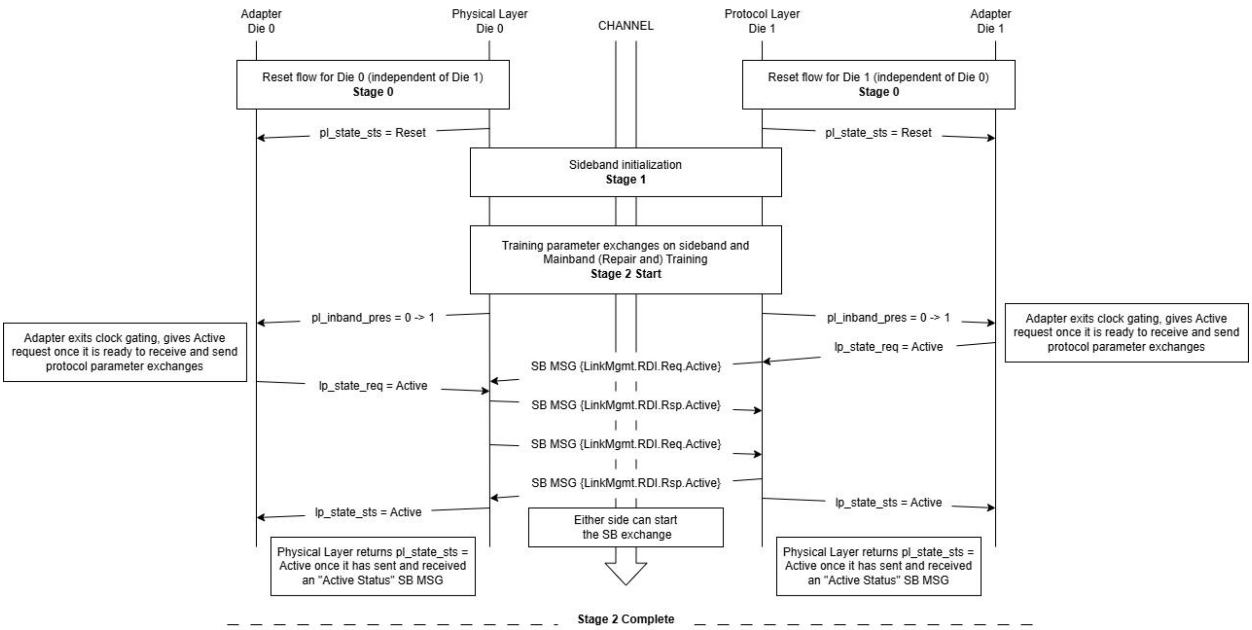


图 25　RDI Bring Up 流程

完成RDI启动进入阶段3后，链路层必须遵循一系列步骤以确定本地能力、完成参数交换并将FDI状态机置于激活状态，如图26所示。

a) 查询本地能力：链路层必须确定物理层训练的结果以及给定的链路速度和配置是否需要重传。如果链路层支持重传，它必须在参数交换期间将此功能告知给对端。

b) 与对端进行参数交换。

c) FDI启动：两侧经过一系列握手协商后使FDI状态机进入激活状态。

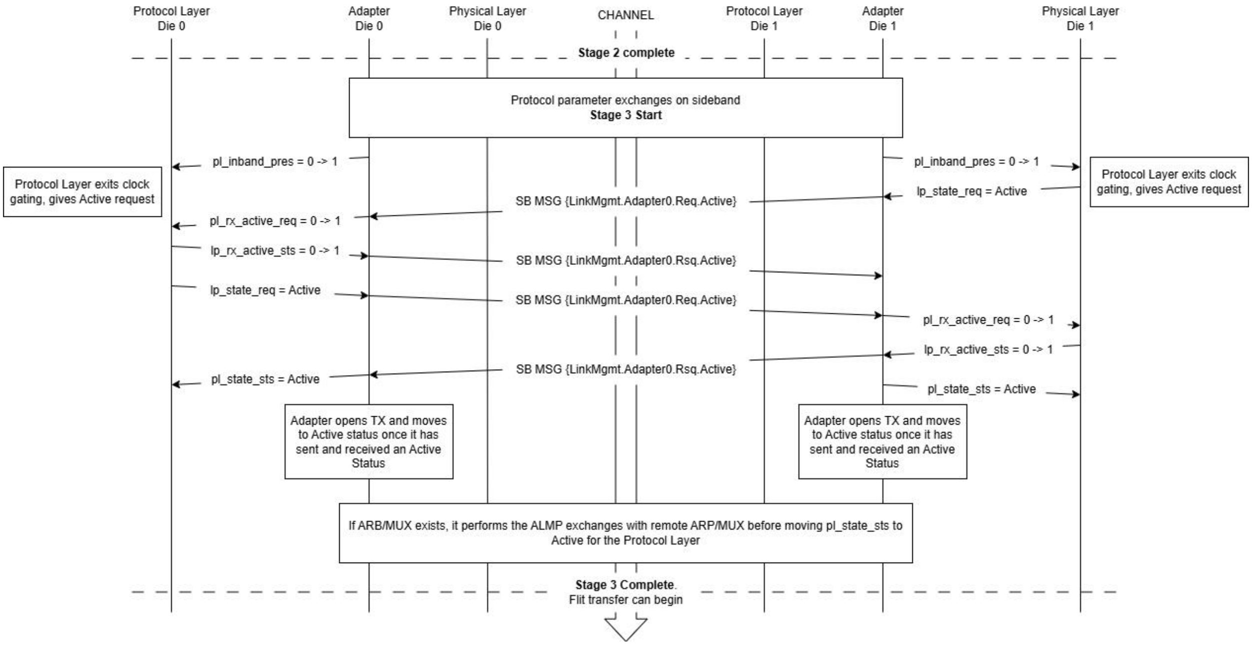


图 26　FDI Bring Up 流程

链路启动的阶段3，此部分突出展示了FDI上的变化情况，FDI进入Active，标志着阶段3以及链路初始化完成，主带可以传输协议层发来的Flit。此阶段需要在FDI上执行如下的链路状态从复位状态转移到激活状态过程相关的流程。

a) 当链路层在 RDI 上完成转换到激活状态并且与对端成功实现参数协商后，其需要与协议层执行 pl_clk_req 握手并且在 FDI 上反映 pl_inband_pres=1。

b) 此为触发协议层请求转换到激活状态。允许协议层在请求转换到激活状态前等待 pl_protocol_vld=1。其需要执行 lp_wake_req 握手过程，需要注意的是，lp_wake_req 握手流程不在图中展示。

c) 当采样到 lp_state_req=Active 后，链路层需要在边带上向对端发送{LinkMgmt.链路层 0.Req.Active}。

d) 链路层需要在保证协议层接收器准备好时，以边带信号{LinkMgmt.链路层.Rsp.Active}在边带上响应边带信号{LinkMgmt.链路层.Req.Active}。边带信号{{LinkMgmt.链路层 0.Rsp.Active}需要在链路层采样到 pl_rx_active_req=lp_rx_active_sts=1 时。如前面所述，pl_clk_req 握手也适用于 pl_rx_active_req；允许链路层在 Bring Up 流程中保持 pl_clk_req 拉高(当其被 pl_inband_pres 拉高之后)。

e) 如果没有使用 ARB/MUX，当链路层收到并发送了边带信号{LinkMgmt.链路层 0.Rsp.Active}之后，其需要把 pl_state_sts 转换为协议层的激活，并且可以开始传输 Flit。

f) 如果使用了 ARB/MUX，边带消息{LinkMgmt.链路层 0.Rsp.Active}的发送与接收将会使得 ARB/MUX 在主带上执行 ALMP 交换，并且最终把 vLSM 状态转移到激活。
g) 上述步骤 3 到步骤 6 构成了 FDI 上的"Active Entry Handshake"(进入 Active 状态握手过程)，并且需要为每一次进入激活状态都执行一次。

链路状态管理还包括链路的Power以及状态等的管理(Power Management, PM)。协议层采用 PCIe、CXL协议时必须支持L1、L2低功耗状态。在满足初始空闲时间需求后，协议层在FDI上请求进入PM状态。初始空闲时间宜依据具体协议及实现而定。链路层通过与协议层及物理层进行握手，使链路进入低功耗状态。L1状态下，对模拟电路进行关闭处理，L2状态下，增加关闭PLL。FDI、RDI都支持 L1、L2 电源状态，但在L1、L2状态下，RDI内部可以将这两个状态映射为常规非低功耗状态。此外，链路层还需监控链路是否正常，链路层会根据触发事件，协调链路状态，比如进入Linkerror/linkreset/Disabled等状态或者重新初始化，保证整个互联系统正常工作。

表24描述了下层在各状态需考虑/忽略的上层请求，上层也需要考虑接口状态并作出必要的请求。

表 24　较低层在每个状态时所需考虑的请求

Request Versus Status	Reset	Active	L1	LinkReset	Retrain	Disable	L2	LinkError
NOP	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
Active	Yes[a]	Ignore[b]	Yes	Yes	Yes	Yes	Yes	Yes
L1	Ignore	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
LinkReset	Yes[a]	Yes	Yes	Ignore	Yes	Ignore	yes	Ignore
Retrain	Ignore	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
Disable	Yes[a]	Yes	Yes	Yes	Yes	Ignore	Yes	Ignore
L2	Ignore	Yes	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore
LinkError (边带 wire)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

- 在表中纵向的为请求，横向的为状态。
- YES：表明此请求被物理层考虑进行下一次转换。
- Ignore：表明此请求被忽略并且对于下一次状态转换没有影响。
- a:需要在请求该转换前先请求 NOP。
- b:如果当前状态为激活.PMNAK，则当较低层采样到激活的请求，则切换到激活。

