Hermes项目

车规级芯粒互联接口标准

Road viachle - Advanced Cost-driven Chiplet Interface 1.0

目 次

[1 范围 2](#_Toc127911844)

[2 总体概要 2](#_Toc127911845)

[3 协议层 Protocol Layer 2](#_Toc127911846)

[4 链路层 Link Layer 11](#_Toc127911847)

[5 链路训练状态机LTSM（Link Training State Machine） 18](#_Toc127911848)

[6 数字物理层 DPL（Digital Physical Layer） 20](#_Toc127911849)

[7 电气物理层（Electrical Physical Layer） 28](#_Toc127911850)

[8 边带传输 Sideband (Optional) 32](#_Toc127911851)

[9 接口定义 33](#_Toc127911852)

[10 调试追踪 Debug&Trace 36](#_Toc127911853)

[11 配置参数 Configuration and Parameters 38](#_Toc127911854)

[12 封装 Package 39](#_Toc127911855)

Hermes车规级芯粒互联接口标准（ACC\_RV 1.0）

1. 范围

本标准给出了芯粒互联场景下的协议层、链路层和物理层，并规定了各层的消息报格式、工作模式、传输方式等，以及调试追踪方法、接口配置参数、封装方案等。

本标准适用于各种芯粒在32Gbps-128Gbps速率互联互通场景下的接口通信。

1. 总体概要

本标准自顶向下，划分为协议层、链路层和物理层。如图4-1所示：

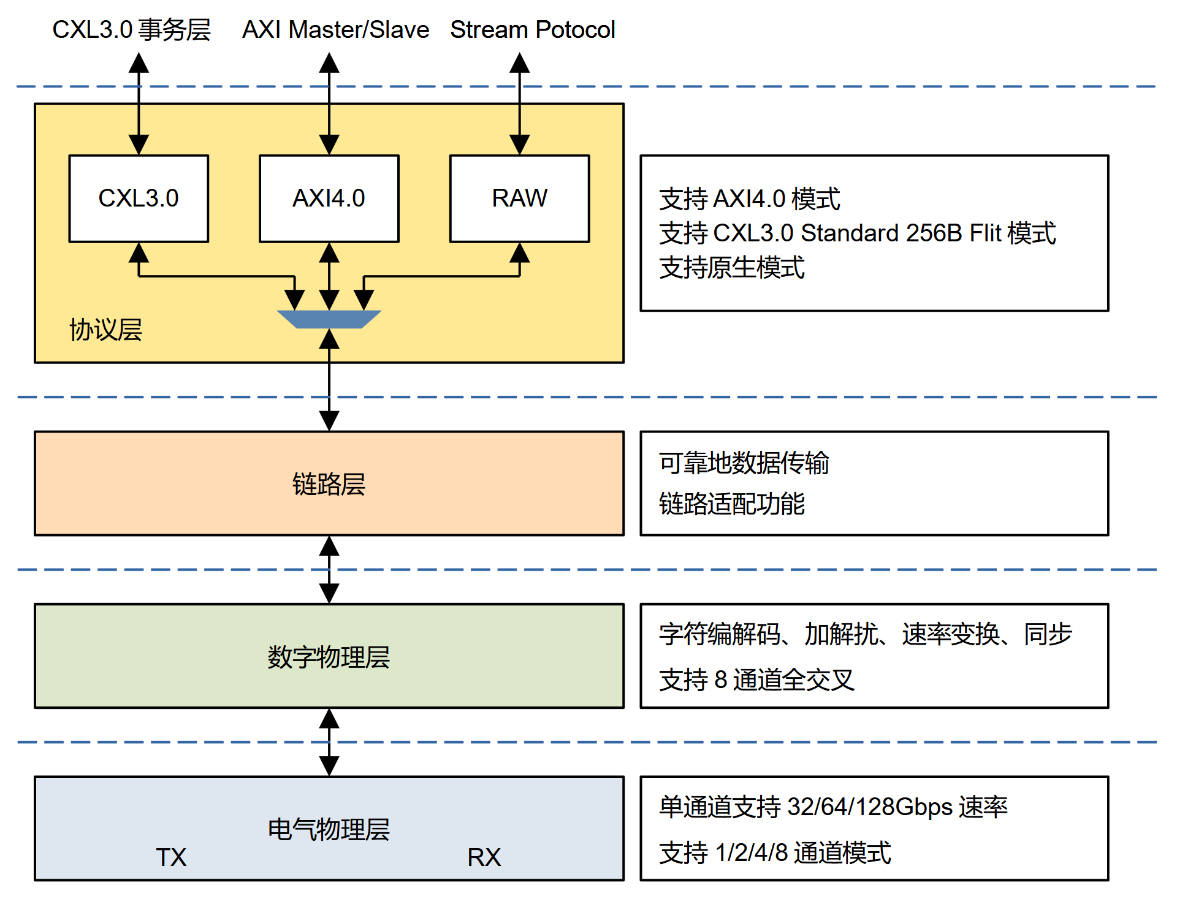


图4-1 协议总览

为了能够适用各类业务Die的Chiplet对接通信，只要遵循同链路层的接口（PLI）协议设计，便可以向上，灵活的对接多种系统协议。本标准默认以常见的高速总线系统AMBA AXI4为例，描述了协议层的具体实现。

链路层提供了快速的建链流程，同时支持自动检测和出错重传，以提供无差错传输。

物理层包含数字物理层和电气物理层。其中电气物理层采用超短距SerDes技术，满足高性能Die的高带宽、低延迟传输。而数字物理层承担SerDes传输必须的加解扰、编解码、数据对齐、链路交叉、极性反转等功能。

1. 协议层 Protocol Layer

本标准支持AMBA AXI4.0协议和CXL3.0协议定义的Standard 256B Flit模式，此外还支持原生模式，即可将任意协议定义的数据流按照用户自定义的规则放置到本标准定义的协议层包中，协议层包做为载体将数据流传递到对端协议层。将其划分为如下3种模式：

* 兼容AXI4.0模式；
* 兼容CXL3.0 Standard 256B Flit模式；
* 原生模式。

设计人员可根据需求，在设计系统及上电初始化时配置所需的模式。

5.1、5.2、5.3章节详细介绍兼容AXI4.0模式、兼容CXL3.0 Standard 256B Flit模式和原生模式下的协议层包格式，5.4章节详细介绍协议层/链路层互联接口。

* 1. 兼容AXI4.0模式

兼容AXI4.0模式是可选的，该模式支持AXI4.0协议定义的传输方式，AXI4.0总线位宽支持512bit。

* + 1. **协议层包**

兼容AXI4.0模式协议层包格式定义如下图所示，其中STP字段预留2个字节，CRC字段预留8个字节，END字段预留6个字节由链路层填充，包长为128字节整数倍，最大长度是640字节，协议层负责将AXI4.0协议信号映射到包头（Head）和负载区域（Payload），映射方法参见5.1.2章节和5.1.3章节。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0 | STP | | Rev | | | | | | Head | | | | | | | | Payload | | | | | | | | | | | | | | | |
| 32 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 96 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 128 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 160 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 192 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 224 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 256 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 288 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 320 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 352 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 384 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 416 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 448 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 480 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 512 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 544 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 576 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 608 | Payload | | | | | | | | | | | | | | | | Rev | | CRC | | | | | | | | END | | | | | |

表5-1 兼容AXI4.0模式下协议层包格式

* + 1. **包头定义**

包头用于标识负载区域，包头内各字段位宽和位置如下表所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Ax | T(3'b000) | | | R | C\_0 | | C\_1 | | Rev | | | | | | CN | Rev | | | | | | | | | | | | | | | | |
| W | T(3'b101) | | | Rev | | | | | | | | | TL | | | R | WL | Rev | | | ST | Rev | | | | | | | | | | |
| R | T(3'b110) | | | R | RL | | | | | | | | TL | | | Rev | | | | | | | | | | | | | | | | |
| Bit | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| Ax | Rev | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | Rev | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R | Rev | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

表5-2 AW/AR/B/W/R通道协议层包包头内各字段位宽和位置

包头内各字段含义如下表所示：

|  |  |
| --- | --- |
| **字段命名** | **字段功能描述** |
| T[2:0] | 用于标识协议层包类型，具体有如下7种，其余值保留：  3'b000：AW/AR/B通道  3'b001：中断拉高  3'b010：中断拉低  3'b011：清中断  3'b100：PRBS序列  3'b101：W通道  3'b110：R通道 |
| C\_x[1:0] | AW/AR/B通道2个命令识别码，优先使用C\_0，其次C\_1：  2'b00：AW通道  2'b01：AR通道  2'b10：B通道 |
| CN[0:0] | AW/AR/B通道命令个数：  1'b0：1个命令  1'b1：2个命令 |
| TL[2:0] | 当前协议层包包含的W/R通道传输的个数：  3'b000：1个传输  3'b001：2个传输  3'b010：3个传输  3'b011：4个传输  3'b100：5个传输  3'b101：6个传输  3'b110：7个传输  3'b111：8个传输 |
| WL[0:0] | 当前协议层包的W通道的最后一个传输是WLAST |
| ST[0:0] | 当前协议层包的W通道的WSTRB信号压缩类型：  1'b0：所有传输都带WSTRB信号  1'b1：只有头尾两笔传输带WSTRB信号，中间位置自动补成全1 |

表5-3 包头内各字段含义

* + 1. **组包规则**

5.1.3.1 AW/AR/B通道组包规则

包头的类型字段为3'b000表示当前协议层包是AW/AR/B通道组成的协议层包，支持最多放置2个AW/AR/B通道命令到1个协议层包，优先放置到A\_0，其次A\_1，组包格式如下图所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0 | STP | | Rev | | | | | | Head | | | | | | | | A\_0[63:0] | | | | | | | | | | | | | | | |
| 32 | A\_1[63:0] | | | | | | | | | | | | | | | | Rev | | | | | | | | | | | | | | | |
| 64 | Rev | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 96 | Rev | | | | | | | | | | | | | | | | | | CRC | | | | | | | | END | | | | | |

表5-4 兼容AXI4.0模式下AW/AR/B通道协议层包格式定义

A\_x内各字段位宽和位置如下图所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Ax | AL | R | ABST | | ASIZE | | | APROT | | | ALEN | | | | | | ACACHE | | | | AREGION | | | | AQOS | | | | ADDR[0:3] | | | |
| B | BRSP | | Rev | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| Ax | ADDR[4:35] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B | Rev | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| Ax | ADDR[36:63] | | | | | | | | | | | | | | | | | | | | | | | | | | | | AID[0:3] | | | |
| B | Rev | | | | | | | | | | | | | | | | | | | | | | | | | | | | BID[0:3] | | | |
| Bit | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| Ax | AID[4:7] | | | | AUSER | | | | | | | | | | | | | | | | Rev | | | | | | | | | | | |
| B | BID[4:7] | | | | BUSER | | | | | | | | | | | | | | | | Rev | | | | | | | | | | | |

表5-5 兼容AXI4.0模式下协议层包A\_x内各字段位宽和位置

A\_x内各字段含义如下表所示：

|  |  |
| --- | --- |
| **字段命名** | **字段功能描述** |
| ADDR[63:0] | AW/AR通道ADDR信号 |
| ALEN[5:0] | AW/AR通道LEN信号 |
| AID[7:0] | AW/AR通道ID信号 |
| ASIZE[2:0] | AW/AR通道SIZE信号 |
| AQOS[3:0] | AW/AR通道QOS信号 |
| AUSER[15:0] | AW/AR通道USER信号 |
| AL[0:0] | AW/AR通道LOCK信号 |
| ABST[1:0] | AW/AR通道BURST信号 |
| APROT[2:0] | AW/AR通道PROT信号 |
| ACACHE[3:0] | AW/AR通道CACHE信号 |
| AREGION[3:0] | AW/AR通道REGION信号 |
| BID[7:0] | B通道ID信号 |
| BRSP[1:0] | B通道RESP信号 |
| BUSER[15:0] | B通道USER信号 |

表5-6 A\_x内各字段含义

5.1.3.2 W通道组包规则

AXI4.0协议W通道主要传递的信号是WSTRB/WDATA，为了节省链路带宽，本标准提出一种基于可变长有效数据动态排列的数据压缩方法，将W通道的带宽利用率提高到实际带宽的70%至95%。

链路层的重传和流控是以协议层包为最小单位进行控制的，若协议层包过大会影响链路控制精细度，且会占用更多的重传缓存，影响架构设计；若协议层包过小会导致开销和有效数据比例变大，影响链路传输效率。权衡之后确定W通道协议层包最多可以装8个传输，最少可以装1个传输，遇到WLAST则停止组包。

W通道拼接数据时，会根据情况附带WSTRB，若除首尾两个传输的WSTRB外，中间传输的WSTRB为全1，则省略中间传输的WSTRB，将包头的ST字段置为1；若中间传输的WSTRB不为全1，则全部传输都带WSTRB，将包头的ST字段置为0。

包头的类型字段为3'b101表示当前协议层包是W通道组成的协议层包，根据WSTRB指示的有效数据长度，将512bit的WDATA压缩之后放入协议层包，压缩方法是先将64bit的WSTRB视作8个字节，每个字节缩位或之后得到1bit，共计8bit。每1个bit对应64bit的WDATA，将缩位或之后，值为1所指示的WDATA放入协议层包。

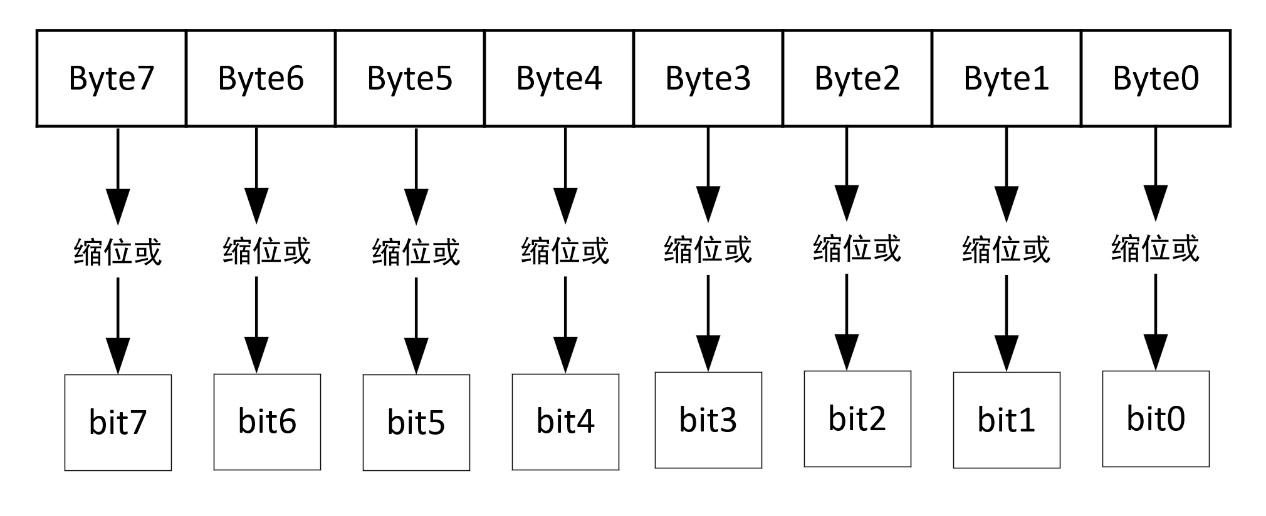


图5-7 WSTRB缩位或

注意：只支持缩位或之后值为1的区域是连续的情况，比如8'b00111000/8'b11111100，不允许出现8'b0001000，组包格式举例如下图所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 | 104 | 112 | 120 |
| 0 | PREFIX | | S\_0 | WD\_0（可变长） | | | | | | | | WD\_1（64字节） | | | | |
| 128 | WD\_1（64字节） | | | WD\_2（64字节） | | | | | | | | WD\_3（64字节） | | | | |
| 256 | WD\_3（64字节） | | | WD\_4（64字节） | | | | | | | | WD\_5（64字节） | | | | |
| 384 | WD\_5（64字节） | | | WD\_6（64字节） | | | | | | | | S\_7 | WD\_7（可变长） | | | |
| 512 | WD\_7（可变长） | | | | Rev | | | | | | | | | | POSTFIX | |

表5-8 W通道协议层包举例（头尾带S标记，8个传输中第1个和第8个带S\_0/S\_7，用于标记WD\_0/WD\_7长度和位置）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 | 104 | 112 | 120 |
| 0 | PREFIX | | S\_0 | WD\_0（可变长） | | | | | WD\_1（64字节） | | | | | | | |
| 128 | WD\_2（64字节） | | | | | | | | WD\_3（64字节） | | | | | | | |
| 256 | S\_4 | WD\_4（可变长） | | | | | | | Rev | | | | | | POSTFIX | |

表5-9 W通道协议层包举例（头尾带S标记，5个传输中第1个和第5个带S\_0/S\_4，用于标记WD\_0/WD\_4长度和位置）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 | 104 | 112 | 120 |
| 0 | PREFIX | | S\_0 | WD\_0（可变长） | | | | | S\_1 | WD\_1（可变长） | | | | | | |
| 128 | S\_2 | WD\_2（可变长） | | | | | S\_3 | WD\_3（可变长） | | | | | | | | S\_4 |
| 256 | WD\_4（可变长） | | | | | | S\_5 | S\_6 | Rev | | | | | | POSTFIX | |