7.2.5　边带访问

提供两大数据通路：主带：用来传输业务数据流；边带：用来处理边带访问业务，如链路训练、链路管理、参数交换及寄存器访问等控制流信息。边带作为主带的幕后通道，能够简化数据链路建立过程、提升主带的带宽利用率、简化主带设计复杂度。协议层控制状态信息主要通过边带传递给各层，边带访问模块主要通过消息传递和寄存器访问实现四种边带数据帧的传递。

三类不同的边带接口：

a) FDI边带：在协议层与链路层之间传递边带信息。FDI接口上跟边带相关的信号为 pl_cfg* 及 lp_cfg*，协议层可以经FDI边带向链路层发送寄存器访问请求，但链路层不能向协议层发送寄存器访问请求；

b) RDI边带：在链路层与物理层之间传递边带信息。RDI接口上有一组跟边带相关的信号（pl_cfg* 与 lp_cfg*），物理层从RDI接口接收到边带相关信息后，封装成帧转换为串行数据流通过PHY传输到对端。同理，对端PHY发来的串行边带数据在逻辑物理层解包，通过RDI口上的边带信号发送到链路层；

c) Link边带：在两个芯粒之间的物理层之间传递边带信息。跟FDI/RDI边带不同，链路边带数据线位宽只有1位，传输的是串行边带数据流；

7.2.5.1 寄存器访问

根据作用的不同，其寄存器也分布在协议层、链路层、物理层等各个层次，通过边带实现芯粒内不同层次间的直接寄存器访问，或者跨die的间接寄存器访问。

本地芯粒寄存器访问：同一芯粒内，上层通过边带访问下层寄存器；

远端芯粒寄存器访问：不同芯粒间，主芯片可以基于邮箱机制，由链路层发起边带寄存器访问操作，间接访问对端链路层和PHY层的相关寄存器（协议层可以通过访问本端链路层的邮箱相关寄存器实现对远端寄存器的访问）。

7.2.5.2 消息传递

通过在层间或者芯粒间的边带来传递消息，所示进而控制状态机的跳转，如图27：

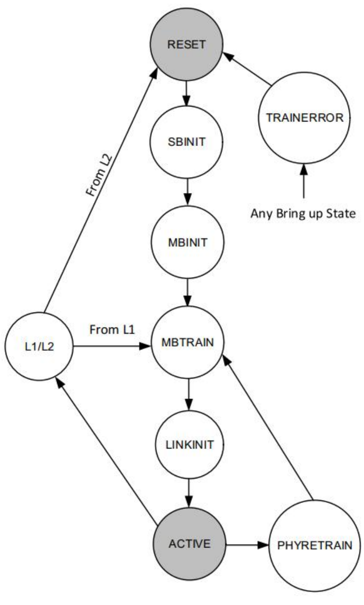


图 27 链路训练状态机

链路训练：SBINIT→MBINIT→MBTRAIN→LINKINIT→ACTIVE；

链路管理：通过边带消息实现PM\Retrain等链路状态管理；
参数交换：参数交换发生于链路初始化期间，主带初始化时在链路物理层之间通过边带交换训练参数，对主带进行训练。MBTRAIN时主带物理链路训练完毕，此时通过边带在链路层之间交换协议参数，协商要采用的协议及Flit格式。

7.2.5.3　边带数据帧格式

边带支持类似于PCIe的4种不同的数据包:配置读写(CfgRd/Wr)、内存读写(MRd/Wr)、完成(Cpl/Cpld)及消息（Msg/MsgD）。其中配置读写和内存读写都属于寄存器访问请求。

a)　配置读写（cfgrd/wr）：访问配置空间内的寄存器采用配置读写,具体帧格式见图28；
b)　内存读写(mrd/wr)：访问MMIO空间的寄存器采用内存读写,具体帧格式见图28；
c)　Completion(cpl/cpld)：一般对应寄存器访问请求的响应。根据是否携带数据及返回数据的位宽不同分为cpl/cpld，cpld又可分为32bit/64bit数据（指寄存器数据位宽）；其跟寄存器访问有几点不同：Status，表明当前 Completion 的状态，包括Success、UR、CA、Stall四种状态。Completion在FDI上的去向不依赖于dstid而是依赖于tag字段与Request对应，且cpl没有addr地址字段,具体帧格式见图29；
d)　消息(msg/msgD)：用于芯粒间参数交换、链路训练、链路管理及其他自定义场景。根据是否需要携带数据分为msg/msgD。跨die的链路训练/链路管理/参数交换的消息帧由链路层的边带访问模块传递，消息帧不会传递到协议层,具体帧格式见图30和图31。

<table>
<tr><th colspan="33">Register Access Request</th></tr>
<tr><th>Bytes</th><th colspan="8">3</th><th colspan="8">2</th><th colspan="8">1</th><th colspan="8">0</th></tr>
<tr><th>Bits</th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><th>Header / Data</th><th colspan="32">Header</th></tr>
<tr><td>Phase0</td><td colspan="3">srcid</td><td colspan="2">rsvd</td><td colspan="5">tag[4:0]</td><td colspan="8">be[7:0]</td><td colspan="8">rsvd</td><td>ep</td><td colspan="5">opcode[4:0]</td></tr>
<tr><td>Phase1</td><td>dp</td><td>cp</td><td>cr</td><td colspan="2">rsvd</td><td colspan="3">dstid</td><td colspan="24">addr[23:0]</td></tr>
<tr><th>Header / Data</th><th colspan="32">Data (if applicable, can be 32 bits or 64 bits)</th></tr>
<tr><td>Phase2</td><td colspan="32">data[31:0]</td></tr>
<tr><td>Phase3</td><td colspan="32">data[63:32]</td></tr>
</table>

图 28　寄存器访问请求帧格式

<table>
<tr><th colspan="33">Register Access Completions</th></tr>
<tr><th>Bytes</th><th colspan="8">3</th><th colspan="8">2</th><th colspan="8">1</th><th colspan="8">0</th></tr>
<tr><th>Bits</th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><th>Header / Data</th><th colspan="32">Header</th></tr>
<tr><td>Phase0</td><td colspan="3">srcid</td><td colspan="2">rsvd</td><td colspan="5">tag[4:0]</td><td colspan="8">be[7:0]</td><td colspan="8">rsvd</td><td>ep</td><td colspan="5">opcode[4:0]</td></tr>
<tr><td>Phase1</td><td>dp</td><td>cp</td><td>cr</td><td colspan="2">rsvd</td><td colspan="3">dstid</td><td colspan="21">rsvd</td><td colspan="3">Status</td></tr>
<tr><th>Header / Data</th><th colspan="32">Data (if completion with data, can be 32 bits or 64 bits)</th></tr>
<tr><td>Phase2</td><td colspan="32">data[31:0]</td></tr>
<tr><td>Phase3</td><td colspan="32">data[63:32]</td></tr>
</table>

图 29　寄存器 Completion 帧格式

<table>
<tr><th colspan="33">Messages without Data</th></tr>
<tr><th>Bytes</th><th colspan="8">3</th><th colspan="8">2</th><th colspan="8">1</th><th colspan="8">0</th></tr>
<tr><th>Bits</th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><th>Header / Data</th><th colspan="32">Header</th></tr>
<tr><td>Phase0</td><td colspan="3">srcid</td><td colspan="2">rsvd</td><td colspan="5">rsvd</td><td colspan="8">msgcode[7:0]</td><td colspan="9">rsvd</td><td colspan="5">opcode[4:0]</td></tr>
<tr><td>Phase1</td><td>dp</td><td>cp</td><td colspan="3">rsvd</td><td colspan="3">dstid</td><td colspan="16">MsgInfo[15:0]</td><td colspan="8">MsgSubcode[7:0]</td></tr>
</table>

图 30　边带 Msg 帧格式

<table>
<tr><th colspan="33">Messages with data</th></tr>
<tr><th>Bytes</th><th colspan="8">3</th><th colspan="8">2</th><th colspan="8">1</th><th colspan="8">0</th></tr>
<tr><th>Bits</th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><th>Header / Data</th><th colspan="32">Header</th></tr>
<tr><td>Phase0</td><td colspan="3">srcid</td><td colspan="2">rsvd</td><td colspan="5">rsvd</td><td colspan="8">msgcode[7:0]</td><td colspan="9">rsvd</td><td colspan="5">opcode[4:0]</td></tr>
<tr><td>Phase1</td><td>dp</td><td>cp</td><td colspan="3">rsvd</td><td colspan="3">dstid</td><td colspan="16">MsgInfo[15:0]</td><td colspan="8">MsgSubcode[7:0]</td></tr>
<tr><th>Header / Data</th><th colspan="32">Data</th></tr>
<tr><td>Phase2</td><td colspan="32">data[31:0]</td></tr>
<tr><td>Phase3</td><td colspan="32">data[63:32]</td></tr>
</table>

图 31　边带 MsgD 帧格式

边带将通过标准帧头中包含的操作码字段(Opcode)来区分数据包类型、地址位宽、是否携带数据等信息，如表25所示。

表 25　边带 Packet 操作码对应的帧类型

Opcode Encoding	Packet Type
00000b	32b Memory Read
00001b	32b Memory Write
00100b	32b Configuration Read
00101b	32b Configuration Write
01000b	64b Memory Read
01001b	64b Memory Write
01100b	64b Configuration Read
01101b	64b Configuration Write
10000b	Completion without Data
10001b	Completion with 32b Data
11001b	Completion with 64b Data
10010b	Message without Data
11011b	Message with 64b Data
Other encodings	Reserved