表5-10 W通道协议层包举例（全带S标记，7个传输，其中任意S\_x可以是全0，表示该笔传输无有效数据）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 | 104 | 112 | 120 |
| 0 | PREFIX | | S\_0 | WD\_0（可变长） | | | | | S\_1 | WD\_1（可变长） | | | | | | |
| 128 | S\_2 | WD\_2（可变长） | | | | | S\_3 | WD\_3（可变长） | | | | | | | | S\_4 |
| 256 | WD\_4（可变长） | | | | | | | S\_5 | WD\_5（可变长） | | | | | | | |
| 384 | S\_6 | WD\_6（可变长） | | | | | S\_7 | WD\_7（可变长） | | | | | | | | Rev |
| 512 | Rev | | | | | | | | | | | | | | POSTFIX | |

表5-11 W通道协议层包举例（全带S标记，8个传输，当最后1拍无法预留CRC/END时，需要额外多发一拍）

WA内各字段位宽和位置如下图所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|  | WUSER\_0 | | | | | | | | | | | | | | | | WUSER\_1 | | | | | | | | | | | | | | | |
| **Bit** | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
|  | WUSER\_2 | | | | | | | | | | | | | | | | WUSER\_3 | | | | | | | | | | | | | | | |
| **Bit** | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
|  | WUSER\_4 | | | | | | | | | | | | | | | | WUSER\_5 | | | | | | | | | | | | | | | |
| **Bit** | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
|  | WUSER\_6 | | | | | | | | | | | | | | | | WUSER\_7 | | | | | | | | | | | | | | | |

表5-12 W通道协议层包WA内各字段位宽和位置

WA内各字段含义如下表所示：

|  |  |
| --- | --- |
| 字段命名 | 字段功能描述 |
| WUSER\_x[15:0] | W通道USER信号，WUSER\_0对应第1个传输，WUSER\_7对应第8个传输 |

表5-13 WA中各字段含义

5.1.3.3 R通道组包规则

包头的类型字段为3'b110表示当前协议层包是R通道组成的协议层包，组包时将RDATA按顺序拼接在一起即可，不需要考虑数据压缩问题，组包格式举例如下图所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 | 104 | 112 | 120 |
|  | PREFIX | | RA | | | | RD\_0 | | | | | | | | RD\_1 | |
|  | RD\_1 | | | | | | RD\_2 | | | | | | | | RD\_3 | |
|  | RD\_3 | | | | | | RD\_4 | | | | | | | | RD\_5 | |
|  | RD\_5 | | | | | | RD\_6 | | | | | | | | RD\_7 | |
|  | RD\_7 | | | | | | Rev | | | | | | | | POSTFIX | |

表5-14 R通道协议层包举例（8个传输）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 | 104 | 112 | 120 |
|  | PREFIX | | RA | | | | RD\_0 | | | | | | | | RD\_1 | |
|  | RD\_1 | | | | | | RD\_2 | | | | | | | | POSTFIX | |

表5-15 R通道协议层包举例（3个传输）

RA内各字段位宽和位置如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|  | RUSER\_0 | | | | | | | | | | | | | | | | RUSER\_1 | | | | | | | | | | | | | | | |
| **Bit** | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
|  | RUSER\_2 | | | | | | | | | | | | | | | | RUSER\_3 | | | | | | | | | | | | | | | |
| **Bit** | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
|  | RUSER\_4 | | | | | | | | | | | | | | | | RUSER\_5 | | | | | | | | | | | | | | | |
| **Bit** | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
|  | RUSER\_6 | | | | | | | | | | | | | | | | RUSER\_7 | | | | | | | | | | | | | | | |
| **Bit** | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
|  | RID\_0 | | | | | | | | RID\_1 | | | | | | | | RID\_2 | | | | | | | | RID\_3 | | | | | | | |
| **Bit** | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
|  | RID\_4 | | | | | | | | RID\_5 | | | | | | | | RID\_6 | | | | | | | | RID\_7 | | | | | | | |
| **Bit** | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
|  | RP\_0 | | RP\_1 | | RP\_2 | | RP\_3 | | RP\_4 | | RP\_5 | | RP\_6 | | RP\_7 | | RL | | | | | | | | Rev | | | | | | | |
| **Bit** | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |
|  | Rev | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

表5-16 R通道协议层包RA内各字段位宽和位置

RA内各字段含义如下表所示：

|  |  |
| --- | --- |
| **字段命名** | **字段功能描述** |
| RUSER\_x[15:0] | R通道USER信号，RUSER\_0对应第1个传输，RUSER\_7对应第8个传输 |
| RID\_x[7:0] | R通道ID信号，RID\_0对应第1个传输，RID\_7对应第8个传输 |
| RL[7:0] | R通道RLAST信号，RL[0]对应第1个传输，RL[7]对应第8个传输 |
| RP\_x[1:0] | R通道RRESP信号，RP\_0对应第1个传输，RP\_7对应第8个传输 |

表5-17 RA内各字段含义

* 1. 兼容CXL3.0 Standard 256B Flit模式

兼容CXL3.0 Standard 256B Flit模式是可选的。

* + 1. 协议层包

兼容CXL3.0 Standard 256B Flit模式协议层包格式定义如下图所示，其中STP字段预留2个字节，CRC字段预留8个字节，END字段预留6个字节由链路层填充，包长固定为256字节，协议层负责将CXL3.0协议Standard 256B Flit模式下定义的FlitData区域映射到负载区域。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0 | STP | | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 32 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 96 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 128 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 160 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 192 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 224 | Payload | | | | | | | | | | | | | | | | | | CRC | | | | | | | | END | | | | | |

表5-18 兼容CXL3.0 Standard 256B Flit模式下协议层包格式

* 1. 兼容CXL3.0 Standard 256B Flit模式

原生模式是可选的，当用户不需要承载AXI4.0协议或CXL3.0 Standard 256B Flit模式数据流时，可以选用原生模式，只要组包符合5.3.1章节描述的规则，仍然可以使用链路层重传功能。

* + 1. 协议层包定义及组包规则

原生模式协议层包格式定义如下图所示，其中STP字段预留2个字节，CRC字段预留8个字节，END字段预留6个字节由链路层填充，包长为128字节整数倍，最大长度是640字节，协议层负责将任意协议信号映射到负载区域，映射方法由用户自定义。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0 | STP | | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 32 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 96 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 128 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 160 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 192 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 224 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 256 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 288 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 320 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 352 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 384 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 416 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 448 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 480 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 512 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 544 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 576 | Payload | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 608 | Payload | | | | | | | | | | | | | | | | | | CRC | | | | | | | | END | | | | | |

表5-19 原生模式下协议层包格式

* 1. 协议层/链路层互联接口

协议层/链路层接口如下表所示：

|  |  |
| --- | --- |
| **信号命名** | **信号功能描述** |
| prot2link\_valid[0:0] | 发送侧，协议层向链路层指示prot2link\_data/prot2link\_tail上的数据有效 |
| link2prot\_rdy[0:0] | 发送侧，链路层向协议层指示可以接收数据，使用valid-rdy握手机制传递数据 |
| prot2link\_data[1023:0] | 发送侧，协议层向链路层传递的协议层包的其中1拍数据 |
| prot2link\_tail[0:0] | 发送侧，协议层向链路层传递的协议层包的尾标记 |
| link2prot\_valid[0:0] | 接收侧，链路层向协议层指示link2prot\_data/link2prot\_tail上的数据有效 |
| prot2link\_rdy[0:0] | 接收侧，协议层向链路层指示可以接收数据，使用valid-rdy握手机制传递数据 |
| link2prot\_data[1023:0] | 接收侧，链路层向协议层传递的协议层包的其中1拍数据 |
| link2prot\_tail[0:0] | 接收侧，链路层向协议层传递的协议层包的尾标记 |

表5-20 协议层/链路层互联接口

协议层到链路层接口时序举例如下图所示：

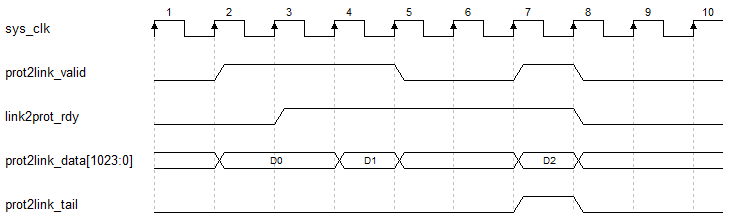


图5-21 协议层到链路层接口时序

链路层到协议层接口时序举例如下图所示：

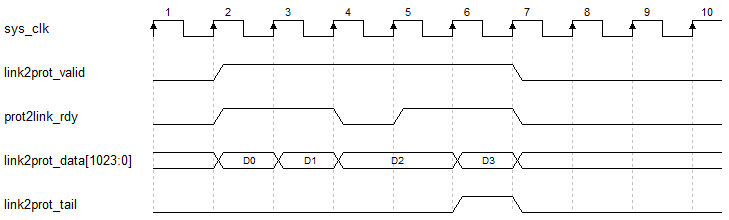


图5-22 链路层到协议层接口时序

1. 链路层 Link Layer

协议层和链路层以协议层包为最小单元传输数据。链路层负责将协议层包可靠地传输到对端，利用ACK/NAK机制保证数据传输的可靠性。为此需要先对协议层包进行编号，然后对编号和负载区域做CRC校验位的计算，并将CRC计算结果放置到协议层包的CRC字段，由对端链路层对接收到的协议层包做CRC校验，决定是否需要重传。