1.1.1.1　边带流控机制

边带数据包在FDI、RDI及链路接口上进行传输时独立进行基于信用的流控。在边带流控中，不论是否携带数据、不论数据位宽多少（目前只有 32、64bit 两种），同一个边带头信息消耗的就是一笔信用，即每一笔信用对应64bit头信息w/wo数据。

对于发送端，Tx向Rx发送寄存器访问请求或消息类型的边带包之前，必须先检查Rx是否还有信用余量。对于接收端，其在处理完相关边带报文之后，及时释放信用出来。发送完成是不占用信用的，发送完成之前无需确认Rx信用情况，Rx收到发送完成后应无条件立即执行。

7.2.5.4 超时统计

为了保证参数交换的有效性，边带处设置了一个8毫秒的超时机制。计时器只会在RDI处于活跃状态时计时。计时器在收到来自对端链路终端发送来的{AdvCap.*.Stall}或{FinCap.*.Stall}时需要执行重置操作。在解析过程中，Retimer有责任需要每4毫秒发送一次对应的stall信息，以确保其他芯粒不会处于超时状态。若寄存器访问请求超时，边带会上报超时中断并返回UR（unsupported request）完成信息。

7.2.6 中断

宜在链路层的内部汇聚管理各类中断，包含链路事件、物理层异常事件、边带信号消息中断、异常报文数据中断等异常事件。

7.2.6.1 PHY 层中断

表26所示PHY层通过RDI接口上报下表所列中断事件至链路层：

表 26 PHY 层中断

中断信号	中断来源	中断描述
pl_error	PPHY	物理层向链路层发出指示，表明其检测到一个与帧相关的错误，该错误可以通过链路重新训练恢复。例如，物理层在有效通道上接收到无效编码。这是一个持续一个或多个时钟周期的脉冲信号，且仅当 RDI 处于活跃状态时发生。允许在状态从活动状态转换的同一时钟沿取消断言。 该错误指示与接收数据路径同步，使得错误指示到达适配器的时间早于或与损坏数据到达的时间相同。物理层在断言此信号后，应经历重新训练流程，并且在链路重新训练完成之前，不得向链路层发送有效数据。 物理层允许在内部对损坏数据的 pl_valid 信号进行抑制。一旦 pl_error 被断言，在状态从成功完成重新训练的进入和退出后重新过渡到活动状态之前，不应断言 pl_valid（除非在同一时钟周期内断言 pl_error）。 如果在同一时钟周期内 pl_error=1 且 pl_valid=1，适配器必须丢弃对应的 Flit（即使在断言 pl_error 时 Flit 仅部分接收）。

		在 Flit 模式下，当启用重传时，链路层负责确保转发到 FDI 的 Flit 的数据完整性，并根据 pl_flit_cancel的规则取消那些被认为可能损坏的 Flit。以下给出了一些示例： 对于 68B Flit 格式，链路层可以丢弃部分接收的 Flit；但在 256B 低延迟优化模式下，可能已经正确处理了一半的数据，而错误发生在另一半上，因此需要跟踪并相应地处理未来的 Flit。 另一个示例是，如果链路层未执行存储/转发操作，并且仅接收到 128B 半段中的 64B，而 pl_error 发生在接收到剩余 64B 之前，则需要为该 Flit 的另一半发送虚拟数据，并执行 pl_flit_cancel。 在Flit 模式下，当链路层启用重传时，重新训练退出将自然导致任何部分接收的 Flit 的重发。在 Flit 模式下，当链路层禁用重传时，必须将 pl_error 断言映射到不可纠正的内部错误，并相应地升级处理。 如果链路以Raw Mode运行，链路层将 pl_error 转发到协议层，使其与数据总线同步匹配，协议层以实现特定的方式处理该错误。
pl_cerror	PHY	物理层向链路层发出指示，表明检测到一个可纠正的错误，该错误不会影响数据路径，也不会导致链路上的重新训练。在 Flit 模式下且启用了重传时，适配器必须将 pl_error 和 pl_cerror 信号进行逻辑或运算，以用于可纠正内部错误日志记录。 在 Flit 模式下且重重传被禁用，或者链路以Raw Mode运行时，链路层只能使用 pl_cerror 信号进行可纠正内部错误日志记录。 这是一个持续一个或多个时钟周期的脉冲信号，可以在任何 RDI 状态下发生。如果该信号发生在允许时钟门控的状态中，物理层有责任在断言此信号之前与适配器完成时钟门控退出握手。一旦 pl_cerror 取消断言，并且满足所有其他允许时钟门控的条件，时钟门控即可重新启动。
pl_nferror	PHY	物理层向链路层发出指示，表明检测到一个非致命错误。目前，物理层没有定义任何架构上的错误条件来触发此信号的断言；然而，该信号被提供在接口上，用于任何实现特定的非致命错误。链路层以与接收到远程链路伙伴发送的侧带非致命错误消息相同的方式处理此信号。 这是一个持续一个或多个时钟周期的脉冲信号，可以在任何 RDI 状态下发生。如果该信号发生在允许时钟门控的状态中，物理层有责任在断言此信号之前与适配器完成时钟门控退出握手。一旦 pl_nferror 取消断言，并且满足所有其他允许时钟门控的条件，时钟门控即可重新启动。
pl_trainerror	PHY	指示来自物理层的致命错误。如果物理层尚未处于 LinkError状态，必须将 pl_state_sts转换为 LinkError。

		根据链路层中不可纠正内部错误的掩码和严重性配置，此错误必须上报至更高的协议层。根据系统级需求，实现可以将任何需要上报至上层（或生成中断）的致命错误映射到此信号。 这是一个电平信号，可以在任何 RDI 状态下断言，但会保持断言状态，直到 RDI 从 LinkError状态退出并进入复位状态。

7.2.6.2 中断接口

链路层自PHY接收到的中断捕获中断状态后，通过表27 FDI接口上报PHY异常事件至协议层：

表 27 FDI 中断

中断信号	中断来源	功能描述
fdi_pl_error	链路层	链路层检测到一个帧错误。链路层将在 RDI 上发起链路重新训练。
fdi_pl_cerror	链路层	链路层检测到一个可纠正的错误，这不会导致链路上的重新训练。
fdi_pl_nferror	链路层	链路层向协议层发出指示，表明检测到一个非致命错误。
fdi_pl_trainerror	链路层	指示来自链路层的致命错误。如果物理层尚未处于 LinkError 状态，则必须将 rdi_pl_state_sts 转换为 LinkError 状态。此错误必须上报至更高的协议层以进行处理。

7.2.6.3 异常事件

建议在链路层内部汇聚各类中断源，并提供了一个中断信号adp_int，用于通知用户重要事件。该信号可由以下来源驱动：初始化完成（非致命）、检测到 CRC 错误（非致命）、重传超时（致命）、重传缓冲区满（非致命）、激励校验失败（致命）、延迟计算完成（debug）。

这些事件根据其严重性被分类为致命或非致命，并通过中断信号adp_int通知用户以采取相应的处理措施。

7.2.6.4 中断列举

表28列举了所支持中断的详细信息：

表 28 中断列表

中断名称	中断类型	中断说明
InbandPres	指示	表示芯粒间链路已经完成协商，并将 FDI 状态转为活跃

ToForPartialFlit	指示	提醒协议层发剩下的 flit
PatternCheckerFail	指示	激励校验失败标志
RetryBufferFull	指示	重传缓存满标志
LatencyCalculationDone	指示	延迟计算完成标志
NormalPhaseEntry	指示	进入正常阶段标志
CRC_MetTrsh	指示	接受侧 CRC 错误达到阈值
fdi_lb_buffer_full	指示	FDI 环回路径的缓存满
seqnum_handhs_error	指示	收到错误序列号的标志
HwAutoBwChanged	指示	硬件自动改变链路位宽和速率以修正可靠性变化标志
LkStsChanged	指示	链路状态变化标志
rdi_pl_error	错误	RDI 上报的错误
CorrectableErrMessagereceived	可纠错误	收到边带可纠正错误的消息的标志位
CRCErrDetected	可纠错误	检测到 CRC 错误
LSMTransitionToRetrain	可纠错误	LSM 到重训练的标志位
CorrectableInternalErr	可纠错误	可纠正错误中断
RuntuneLinkTestingParityErr	可纠错误	链路测试检测到极性错误
rdi_pl_cerror	可纠错误	RDI 上报的可纠正的错误
SbNonFatErrMsgRcved	非致命错误	收到来自边带的非致命错误消息
rdi_pl_nferror	非致命错误	RDI 上报的非致命错误
SbFatErrMsgRcved	训练错误	收到来自边带的致命错误消息标志位
AdpTO	训练错误	链路超时标志
RcvOFlow	训练错误	边带、RDI、重传缓存溢出标志
UncorInternalErr	训练错误	数据通路不可纠正的错误
InParaRx	训练错误	发现非标准协议时产生的中断
rdi_pl_trainerror	训练错误	RDI 训练失败中断

7.2.7 链路层环回

链路层的环回测试模式，用于测试信号完整性，硬件功能或故障排查。其核心机制为通过主设备发送数据，从设备原样回传，检测链路传输稳定性。

环回既可以在本端内测环回，也可以在对端数据外侧环回，支持通过环回进行链路延迟计算。

如图32在内测环回时，TX方向FDI接收到数据给RDI时，RDI会将数据环回给RDI的RX方向，然后数据传递到FDI的Rx。数据通路与正常通路基本一致区别在于正常数据由RDI口发送给对端，RDI内测环回由RDI口MUX至本端的RX。