链路层发送端支持如下功能：

● 填充协议层包ID编号；

● 填充协议层包CRC校验位；

● 发送协议层包，并利用接收端反馈的ACK/NAK+ID编号指令释放/重传协议层包；

● 发送ACK/NAK包。

链路层接收端支持如下功能：

● 对链路层包进行CRC校验；

● 接收ACK/NAK包并控制TX重传缓存待重传的数据；

● 对协议层包进行CRC校验，并将校验结果整合到ACK/NAK包请求发送端发送给远端；

● 利用NAK包流控。

* 1. ID编号方法

本标准规定协议层包中STP字段的bit[15:8]用于放置协议层包ID，ID编号0~255循环递增，255计满翻转到0，复位初始值为0，STP字段内ID编号位置如下图所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|  | STP | | | | | | | | ID | | | | | | | |

表6-1 STP字段内ID编号位置

* 1. CRC计算及校验方法

如下图所示，将协议层包每16字节划分为1个独立的CRC校验区域，共划分为8个独立的CRC校验区域。每个CRC校验区域对应1个CRC校验单元，每个CRC校验单元输入16字节数据，加上1字节的初始值或同一个协议层包上1拍该位置计算得到的CRC校验值，计算得到新的1字节CRC校验值，注意STP（不含ID）/CRC/END字段用0占位送入CRC计算单元。在协议层的包尾处，将总共8个字节的CRC校验值填充到CRC字段。

协议层包的CRC产生多项式为1+x^5+x^7+x^8，verilog代码在附录中提供。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0 | CRC\_0 | | | | | | | | | | | | | | | | CRC\_1 | | | | | | | | | | | | | | | |
| 32 | CRC\_2 | | | | | | | | | | | | | | | | CRC\_3 | | | | | | | | | | | | | | | |
| 64 | CRC\_4 | | | | | | | | | | | | | | | | CRC\_5 | | | | | | | | | | | | | | | |
| 96 | CRC\_6 | | | | | | | | | | | | | | | | CRC\_7 | | | | | | | | | | | | | | | |
| 128 | CRC\_0 | | | | | | | | | | | | | | | | CRC\_1 | | | | | | | | | | | | | | | |
| 160 | CRC\_2 | | | | | | | | | | | | | | | | CRC\_3 | | | | | | | | | | | | | | | |
| 192 | CRC\_4 | | | | | | | | | | | | | | | | CRC\_5 | | | | | | | | | | | | | | | |
| 224 | CRC\_6 | | | | | | | | | | | | | | | | CRC\_7 | | | | | | | | | | | | | | | |
| 256 | CRC\_0 | | | | | | | | | | | | | | | | CRC\_1 | | | | | | | | | | | | | | | |
| 288 | CRC\_2 | | | | | | | | | | | | | | | | CRC\_3 | | | | | | | | | | | | | | | |
| 320 | CRC\_4 | | | | | | | | | | | | | | | | CRC\_5 | | | | | | | | | | | | | | | |
| 352 | CRC\_6 | | | | | | | | | | | | | | | | CRC\_7 | | | | | | | | | | | | | | | |
| 384 | CRC\_0 | | | | | | | | | | | | | | | | CRC\_1 | | | | | | | | | | | | | | | |
| 416 | CRC\_2 | | | | | | | | | | | | | | | | CRC\_3 | | | | | | | | | | | | | | | |
| 448 | CRC\_4 | | | | | | | | | | | | | | | | CRC\_5 | | | | | | | | | | | | | | | |
| 480 | CRC\_6 | | | | | | | | | | | | | | | | CRC\_7 | | | | | | | | | | | | | | | |
| 512 | CRC\_0 | | | | | | | | | | | | | | | | CRC\_1 | | | | | | | | | | | | | | | |
| 544 | CRC\_2 | | | | | | | | | | | | | | | | CRC\_3 | | | | | | | | | | | | | | | |
| 576 | CRC\_4 | | | | | | | | | | | | | | | | CRC\_5 | | | | | | | | | | | | | | | |
| 608 | CRC\_6 | | | | | | | | | | | | | | | | CRC\_7 | | | | | | | | | | | | | | | |

表6-2 CRC校验区域划分

* 1. 链路层包

链路层包使用SDP字符标记头，END字符标记尾。链路层包和协议层包共享链路带宽，优先调度链路层包，其次调度协议层包，链路层包定义如下图所示：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Byte** | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 | 104 | 112 | 120 |
|  | SDP | DLP | END | PAD | PAD | | PAD | | PAD | | PAD | | PAD | | PAD | |

表6-3 链路层包定义

* + 1. ACK/NAK包定义

ACK/NAK包格式定义如下图所示，通过bit[15]区分ACK/NAK。若接收端接收到的协议层包通过CRC校验且ID符合预期，则将协议层包的ID填充到ACK/NAK包的ID字段。

链路层包的CRC产生多项式为1+x^2+x^15+x^16，输入为6字节（不含CRC字段），输出2字节CRC校验值填充到CRC字段，verilog代码在附录中提供。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| ACK | 8'ha5 | | | | | | | | Rev | | | | | | | 0 | ID | | | | | | | | Rev | | | | | | | |
| NAK | 8'ha5 | | | | | | | | Rev | | | | | | | 1 | ID | | | | | | | | Rev | | | | | | | |
| **Bit** | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| ACK | Rev | | | | | | | | | | | | | | | | CRC | | | | | | | | | | | | | | | |
| NAK | Rev | | | | | | | | | | | | | | | | CRC | | | | | | | | | | | | | | | |

表6-4 链路层ACK/NAK包格式

* 1. 重传机制

协议层包需要完整保存在链路层发送端的重传缓存中，当发送端收到ACK/NAK握手之后，则允许从重传缓存中释放已经握手完成的协议层包。设计电路时，建议将重传缓存的大小设计为大于2倍链路发送端链路层到接收端链路层缓存的数据拍数，从而允许接收端能适当放大回复ACK的间隔，提高链路传输效率。

* + 1. 重传规则

链路层发送端将所有协议层包缓存在重传缓存中，只在收到有效的ACK/NAK包之后，释放ACK/NAK包携带的ID之前（含）的协议层包，若收到NAK包，则重传NAK包携带的ID编号加1的协议层包；若长时间未收到有效的ACK/NAK包，则上报超时告警，并从头开始发送所有协议层包。注意上述NAK重传和超时重传需要保证当前正在发送的协议层包发送完毕之后，再去发送需要重传的协议层包。

链路层接收端对当前协议层包的判定流程如下图所示，先计算当前协议层包的CRC并和CRC校验位比较，注意计算区域为协议层包的ID/包头/负载，STP/CRC/END字段用0占位，若CRC校验正确，再去判断协议层包ID和期望ID是否相等，若相等，则上送协议层包给协议层，将期望ID加1，将NAK标记置为0，将ACK包放入发送队列并在适当间隔后交由链路层发送端发送；若CRC校验错误或者协议层包ID和期望ID不相等，则上报告警，然后判断NAK标记是否为1，若不为1，则触发重传，发送NAK包，将NAK标记置为1，之后丢弃当前协议层包，释放该协议层包占用的缓存；若NAK标记为1，则只丢弃当前协议层包，释放该协议层包占用的缓存。

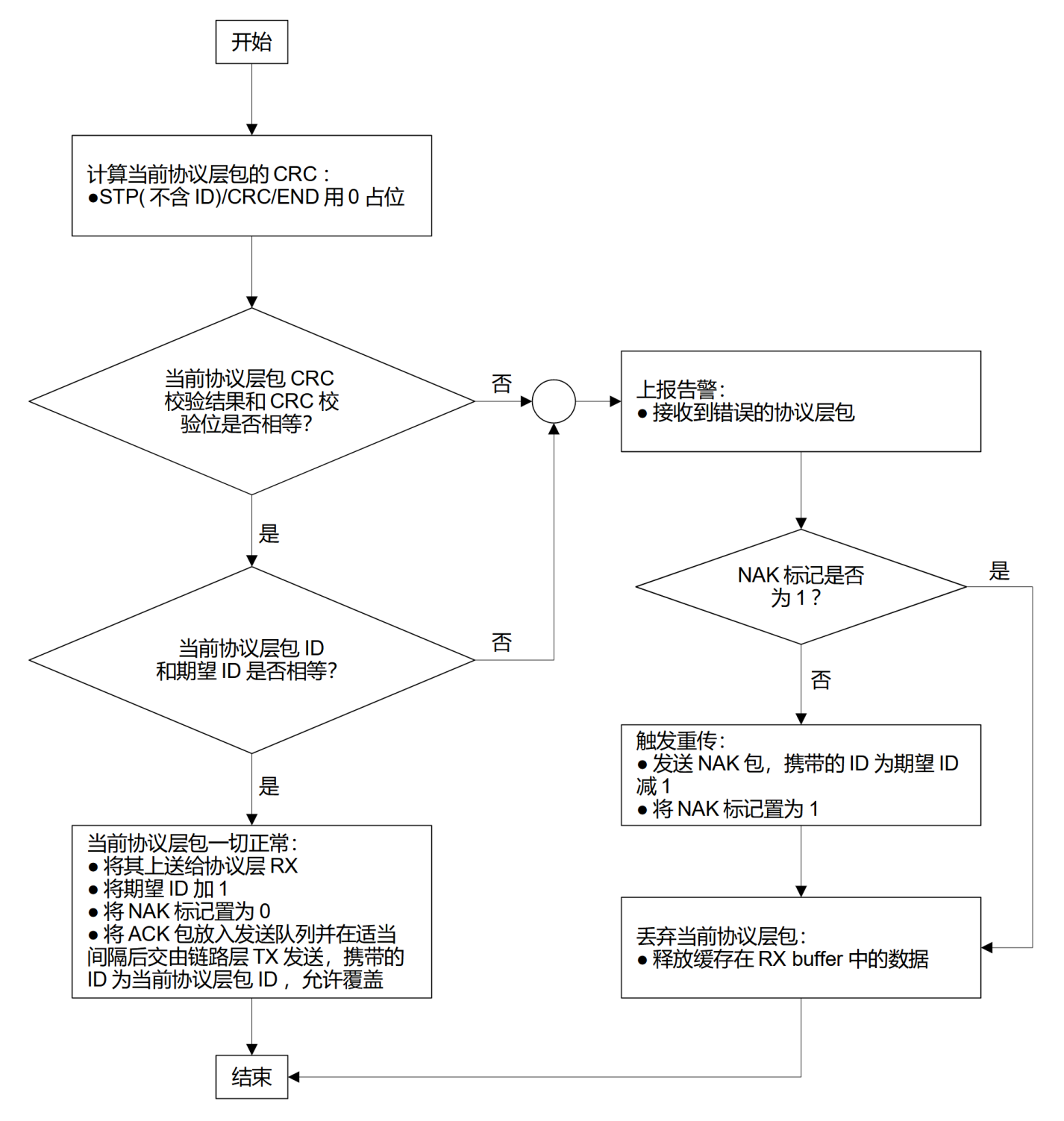


图6-5 链路层接收端触发重传判定流程

* + 1. 利用NAK包实现流控

当链路层接收端发现协议层接收端无法接收新的协议层包，并且自身缓存已满时，允许通过发送NAK包的方式，让远端以协议层包为单位重传，直到协议层接收端畅通。

* 1. 链路适配功能

链路适配功能用于将链路层组装好的协议层包/链路层包标记为数据/控制字符，并同COM字符/IDL字符三选一调度，调度1次得到8个数据/控制字符，再通过通道分发将8个字符分发到数字物理层的1/2/4/8个通道，接收端基于同样的原理将字符对齐，并还原为发送端发出时的排序之后，上送给链路层。

发送端支持如下功能：

● 将协议层/链路层包标记为数据字符并使用STP/SDP/END/PAD字符标记协议层/链路层包的头尾；

● 周期性发送COM字符；

● 空闲状态下发送IDL字符；

● 上述3种类别字符的调度，优先发送COM字符，其次发送协议层/链路层包编码之后的字符，再次发送IDL字符；

● 1/2/4/8通道分发。

接收端支持如下功能：

● 利用COM字符实现2/4/8通道去偏移；

● 1/2/4通道合并；

● 过滤控制字符，并将STP/SDP/END字符转换为协议层包/链路层包头尾信息。

* + 1. 协议层/链路层包头尾标记方法

协议层包头尾分别是STP/END字符，需要将其填充为STP/END字符的值，并将其所在的128bit码块标记为控制字符，其余部分标记为数据字符。规定STP字符只占用通道0的bit[7:0]共8bit，通道0的bit[127:16]共112bit留给协议层，通道0的bit[15:8]共8bit留给链路层使用，规定END字符只占用通道7的bit[127:80]共48bit，通道7的bit[79:0]共80bit留给链路层使用。

链路层包头尾分别为SDP/END字符，需要将其填充为SDP/END字符的值，并将其所在的128bit码块标记为控制字符。规定SDP字符只占用通道0的bit[63:0]共64bit，通道0的bit[127:64]共64bit留给链路层使用，规定END字符只占用通道1的bit[63:0]共64bit，通道1的bit[127:64]填充PAD字符的bit[127:64]。

* + 1. 控制字符

控制字符是指STP/SDP/END/PAD/COM/IDL字符，用于标记链路传输的数据流。

6.5.2.1 STP/SDP/END/PAD字符

STP字符用于标记协议层包开始，位宽为16bit；SDP字符用于标记链路层包开始，位宽为64bit；END字符用于标记协议层包/链路层包结束，位宽为64bit，其中协议层包使用END字符的bit[63:16]；PAD字符用于将链路层包补足到8个字符的数据量。

STP字符的值为：{2{8’hFB}}

SDP字符的值为：{8{8’h5C}}

END字符的值为：{8{8’hFD}}

PAD字符的值为：{16{8’h0}}

6.5.2.2 COM字符

COM字符位宽为128bit，发送周期可配，1/2/4/8通道模式下分别发送1/2/4/8个COM字符，使用IDL字符补满8个字符后，调度发送到链路上，以便每个通道都能获取到1个COM字符。

COM字符的值为：{15{8’hBC},8’h7D}

6.5.2.2.1 同步数字物理层加解扰多项式

数字物理层使用COM字符同步加解扰多项式。

6.5.2.2.2 去偏移

链路层接收端工作在2/4/8通道模式时，需要利用COM字符实现去偏移，将COM字符对齐到同一拍。

6.5.2.2.3 多通道合并

链路层接收端工作在2/4/8通道模式时，需要利用COM字符实现多通道合并，将第1个COM字符挪到通道0，后续字符依次跟随挪动，从而还原为链路层发送端发出时的字符排列顺序。

6.5.2.3 IDL字符

IDL字符位宽为128bit，用于填充链路空闲带宽，每调度1次IDL字符得到8个IDL字符，以便每个通道都能获取到1个IDL字符，需要注意IDL字符也需要加解扰，以确保链路在空闲时能维持在正常状态。

IDL字符的值为：{16{8’hDC}}

6.5.2.3.1 填充空闲带宽

IDL字符由链路适配发送端发出，用于填充链路空闲带宽，链路适配接收端将其丢弃。

* + 1. 多通道分发与合并

协议层包/链路层包、COM字符、IDL字符每调度1次获得的数据量都是8个字符，只允许在协议层包/链路层包结束之后，调度COM/IDL字符，从而保证链路上协议层包/链路层包的连续性。获取到8个字符之后，1/2/4通道工作模式需要将其分发8/4/2次给通道0/通道0~1/通道0~3，从而实现向下兼容1/2/4通道，8通道不需要做通道分发，每个通道发送1个字符即可。

通道合并是通道分发的逆过程，需要注意利用COM字符将字符序列还原为发送端发出时的字符的排列顺序。

* + 1. 链路层/物理层互联接口

链路层/物理层接口如下表所示：

|  |  |
| --- | --- |
| 信号命名 | 信号功能描述 |
| link2phy\_valid[0:0] | 发送侧，链路层向物理层指示link2phy\_data/link2phy\_dk上的数据有效 |
| phy2link\_rdy[0:0] | 发送侧，物理层向链路层指示可以接收数据，使用valid-rdy握手机制传递数据 |
| link2phy\_data[1023:0] | 发送侧，链路层向物理层传递的数据流，每个通道128bit，共8个通道 |
| link2phy\_dk[7:0] | 发送侧，链路层向物理层传递的数据流的控制字符/数据字符标记，每个通道1bit，共8个通道  1'b0：控制字符  1'b1：数据字符 |
| phy2link\_valid[0:0] | 接收侧，物理层向链路层指示phy2link\_data/phy2link\_dk上的数据有效 |
| phy2link\_data[1023:0] | 接收侧，物理层向链路层传递的数据流，每个通道128bit，共8个通道 |
| phy2link\_dk[7:0] | 接收侧，物理层向链路层传递的数据流的控制字符/数据字符标记，每个通道1bit，共8个通道  1'b0：控制字符  1'b1：数据字符 |