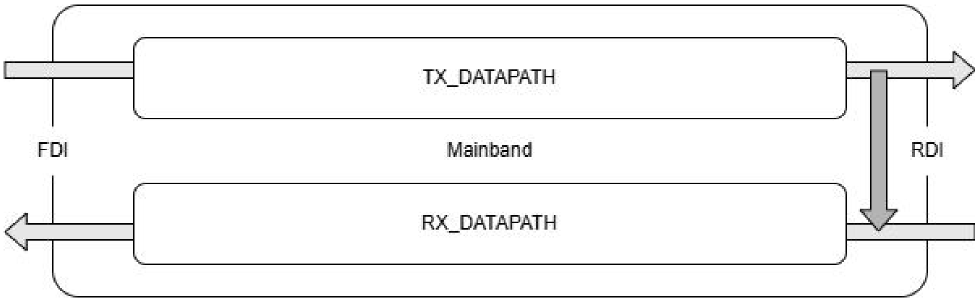


图 32 RDI 回环

如图33在外侧环回测试时，RX方向RDI接收到数据给FDI时，会通过一个FIFO回环给FDI的TX，TX的数据通路将传递到RDI的TX；但FDI环回不支持测全速。若进行FDI环回的链路层发生重传，全速发送数据的对端链路层持续将数据发送给本端进行重传的链路层,将会使得本端的TX 重传缓存发生溢出,丢失数据。因此，在于正常通路数据由协议层经FDI口发送给本端TX,FDI外侧环回由本端RX将数据MUX至本端TX的FDI侧入口，且进入TX时先存入一个lb_fifo，且限制对端TX发送的速率。

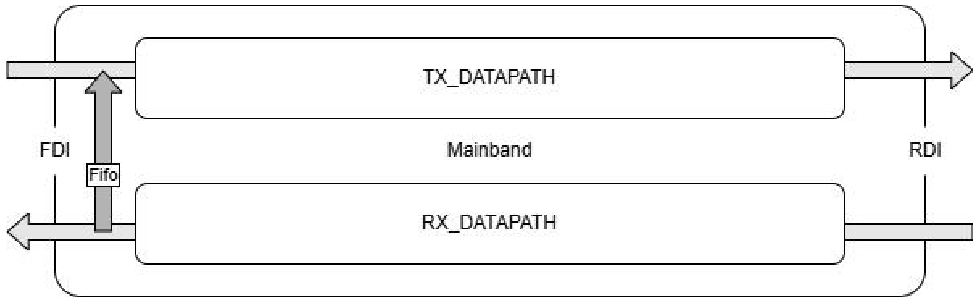


图 33 FDI 回环

7.3 物理层

物理层定义了PHY的相关电气特性，封装芯粒和PHY摆放位置，如图3所示接口2。

封装宜采用先进封装，可采用标准封装。

7.3.1 基于先进封装的物理层要求

宜采用先进封装技术用于性能优化的应用场景。因此，通道距离很短（从一个晶粒上的bump到远端晶粒的bump连接距离小于等于2mm)，并且互连有望针对高带宽和低延迟进行优化，具有最佳性能和能效特性。下面显示了使用先进封装的示例应用和先进封装的主要特征摘要，如图34和表29。

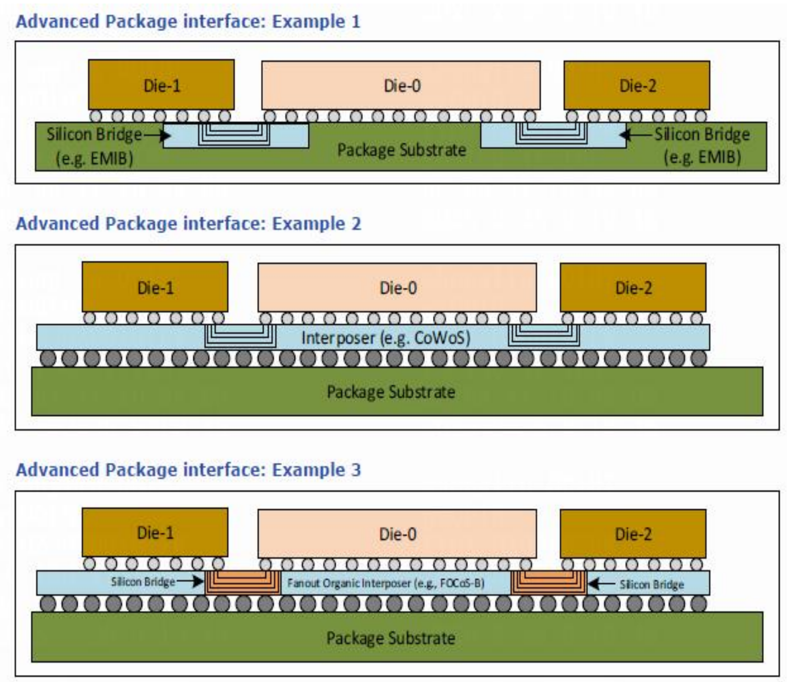


图 34　先进封装

表 29　先进封装摘要

指标	值
支持速率/Lane	4GT/s，8GT/s，12GT/s，16GT/s，24GT/s，32GT/s
Bump Pitch	25um - 55um
通道长度	<= 2mm
Raw Bit Error Rate（BER）	1e-27 (<=12GT/s)
	1e-15 (>=16GT/s)

Bump上的一组主数据链路被称为一个模块。该模块构成了AFE的结构设计实现的最小粒度。物理链路由两种类型的连接组成：边带信号，主数据信号。

1. 边带信号：

此链接用于参数交换、用于调试、合规性的寄存器访问以及与对端协调，以进行链接建立和管理。它在每个方向上由一个前向时钟引脚和一个数据引脚组成。无论主带数据速率如何，时钟都固定为800MHz。物理层的边带逻辑必须采用辅助电源和“始终开启”域。每个模块都有自己的一组边带信号。

对于先进封装模块，每个方向都提供了一对冗余的时钟和数据引脚用于修复。

2. 主数据信号：

此连接构成了主数据路径。它由一个转发时钟、一个有效数据引脚、一个跟踪引脚和每个模块的N个Lane数据通道组成。

对于先进封装，见表30，bump图中提供了N=64（即x64位）和额外的四个用于通道修复的引脚。

对于标准封装模块，N=16（即x16位），且没有提供额外的修复引脚。

表 30　先进封装 TX/RX Signal

信号名	数量	描述
主数据信号		
TXDATA [63:0]	64	传输数据
TXVLD	1	传输数据有效；启用相应模块的时钟
TXTRK	1	传输跟踪信号
TXCKP	1	传输时钟相位 1
TXCKN	1	传输时钟相位 2
TXCKRD	1	用于时钟和轨道带信号修复的冗余信号
TXDATARD [3:0]	4	用于数据修复的冗余信号
TXVLDRD	1	传输数据有效的冗余信号
RXDATA [63:0]	64	接收数据
RXVLD	1	接收数据有效；启用相应模块的时钟
RXTRK	1	接收跟踪信号
RXCKP	1	接收时钟相位 1
RXCKN	1	接收时钟相位 2
RXCKRD	1	用于时钟和轨道带信号修复的冗余信号
RXDATARD [3:0]	4	用于数据修复的冗余信号
RXVLDRD	1	接收数据有效的冗余信号
边带信号		
TXDATASB	1	边带传输数据信号
RXDATASB	1	边带接收数据信号
TXCKSB	1	边带传输时钟信号
RXCKSB	1	边带接收时钟信号
TXDATASBRD	1	用于边带传输数据修复的冗余信号
RXDATASBRD	1	用于边带接收数据修复的冗余信号
TXCKSBRD	1	用于边带传输时钟修复的冗余信号
RXCKSBRD	1	用于边带接收时钟修复的冗余信号

先进封装模块引脚相比于标准封装模块，数据信号部分，TX/RX 数据从 16 位增加为 64 位，并且增加了 TX/RXRD[3:0]和 TX/RXCKRD,TX/RXVLDR 信号。边带信号部分，增加了 TX/RXDATASBRD 及 TX/RXCKSBRD

信号。总体信号数量从标准封装模块的 44 个增加到 156 个，见表 30。

7.3.1.1 先进封装 Bump Map

为PHY实施提供一组参考64位bump Map，以包含指定的最大速度。这些bump map采用的模式，间距范围，速率选择等具体内容。

Bump map选项的相应最大速度及其建议的bump间距范围见表31。

表 31 先进封装 Bump Map

Bump Map	Bump Pitch(um)	Max Speed
16 column	25-30	12
	31-37	16
10 column	38-44	24
	45-50	32
8 column	51-55	32

Bump Map采用8列的bump Pattern，以追求最好的实施要求和最大的性能目标。基于业内大多数先进的封装技术，针对信号完整性、电源完整性、通道间偏斜、电迁移应力和凸块面积进行了优化。具体参考下图。