表6-6 链路层/物理层互联接口

链路适配到物理层接口时序举例如下图所示：

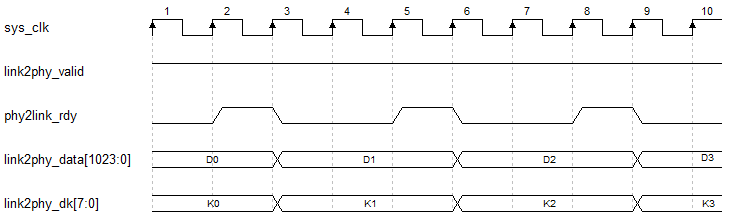


图6-7 链路适配到物理层接口时序

物理层到链路适配接口时序举例如下图所示：

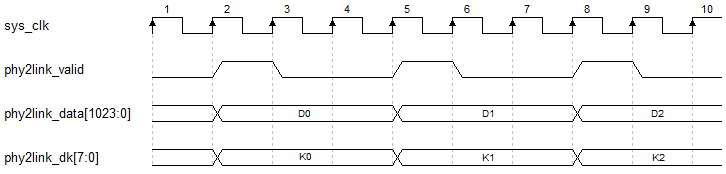


图6-8 物理层到链路适配接口时序

* + 1. 链路数据流举例

链路数据流可以是任意协议层包/链路层包/8个通道的IDL字符/8个通道的COM字符组合，若工作于兼容AXI4.0模式或原生模式，协议层包包长为128字节整数倍，最大长度是640字节，如下图举例中协议层包A是640字节，协议层包B是256字节，协议层包C是128字节，协议层包D是384字节，协议层包E是512字节；若工作于兼容CXL3.0 Standard 256B Flit模式，协议层包包长固定为256字节。1个链路层包占用的带宽和1个包长为128字节的协议层包相等。COM字符发送周期可配，IDL字符用于填充链路空闲带宽。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **通道** | 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | |
| **Byte** | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 | 104 | 112 | 120 |
|  | 协议层包A | | | | | | | | | | | | | | | |
|  | 协议层包A | | | | | | | | | | | | | | | |
|  | 协议层包A | | | | | | | | | | | | | | | |
|  | 协议层包A | | | | | | | | | | | | | | | |
|  | 协议层包A | | | | | | | | | | | | | | | |
|  | 协议层包B | | | | | | | | | | | | | | | |
|  | 协议层包B | | | | | | | | | | | | | | | |
|  | 链路层包A | | | | | | | | | | | | | | | |
|  | 协议层包C | | | | | | | | | | | | | | | |
|  | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | |
|  | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | |
|  | COM | | COM | | COM | | COM | | COM | | COM | | COM | | COM | |
|  | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | |
|  | 协议层包D | | | | | | | | | | | | | | | |
|  | 协议层包D | | | | | | | | | | | | | | | |
|  | 协议层包D | | | | | | | | | | | | | | | |
|  | 协议层包E | | | | | | | | | | | | | | | |
|  | 协议层包E | | | | | | | | | | | | | | | |
|  | 协议层包E | | | | | | | | | | | | | | | |
|  | 协议层包E | | | | | | | | | | | | | | | |
|  | 链路层包B | | | | | | | | | | | | | | | |
|  | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | |
|  | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | |
|  | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | | IDL | |

表6-9 链路数据流举例（8通道模式）

1. 链路训练状态机LTSM（Link Training State Machine）
   1. 概述

LTSM主要用于D2D接口在建链过程中的状态控制，当发起建链后将进行状态的跳转，并在不同的状态下发出不同的控制信息，控制信息包括对链路层与物理层的控制，从而控制D2D接口整个建链与数据传输流程。

LTSM包括3个主要状态：Config、Traning、Normal。在Config状态，近端D2D通过APB配置通道对其进行配置，同时近端D2D通过Sideband信号对远端D2D进行配置，配置内容包括需要使能的通道数、链路层与物理层的通道交叉配置、速率配置等等；Training状态下，近端与远端互相发送与检测NULL码；Normal状态为正常传输数据状态。具体如下：



图7-1 LTSM状态示意图

* 1. 建链流程

D2D通过寄存器配置发起建链操作，本标准中将D2D接口的两端区分为近端和远端，主动发起建链的一端为近端，被动接收建链操作的一端为远端，具体连接关系如图7-2所示，远端与近端的数据交换，通过电气物理层SerDes的差分接口进行。寄存器参数的配置，则通过边带信号进行，并且只能由近端主动发起对远端的读写操作。



图7-2 D2D接口连接示意图

本标准的建链过程如下：

1、D2D一端(近端)通过寄存器配置发起建链后，进入Config状态；

2、近端通过边带信号对远端进行参数配置同步，完成配置后通过配置确定是否进行步骤3，如果使能配置同步检查则进入步骤3，否则进入步骤4，状态机跳转至Training状态；

3、当需要同步的寄存器配置（包括使能对应的通道以及PLL）完成后，则进行配置确认，通过边带信号对远端寄存器进行读取并确认是否配置正确；

4、近端发送端开始向远端发送Null码（见章节7.3），该状态下近端发送Null码的数量可配（默认值为2048个Null码）；

5、远端接收端连续正确接收到足够数量（数量可配，默认值为16）的Null码后，则远端发送端开始向近端接收端发送Null码（发送数量可配，默认值为1024）；

6、当近端接收端连续正确接收到足够数量（数量可 配，默认值为16）的Null码且发送了足够数量（步骤4对应的数量）的Null码后，则表示链路训练完成，结束Training状态，进入Normal状态，可以正常传数据。

流程如下图所示：



图7-3 建链流程

* 1. NULL码型

在Training状态下同时向所有使能的通道发送Null码，具体组成为1COM + 7\*IDL，如下图所示：

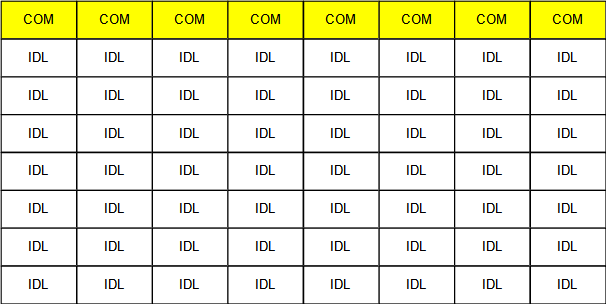


图7-4 Null码示意图

1. 数字物理层 DPL（Digital Physical Layer）
   1. DPL层功能概述

本标准定义了通过DPL层发送和接收物理层传输的逻辑数据传输流。物理层分为数字物理层和电气物理层。数字物理层将链路层与SerDes结构的电气物理层衔接起来，并进行数据通路的编码隔离。



图8-1 PL结构框图

DPL最多接收发送端LINK层的1024bit数据。每一条通道内部结构相同，可以通过链路模式寄存器配置为不同的工作模式，如表所示：

|  |  |
| --- | --- |
| **链路模式寄存器** | **工作模式** |
| 00 | 单通道模式 |
| 01 | 双通道模式 |
| 10 | 四通道模式 |
| 11 | 八通道模式 |

表8-2 寄存器配置模式

物理层最多允许8条通道同时工作，通道的使能由上层控制。

包括以下功能：

* 链路全交叉；
* 通道的极性反转；
* 链路的加解扰和编解码；
* 支持可信度的Block Alignment；
* PRBS/FIX的生成和检测；
* 每条通道数据位宽为128bit。

**性能指标要求：**

|  |  |  |  |
| --- | --- | --- | --- |
| **时钟域** | **低速频率** | **中速频率** | **高速频率** |
| DPL工作时钟 | >=300Mhz | >=600Mhz | >=1200Mhz |
| EPL读写时钟 | 250Mhz | 500Mhz | 1000Mhz |
| 每通道传输速率 | 32GHz | 64Ghz | 128Ghz |
| 总传输速率 | 32/64/128/256Ghz | 64/128/256/512Ghz | 128/256/512/1024Ghz |

表8-3 性能指标

发送端DPL中的每条通道包含有加扰器、128b/130b编码器构成。接收端通道中包含有解扰器、128b/130b解码器以及对应alignment等模块构成。其整体传输流如下所示：



图8-4 DPL数据流框图

链路层发来的数据经过扰码器加扰后，送给最大128b/130b编码器。编码后的数据需要经过一个异步FIFO来进行跨时钟域处理。之后即进入EPL中串行化。

在测试模式或者是建链状态下，提供一个PRBS生成器，产生伪随机码流。

接收端通过Alignment识别对比COM码来进行数据的定帧，再送给128b/130b解码器。解码后的数据进行128bit解扰处理，并进行基于编码规则的错误检测。

注意:接收端由于EPL在开启工作后会一直向DPL层传输数据，因此接收端的DPL层不能反压EPL层的数据。

* 1. 加扰和随机码生成

当数据重复传输时，能量就会集中在某一频率上，产生EMI噪声。因此需要将集中的能量分散开来，让其几乎变成白噪声。并且一定程度上，更有利于时钟的提取，同时起到加密的效果。

扰码是一种将数据重新排列或者进行编码以使其随机化的方法，且能解扰恢复。打乱长的连“0”和长的连“1”序列，将数据随机化。定义了一个线性反馈移位寄存器(LFSR)，用于扰码和随机码的生成。目前选用的扰码器采用128bit位宽，多项式为：