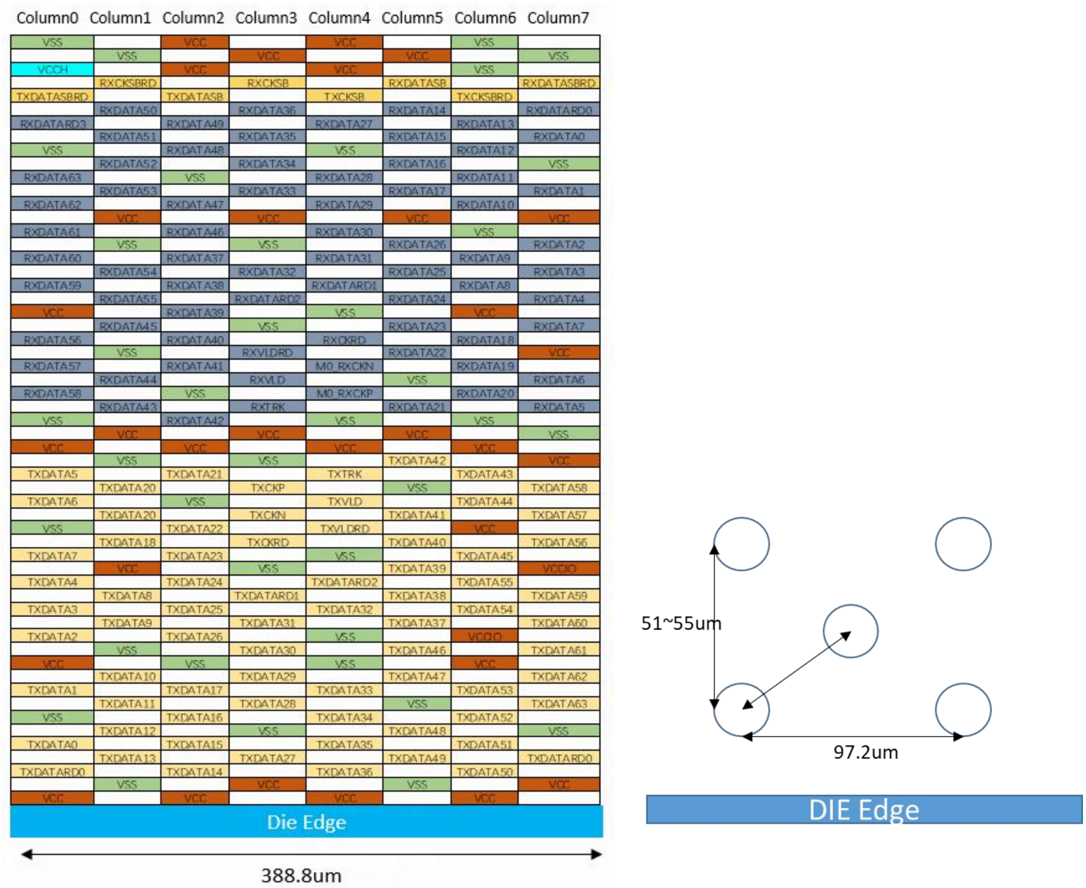


图 35　先进封装 Bump Map

先进封装的bump map兼容UCIe 2.0，见图35，相比于标准封装，bump map互连不会出现长对长，短对短的互连形式，对于数据倾斜的控制更为友好。

7.3.1.2　先进封装电气特性

先进封装的电气特性定义和选择，具体内容和定义见下描述：

先进封装的电气摘要见表32：

表 32　先进封装 Phy 电气参数

参数	先进封装
数据位宽(/模块)	64
数据速率(GT/s)	4/8/12/16/24/32
通道长度(mm)	2
PHY 面宽(um)	388.8

输出端电气规格件表33：

表 33　先进封装 Phy 电气规格

	最小值	典型值	最大值	单位
信号输出幅度	0.4			V
单端时钟上升下降时间	0.1	0.22	0.25	UI
数据信号上升下降时间		0.35		UI
驱动高低电平电阻	22	25	28	Ohms
信号偏移修正范围(16 GT/s)	-0.1	-	0.1	UI
信号偏移修正范围(32 GT/s)	-0.15	-	0.15	UI
发送端寄生电容(16 GT/s)			200	fF
发送端寄生电容(24、32 GT/s)			125	fF

建议对16 GT/s进行发射机均衡，并且必须在24 GT/s和32 GT/s数据速率下支持，以减轻通道ISI的影响。Tx均衡仅对所有适用的数据速率进行去加重。

24 GT/s和32 GT/s的Tx均衡系数基于下图输出去加重所示的FIR滤波器。均衡系数受最大单位摆动约束的约束。

发射器必须支持表中所示的均衡设置。去加重设置的确定基于初始配置或训练序列，其中将选择眼图张开度较大的值。

输出去加重如图36所示：

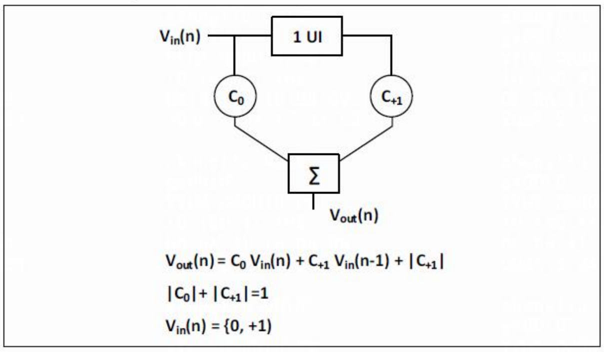


图 36　先进封装输出去加重

输出去加重波形如图37所示：

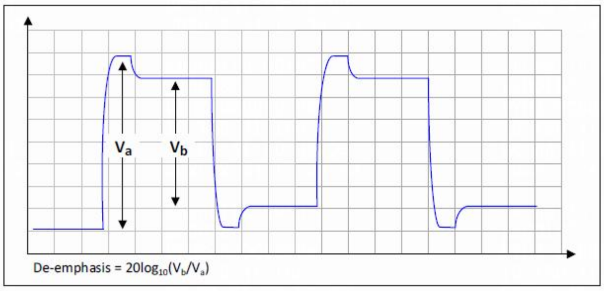


图 37　输出去加重波形

输出去加重值见表34：

表 34　先进封装输出去加重值

设置	去加重	精准度	C+1	Vb/Va
1	0.0dB	-	0.000	1.000
2	-2.2dB	±0.5dB	-0.112	0.776

接收器均衡可以在24 GT/s和32 GT/s数据速率实现，如图38所示，这样，即使TX均衡不可用，也能实现链接操作。可以通过CTLE、一阶DFE或其他实现。预期的RX均衡能力相当于一阶CTLE。

$$H(s) = \omega_{p2}\left(\frac{s + A_{DC}\omega_{p1}}{(s+\omega_{p1})(s+\omega_{p2})}\right)$$

where, ω_{p2} = 2π*DataRate, ω_{p1} = 2π*DataRate /4, and A_{DC} is the DC gain.

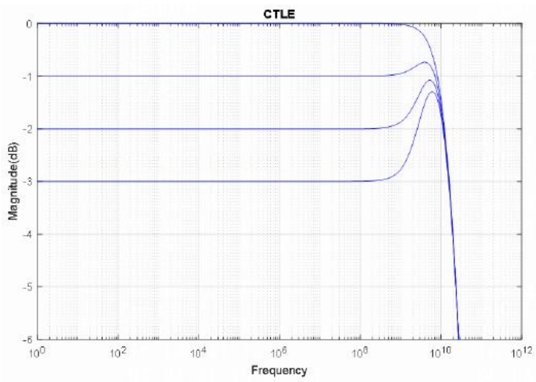


图 38　接收器均衡

先进封装模块的通道约束，VTF指标见表35，需要注意先进封装模块默认不开启接收端的ODT，在计算通道指标时，TX的端接为25ohm/0.25pF，RX的端接为0.2pF。

表 35　先进封装 VTF 指标

	4-16 GT/s	24-32 GT/s
VTF Loss (dB)	L(fN) > -3	L(fN) > -5
VTF Crosstalk (dB)	XT(fN) < 1.5 L(fN)-21.5 and XT(fN) < -23	XT(fN) < 1.5 L(fN)-19 and XT(fN) < -24

互连通道需要满足表36规定的最小矩形眼张开度要求，在通道一致性仿真条件下，使用无噪声和无抖动行为 TX 和 RX 模型，如图39所示。

表 36　先进封装通道眼图数值

接口速率(GT/s)	眼高(mV)	眼宽(UI)
4,8,12,16	40	0.75
24,32	40	0.65

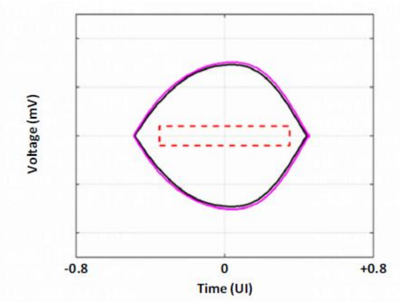


图 39　先进封装通道眼图示例

7.3.1.3　先进封装电源规范

互通时推荐两方信号电平维持一致，见表37，Analog IO Supply采取0.5V的低电压模式，在提供良好的信号驱动和时序的同时可以降低功耗需求。

表 37　先进封装电源规范

电源	最小电压	典型电压	最大电压	Unit	描述
VCC	-5%	0.75	+10%	V	PHY core supply
VCCIO	0.45	0.5	+10%	V	Analog IO supply
VCCH	-5%	1.2	+10%	V	PLL Analog supply

7.3.2　基于标准封装的物理层要求

可采用标准技术用于低成本和长距离（从一个晶粒上的bump到远端晶粒的bump连接距离为10mm至25mm）,使用有机封装基板上的走线，与非封装的SerDes相比能提供明显更好BER的特性。下面显示了使用标准封装的示例应用如图40所示和标准封装的主要特征摘要见表38。

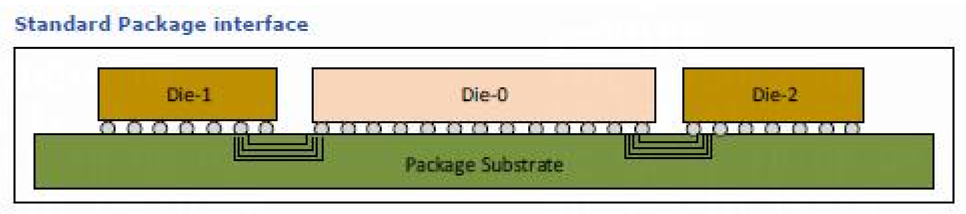


图 40　标准封装

表 38　标准封装摘要

指标	值
支持速率/Lane	4GT/s，8GT/s，12GT/s，16GT/s，24GT/s，32GT/s
Bump Pitch	150um
通道长度	5mm – 25mm
Raw Bit Error Rate（BER）	1e-27 (<=8GT/s)
	1e-15 (>=12GT/s)

7.3.2.1 标准封装 Bump Map

标准封装的bump Map如图41所示，建议bump间距控制在150um，可方便信号在有机基板上无障碍互连，保障信号与电源完整性。

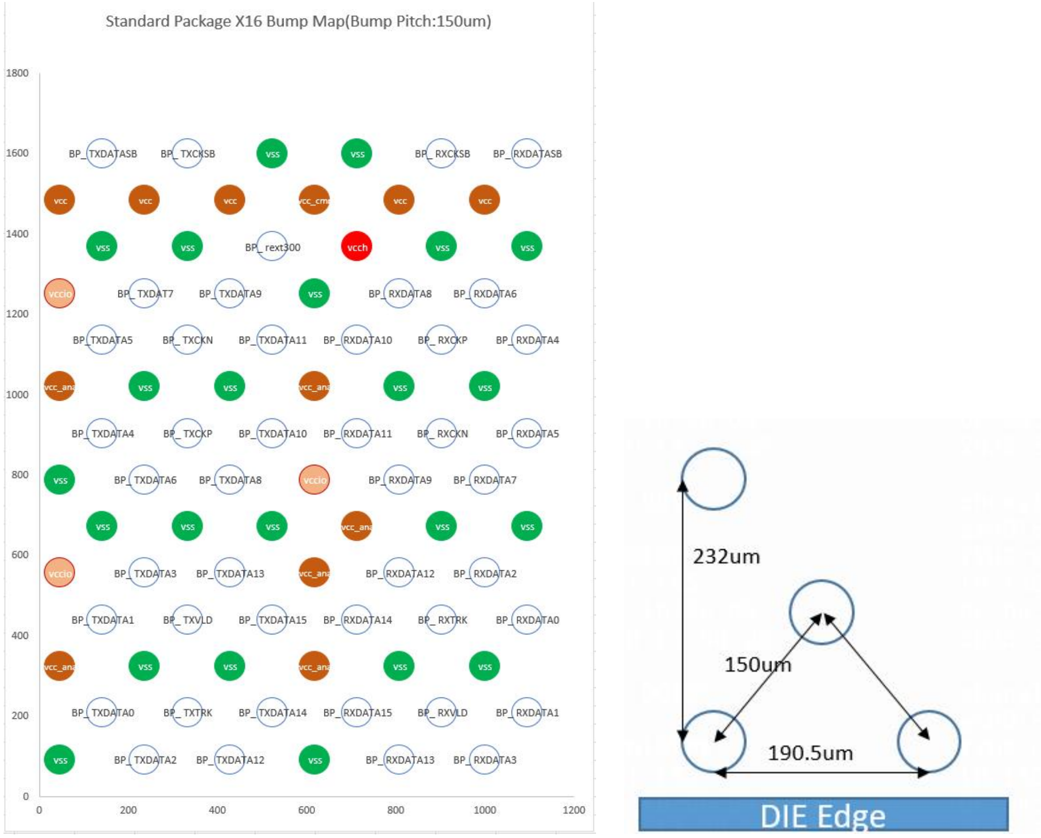


图 41 标准封装 Bump Map

7.3.2.2 标准封装电气规范

标准封装的电气规范等数据，通道眼图等物理层，具体内容描述见下面描述：

标准封装的电气摘要见表39：

表 39 标准封装 Phy 电气参数

7.3.2.3 参数	7.3.2.4 标准封装
数据位宽(/模块)	16
数据速率(GT/s)	4/8/12/16/24/32
通道长度(mm)	5 - 25
PHY 面宽(um)	1143
PHY 深度(um)	1691.136 *实际深度取决于 PHY 的 bump pitch 规划

输出端电气规格见表40：

表 40 标准封装 Phy 输出电气规格

	最小值	典型值	最大值	单位
信号输出幅度	0.4			V
单端时钟上升下降时间	0.1	0.22	0.25	UI
数据信号上升下降时间		0.35		UI
驱动高低电平电阻	27	30	33	Ohms
信号偏移修正范围(16 GT/s)	-0.14	-	0.14	UI
信号偏移修正范围(32 GT/s)	-0.22	-	0.22	UI
发送端寄生电容(16 GT/s)			200	fF
发送端寄生电容(24、32 GT/s)			125	fF

接收端电气规格见表41：

表 41 标准封装 Phy 接收电气规格

	最小值	典型值	最大值	单位
接收端端接阻抗	45	50	55	Ohms
输出上升时间			0.1	UI
输出下降时间			0.1	UI
信号偏移修正范围(16 GT/s)	-0.07		0.07	UI
信号偏移修正范围(>16 GT/s)	-0.012		0.012	UI
接收端寄生电容(16 GT/s)			200	fF
接收端寄生电容(24、32 GT/s)			125	fF

标准封装模块的通道约束，VTF指标如下，在计算通道指标时，在4-8Gbps时，TX的端接为30ohm/0.3pF，RX的端接为50ohm/0.3pF，在12-16Gbps时，TX的端接为30ohm/0.2pF，RX的端接为50ohm/0.2pF；16Gbps以上时，TX的端接为30ohm/0.125pF，RX的端接为50 ohm/0.125pF。

	4-8 GT/s	12-16 GT/s	24-32 GT/s
VTF Loss (dB)	L(0) > -4.5 L(fN) > -7.5	L(0) > -4.5 L(fN) > -6.5	L(0) > -4.5 L(fN) > -7.5
VTF Crosstalk (dB)	XT(fN) < 3*L(fN)-11.5 and XT(fN) < -25	XT(fN) < 3*L(fN)-11.5 and XT(fN) < -25	XT(fN) < 2.5*L(fN)-10 and XT(fN) < -26

互连通道需要满足表中规定的最小矩形眼张开度要求见表42，在通道一致性仿真条件下，使用无噪声和无抖动行为 TX 和 RX 模型。

表 42　标准封装通道眼图数值

接口速率(GT/s)	眼高(mV)	眼宽(UI)
4,8,12,16	40	0.75
24,32	40	0.65

7.3.2.5　标准封装电源建议

互通时推荐两方信号电平维持一致，Analog IO supply采取0.8V的高电压模式或者IP供应商推荐的值，见表43，在提供强大的信号驱动能力时，最大程度满足通道在标准封装基板上的长距离互连。

表 43　标准封装电压规范

电源	最小电压	典型电压	最大电压	Unit	描述
VCC	-10%	0.8	+10%	V	PHY core supply
VCCIO	-10%	0.8	+10%	V	Analog IO supply
VCCANA	-10%	0.8	+10%	V	Analog supply
VCCCMN	-10%	0.8	+10%	V	Analog supply
VCCH	-10%	1.5	+10%	V	PLL Analog supply

7.3.3　时钟频率与相位

下表显示了在不同数据速率下必须支持的时钟频率和相位。在 24 GT/s 和 32 GT/s 时，接收器可以选择支持差分时钟或正交时钟。但为了实现与较低最大数据速率设计的互操作性，差分时钟必须始终在 16 GT/s 及以下使用，见表44。

表 44　时钟频率与相位

Data rate（GT/s）	Clock freq.（fCK）（GHZ）	Phase-1	Phase-2	Deskew(Req/Opt)
32	16	90	270	Required
	8	45	135	Required
24	12	90	270	Required
	6	45	135	Required
16	8	90	270	Required
12	6	90	270	Required
8	4	90	270	Optional
4	2	90	270	Optional

8　通信性能要求

8.1　带宽

芯粒封装形式的芯片内总线互联带宽要求如下：

a)　芯片内总线互联带宽包括DDR带宽、HBM带宽等。

b)　云侧芯片总线互联双向带宽应不低于128GB/s，宜达到256GB/s,可达到900GB/s及以上。

c）边缘侧芯片总线互联双向带宽应不低于15GB/s，宜达到64GB/s,可达到408GB/s及以上。

d)　端侧芯片总线互联双向带宽应不低于10GB/s，宜达到102.4GB/s,可达到128GB/s及以上。

8.2　时延

AI芯片延迟要求如下：

a）　边缘侧芯片：卡间互联接口的延迟宜小于100μs，可达到1μs，以减少训练过程中的通信开销。

b）　云侧芯片：卡间互联接口的延迟宜小于10μs，可达到1μs，以满足实时推理的需求。

9　其它要求

9.1　拓扑结构

a）训练芯片：宜采用高带宽、低延迟的拓扑结构，例如胖树、3D Torus或Dragonfly。

b）推理芯片：宜采用简单高效的拓扑结构，例如星型拓扑、总线拓扑、环形拓扑和网格拓扑。

9.2　组网规模

卡间互联接口应具有良好的可扩展性，在性能（如带宽或延迟等）保持相对稳定时，支持节点数从几个到数千个节点扩展。

附　录　A
（资料性）
先进封装

A.1　概述

GPU与通信芯粒合封，由于GPU和通信芯粒面积较大，同时对芯粒间互连速率要求较高，导致整体封装尺寸较大，功耗较高，信号完整性和电源完整性问题严重，封装良率相对较低，封装成本较高。