每条通道的种子是不同的，同一条通道接收端和发送端的种子必须相同。默认情况下的种子值如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 通道序列 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 种子值（h） | 1DBFBC | 0607BB | 1EC760 | 18C0DB | 010F12 | 19CFC9 | 0277CE | 1BB807 |

不同的种子决定了LFSR的周期性序列。若是链路进行了交叉，则内部需要根据交叉后的值进行种子的重新映射。

LFSR实现：

图8-5 加扰原理图

加扰规则：

* 数据和控制字符在正常的传输过程中需要进行加扰。
* 测试模式下支持不加扰。
* COM字符不加扰，且接收或发送COM字符时，LFSR当前值重置为初始种子。
  1. 编解码

本标准采用128/130b编码，将128bit数据或控制字符编码为130bit块传输，bit[129:128]表示同步头，主要用于接收端的数据对齐。同步头有“01”和“10”两种，“01”表示bit[127:0]是数据字符，表示为D码，“10”表示bit[127:0]是控制字符，表示为K码。

K码可以根据协议层定义混合K码和D码的方式。需要注意的是，COM码作为特殊的控制码，用于做定帧和同步，因此不包含数据，也不能被加扰。128b/130b编码格式如下图所示：

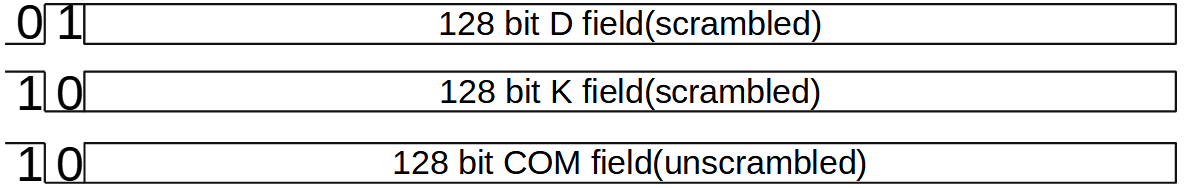


图8-6 128b/130b编码格式

接收到同步头为“11”或“00”的块是同步错误情况，并将此错误上报。

* 1. **Alignment**

由于SerDes的发送端和接收端的启动时间不同，并且在发送数据的过程中，容易出现数据偏移，因此需要Alignment进行数据同步，使得DPL层能够准确找到字符块的起始位置。

LINK 发送端会周期性的发送COM码作为同步信息。Alignment模块的目标是检测到COM码，并将数据对齐发送。因此需要在内部缓存一拍数据，并和下一拍的数据拼接成一个260bit的buffer进行检测。检测到COM码时，将输出数据重新同步到正确的位置，并输出当前位置为label。



图8-7 alignment识别COM码示意图

* + 1. 可信度计数器

由于alignment功能是依靠识别COM码的固定码方式，有可能会存在着两个数据拼接中恰好和COM码相同的情况。因此在这里增加一个可信度计数器credible\_cnt：每当收到一个COM码时，label的位置没有发生改变，则可信度计数器加一。若收到COM码时，label的位置发生了改变，则可信度计数器减一，若可信度计数器等于0，则可以更新最新的label值。如图所示：



图8-8 可信度计数器流程图

最大可信度作为可配置使用，且最大可信度建议不要配置为0。

若是出现了由于时钟偏差而导致的累积错误，即链路可能会在运转一段时间后多出一个或少一个bit，那么此时的label应该需要改变，才能够正确接收到数据。此时可信度计数器credible\_cnt会在每接收一个COM后减一，直到减到0，label才会改变。因此最大可信度credible\_max会影响出现这个问题后的恢复时间：最大可信度配置的越大，则需要收到COM码的数量越多。

alignment收到一个com码，会上报align\_down信号。

如果label发生改变，会上报align\_change信号。

在align\_done信号之前，所有传输的数据都是无效数据。

* 1. 链路交叉

Die To Die在封装打线的时候，不一定会按照顺序连接。因此需要支持链路全交叉，即可以通过链路选择寄存器和链路模式寄存器配置通道的数量和物理交叉。

链路选择寄存器只作用在发送端，用于控制发送端的每一条通道应该取哪一个发送端的LINK层通道的数据。这里默认定义link层的数据是从小到大依次进行分发。即如果是配置的链路模式寄存器为2，则表示有四条通道进行工作，那么link层下发的数据即是0号、1号、2号、3号数据。



图8-9 发送端LINK层到DPL层数据有效通路

RX端会根据EPL层检测到的产生能量的signal\_detect信号来判断，是哪条通道上获取到了数据。这里默认定义了，RX端的数据会依通道序号的排列顺序，从小到大依次将接收端的DPL层层通道的数据发送给RL层。如果是配置的链路模式寄存器为2，则表示有四条通道进行工作，那么DPL层接收的数据需要送给link层的0号、1号、2号、3号数据通路。



图8-10 接收端DPL层到LINK层数据有效通路

例如：下图为在链路模式寄存器等于2，即开启四条通道的示意图。此时物理上的连线如下：

* 发送端的EPL的0号通道和接收端的EPL的4号通道相连；
* 发送端的EPL的2号通道和接收端的EPL的7号通道相连；
* 发送端的EPL的3号通道和接收端的EPL的3号通道相连；
* 发送端的EPL的7号通道和接收端的EPL的1号通道相连。

由于接收端的DPL层会将激活的通道重新排序，从小到大依次推入Received Link层。那么可以判断出来，发送端的DPL层需要做如下操作才能够满足数据链路的正常通路：

* 发送端的DPL层的0号通道和发送端的EPL的7号通道相连；
* 发送端的DPL层的1号通道和发送端的EPL的3号通道相连；
* 发送端的DPL层的2号通道和发送端的EPL的0号通道相连；
* 发送端的DPL层的3号通道和发送端的EPL的2号通道相连。

因此需要根据得到的连接关系，配置发送端的DPL层的各个通道的数据来源。如下图表示在上述连接关系中，DPL到EPL的链路交叉示例。



图8-11 链路交叉示例

全交叉功能亦可用于链路维修。若在使用的1/2/4条通道的情况下，其中一条通道出现物理性损坏，可以通过全交叉功能，将未被使用的通道利用起来。

注：交叉功能改变后，必须重新建链才可使用。

* 1. 通道极性反转

设备需要支持通道的极性反转，以应对复杂多变的Die To Die封装情况。本标准定义了Master设备和Slave设备，因此可以通过配置Master Die的发送端和接收端进行单边的极性反转即可，如下图所示：

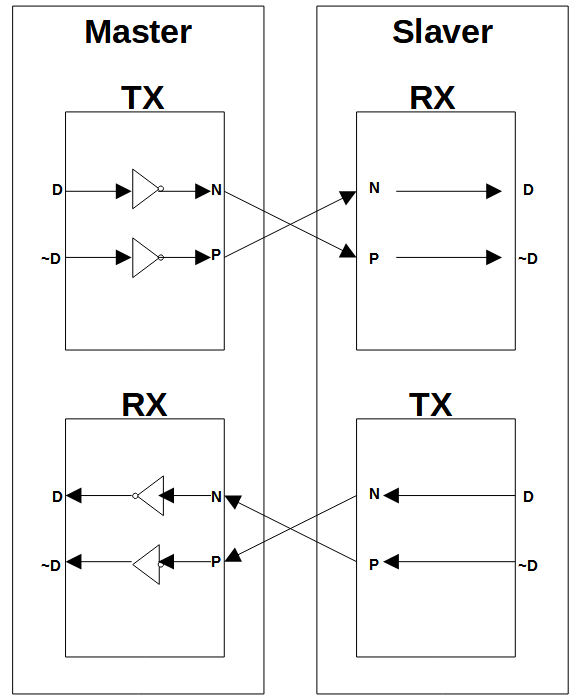


图8-12 链路极性反转示意图

1. 电气物理层（Electrical Physical Layer）

电气规格的关键属性包括:

* 支持单通道128/64/32GT/s的数据率；
* 差分数据传输；
* 直流或者交流耦合的Die-to-Die互联；
* 数据中内嵌时钟,不需要单独的通道传送时钟信号。
  1. 数据率

最高支持128GT/s的数据传输，并向下兼容64/32GT/s的数据传输。

* 1. 电学参数

|  |  |  |  |
| --- | --- | --- | --- |
|  | **标准封装** | | |
| Data Rate (GT/s) | 32 | 64 | 128 |
| Power Efficiency Target (pJ/b) | 2.5 | 2.5 | 2.5 |
| Latency Target (TX+RX) (ns) | 6 | 6 | 6 |
| Idle Power (% of peak power) | 10 | 10 | 10 |
| Channel Reach (mm) | 1~50 | 1~50 | 1~50 |
| Die Edge Bandwidth Density (GB/s/mm) | 32 | 64 | 128 |
| Bandwidth area density (GB/mm^2) | 40 | 80 | 160 |
| PHY dimension width (um) | 800 | 800 | 800 |
| PHY dimension Depth (um) | 2000 | 2000 | 2000 |
| ESD | 100V CDM | 100V CDM | 100V CDM |