GPU与通信芯粒、HBM的合封方式1示意图如图A.1所示。一颗GPU + 4颗HBM + 2颗通信芯粒采用先进封装合封在一起，通信芯粒的上下两侧均有先进封装 PHY，通过在中介层上的互连线，完成通信芯粒与GPU的互联，通信芯粒侧面高速Serdes PHY，用于与外部GPU或其他芯片互联。

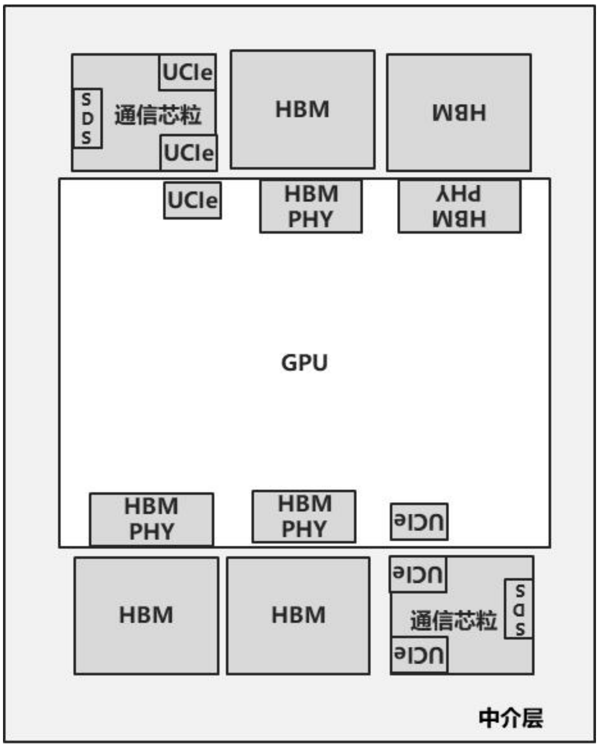


图 A.1　GPU 合封示意图方式 1

GPU与通信芯粒、HBM的合封方式2示意图如图A.2所示。一颗GPU + 2颗HBM + 2颗通信芯粒同样采用先进封装合封在一起，通信芯粒的上下两侧的先进封装 PHY，通过在中介层的互连线，完成通信芯粒与GPU的互联，通信芯粒侧面的高速Serdes PHY，用于与外部GPU或其他芯片互联。

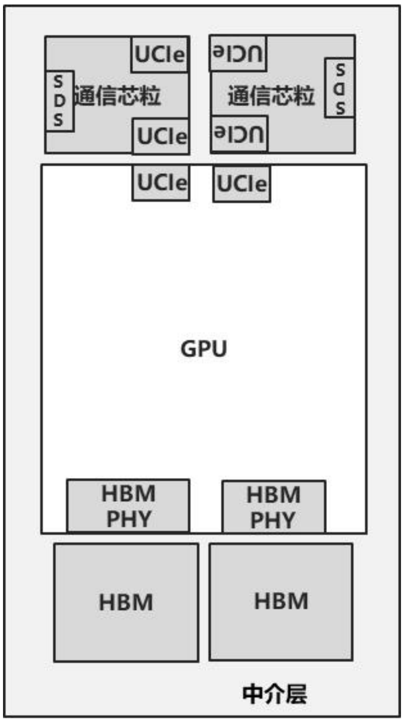


图 A.2　GPU 合封示意图方式 2

A.2　通信芯粒封装

A.2.1　通信芯粒Floorplan

先进封装通信芯粒Floorplan，使用2个X64位单元与主芯片交互，如图A.3。

图 A.3　先进封装 Die 尺寸

A.2.2　通信芯粒基本信息

芯粒基本信息如表A.1：

表 A.1　芯粒基本信息

Chip Size	10.300128mm*10.30008mm (wo seal ring)
Chip Thickness	780um
Die size shrinkage	
Min bump Pitch	50 - 55um
UBM Size	25.02um
ESD requirement	CDM 500V；HBM 2000V

A.2.3　通信芯粒POD

Wafer提供KGD与GPU进行合封，切割道宽度等。

A.2.4　通信芯粒Micro Bump规格

Wafer可提供Bumped Wafer或Wafer without Bumping。如需包含Bumping部分，芯粒Micro Bump Cell规格如表A.2和图A.4：

表 A.2　芯粒 Bumping 规格建议

Design Guidelines		Pillar Bump Specification
1	Bump Diameter	25um
2	Bump Space	20-30um
3	Bump Pitch	45-55um
4	Bump Height	>= 36um
5	Bump Material	Cu
6	Bump Material	SnAg
7	Passivation Opening	18um
8	Passivation Open to Bump Space	3.5um
9	Al Pad Size	29um

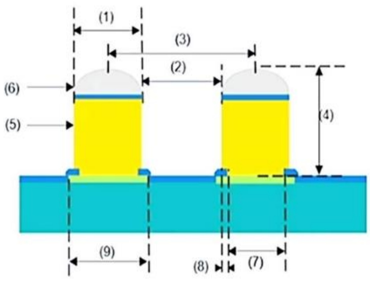


图 A.4　Micro Bump Dimension

A.2.5　通信芯粒Micro Bump信号

通信芯粒Micro Bump信号具体见表A.3：

表 A.3　互连 ubump list

信号名	数量	描述
主数据信号		
TXDATA [63:0]	64	传输数据信号
TXVLD	1	传输数据有效;启用相应模块的时钟
TXTRK	1	传输跟踪信号
TXCKP	1	传输时钟相位 1
TXCKN	1	传输时钟相位 2

TXCKRD	1	用于时钟和轨道带信号修复的冗余信号
TXDATARD [3:0]	4	用于数据修复的冗余信号
TXVLDRD	1	
RXDATA [63:0]	64	
RXVLD	1	
RXTRK	1	
RXCKP	1	
RXCKN	1	
RXCKRD	1	
RXDATARD [3:0]	4	
RXVLDRD	1	
边带信号		
TXDATASB	1	边带传输数据信号
RXDATASB	1	
TXCKSB	1	边带传输时钟信号
RXCKSB	1	
TXDATASBRD	1	用于边带数据修复的冗余信号
RXDATASBRD	1	
TXCKSBRD	1	用于边带时钟修复的冗余信号
RXCKSBRD	1	

A.2.6 通信芯粒2.5D封装集成架构

通信芯粒可适配如图A.5和图A.6所示Floor Plan的2.5D集成封装，相关结构经翘曲与应力仿真，可满足组装与可靠性要求。

注：斜线标注位置为通信芯粒。

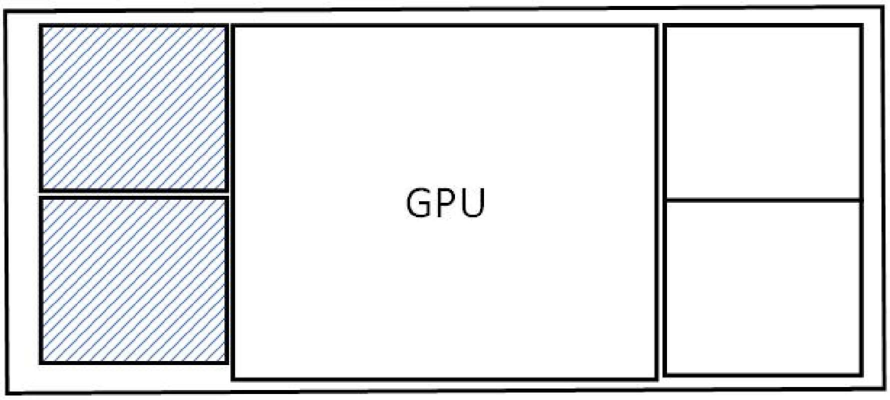


图 A.5 集成通信芯粒 Floor Plan 建议 1

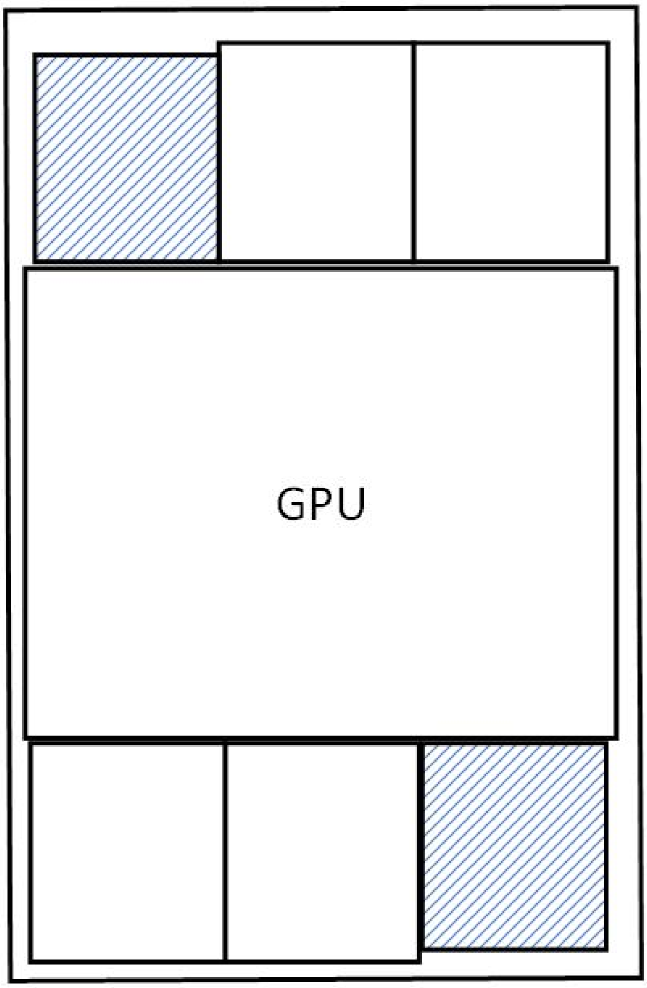


图 A.6 集成通信芯粒 Floor Plan 建议 2

A.3 主芯片封装

A.3.1 主芯片Floorplan

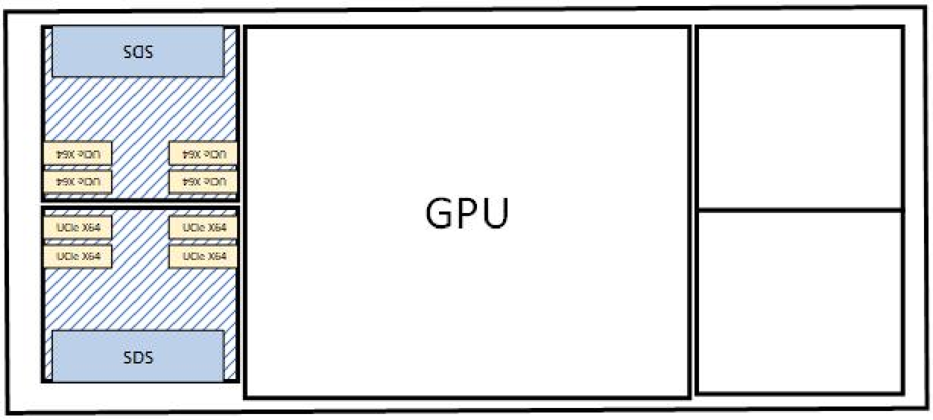
先进封装芯片Floorplan，使用2个X64单元与通信芯粒间交互。定义PHY的物理位置。

A.3.2　主芯片2.5D封装集成架构

主芯片可适配如下Floor Plan的2.5D集成封装，相关结构经翘曲与应力仿真，可满足组装与可靠性要求。先进封装 Die 2.5D封装集成架构。

图A.7和图A.8为通信芯粒与主芯片的两种互连案例。一种是将通信芯粒放置在主芯片一侧，由于通信芯粒左右两侧都预留了PHY位置，上下通信芯粒通过旋转180° 摆放，同时不影响通信芯粒是选择位于主芯片左侧还是右侧，主芯片另一侧则可以摆放HBM，构成1+2+2的芯片结构。另外一种是将通信芯粒放置在主芯片的对称角上，由于通信芯粒左右两侧都预留了PHY位置，所有无论是左上角+右下角，还是左上角+右下角的通信芯粒摆放结构，都可以实现。案例构成了1+4+2的芯片结构。

注：斜线标注位置为通信芯粒。



图A.7　主芯片集成通信芯粒Floor Plan建议1

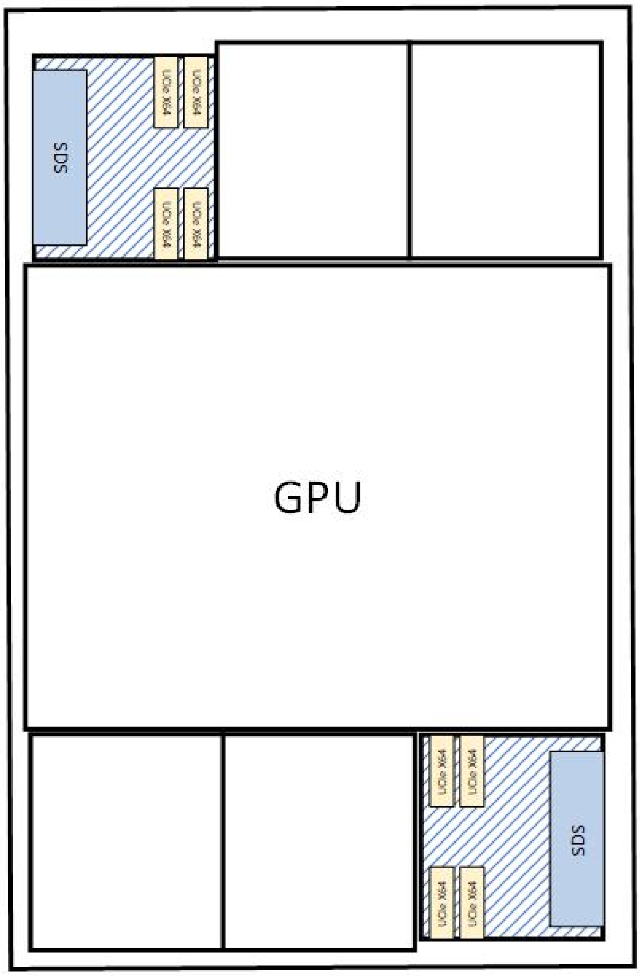


图 A.8　主芯片集成通信芯粒 Floor Plan 建议 2

附 录 B
（资料性）
标准封装

B.1 概述

GPU与通信芯粒合封，也可以通过标准封装形式完成。在维持芯粒间高速互连速率要求的同时，标准封装有更成熟的封装工艺技术，更好的封装成本和良率。

GPU与通信芯粒的合封方式示意图如图B.1所示。一颗GPU + 多颗通信芯粒采用标准封装，通过MCM形式合封在一起。通信芯粒的一侧为SP PHY，可通过在封装基板上的互连线，完成通信芯粒与GPU的互联。通信芯粒另一侧为高速Serdes PHY，用于与外部GPU或其他芯片互联。

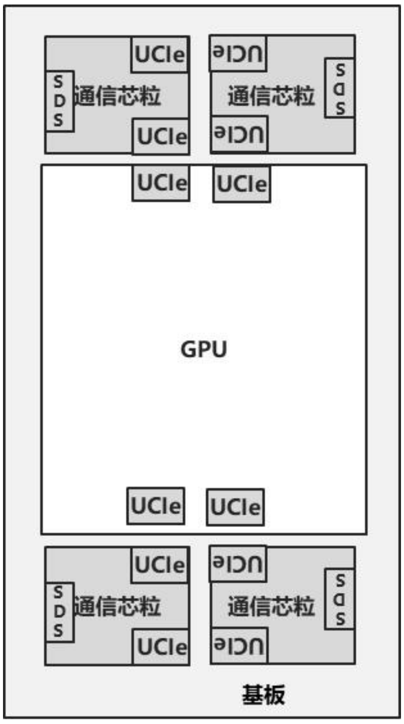


图 B.1 SP 合封示意图方式

B.2 通信芯粒封装

B.2.1 通信芯粒Floorplan

标准封装芯片Floorplan，使用多个X16位单元进行die间交互。

B.2.2 通信芯粒信息

标准封装将提供Die的chip size，Chip thickness，bump pitch UBM Size等信息

B.2.3　通信芯粒POD

Wafer提供KGD与GPU进行合封，切割道宽度等。

B.2.4　通信芯粒Bump规格

Wafer可提供Bumped Wafer或Wafer without Bumping。如需包含Bumping部分，芯粒Solder Bump Cell规格如表B.1和图B.2：

表 B.1　芯粒 SP 封装 Solder Bumping 规格建议

Design Guidelines		Solder Bump Specification
1	Bump Pitch	150-160um
2	Bump Space	50-60um
3	Bump Size	100um
4	Bump Height	75um
5	Bump Material	Ni/SnAg1.8
6	Passivation Opening	50-60um
7	Al Pad Size	≥84um
8	PM Opening	30-40um
9	UBM Size	80um
10	SRO Size	80um

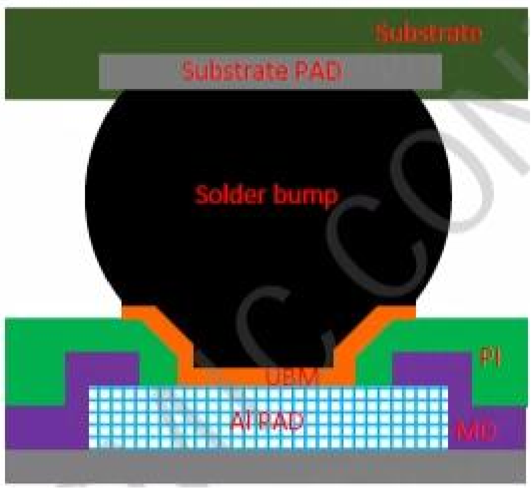


图 B.2

B.2.5　通信芯粒Bump信号

通信芯粒Bump信号见表B.2：

表 B.2 互连 bump list

信号名	数量	描述
主数据信号		
TXDATA [15:0]	16	传输数据信号
TXVLD	1	传输数据有效;启用相应模块的时钟
TXTRK	1	传输跟踪信号
TXCKP	1	传输时钟相位 1
TXCKN	1	传输时钟相位 2
RXDATA [15:0]	16	接收数据信号
RXVLD	1	接收数据有效;启用相应模块的时钟
RXTRK	1	接收跟踪信号
RXCKP	1	接收时钟相位 1
RXCKN	1	接收时钟相位 2
边带信号		
TXDATASB	1	边带传输数据信号
RXDATASB	1	边带接收数据信号
TXCKSB	1	边带传输时钟信号
RXCKSB	1	边带接收时钟信号

B.2.6 通信芯粒集成架构

通信芯粒可适配如下FloorPlan的MCM标准封装集成方案，相关结构经翘曲与应力仿真，可满足组装与可靠性要求。

B.3 主芯片封装

B.3.1 主芯片 Floorplan

标准封装芯片Floorplan，需使用多个X16单元进行die间交互。定义PHY的物理位置。

B.3.2 芯片集成架构

主芯片可适配如下Floor Plan的MCM标准封装集成方案，相关结构经翘曲与应力仿真，可满足组装与可靠性要求。

参 考 文 献