表9-1 PHY的电学参数

* 1. 发送端

32GT/s发送端的框架如图9-2所示，发送端包含Serializer，FFE，DRV，T-coil等子模块。

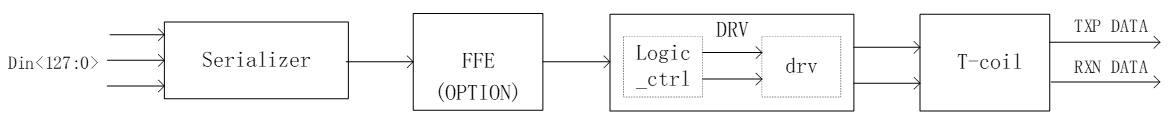


图9-2 发送器架构框图

Serializer是串化器，将并行输出信号转化为串行输出信号，发送给接收端。

FFE是前馈均衡器。由于通道良好，它的功能是可选的，只在128GT/s的应用中需要，64GT/s和32GT/s的应用中则不需要。

* + 1. 发送端的电学参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数名称** | **最小值** | **典型值** | **最大值** | **单位** | **备注** |
| 发送信号摆幅 | 0.8 | 0.9 | 1 | Vpp | NRZ |
| 信号摆幅修调的范围 | 340 |  | 500 | mV |
| 信号摆幅修调的步长 |  | 11.1 |  | mV |
| PAM-4 信令00对应的电平 |  | 0 |  | V | PAM-4 |
| PAM-4 信令01对应的电平 |  | 0.3 |  | V |
| PAM-4 信令10对应的电平 |  | 0.6 |  | V |
| PAM-4 信令11对应的电平 |  | 0.9 |  | V |
| 差分输出阻抗 | 80 | 100 | 120 | Ohm |  |
| 输出总的Jitter |  | 6 |  | ps | 32GT/s |
|  | 6 |  | ps | 64GT/s |
|  | 3 |  | ps | 128GT/s |

表9-3 发送端的电学参数

* + 1. 驱动电路

如图9-4所示，驱动电路由驱动逻辑电路（Driver），输出OPTION电路和T-coil组成：

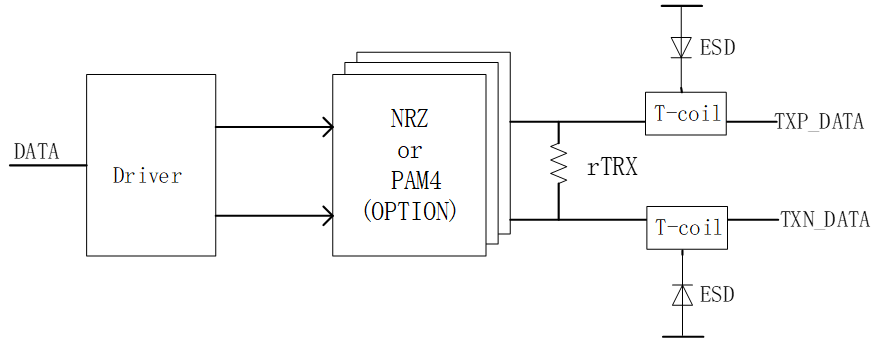


图9-4 发送端驱动的框图

驱动电路针对NRZ和PAM-4两种编码方式进行了电路结构的优化，通过寄存器配置(OPTION)就可以切换编码输出方式，满足不同应用的需求。128/64GT/s的应用采用PAM-4结构，而32GT/s的应用采用NRZ结构。

驱动电路支持减小输出信号摆幅；根据不同的通道特性，改变输出电阻网络的寄存器配置，可以实现对输出信号摆幅的控制。

* + 1. 发送端的前馈均衡

为了缓解接收端均衡的压力，在发送端使用FFE(Feed Forward Equalizer)技术对信号进行预均衡。FFE基于有限冲激响应滤波器设计实现，将延时的信号按照不同的权重（C-1，C0，…，Cn）相加。控制权重的大小即可调整均衡强度。实质是使用数字线性高通滤波器提高信号的高频分量，实现对信道的补偿，减小ISI(Inter-Symbol Interference)效应的影响。

发射端的前馈均衡，基于FIR滤波器设计，2阶3TAP的结构，电路框图如图9-5所示：

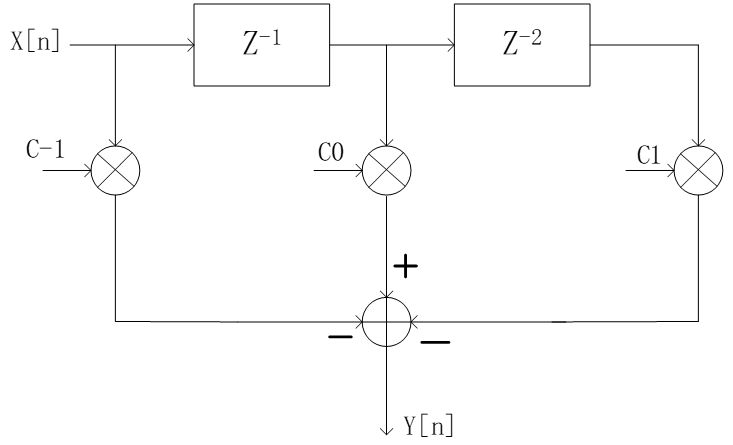


图9-5 前馈均衡器的框图



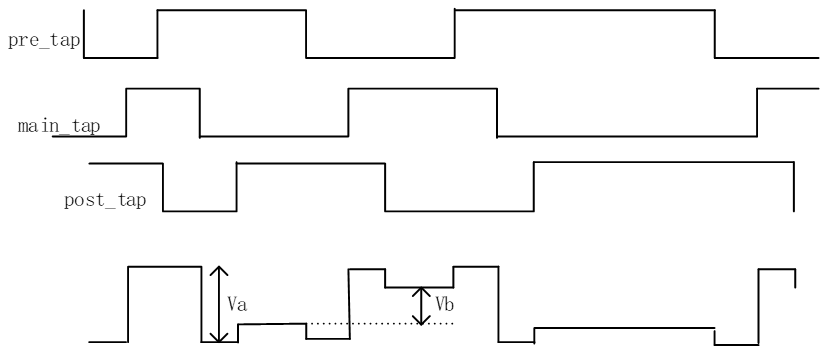


图9-6 发送端均衡的原理图





注1：C-1，C0，C1是发送端main-tap，pre-tap，post-yap的系数；

注2：C-1和C1表示的系数是负的；

注3：均衡的效果由C-1，C0，C1，的比例决定。

* 1. 接收端

接收端的框图如图9-7所示，接收端包含TERN，CTLE(OPTION),CDR, Deserializerd等子模块。

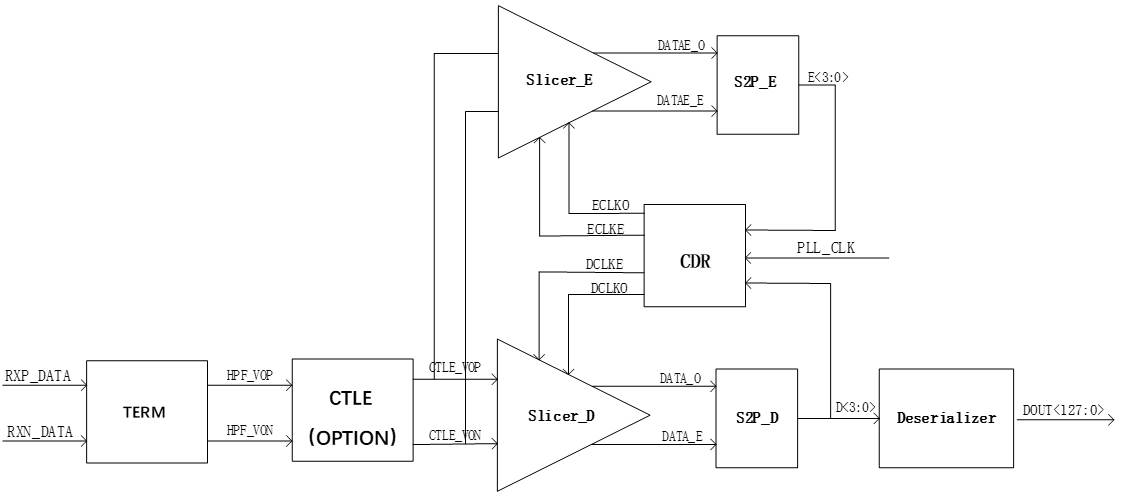


图9-7 接收端框图

由于通道的良好特性，CTLE的功能是可选的，只在128GT/s的应用中需要，64GT/s和32GT/s的应用中则不需要。

CDR是时钟恢复电路，主要从串行输入的信号中恢复出时钟信号。

Deserializerd是解串器，将串型输入数据转为并行输出数据。

* + 1. 接收端的电学参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数名称** | **最小值** | **典型值** | **最大值** | **单位** | **注释** |
| TERM电阻的阻抗 | 80 | 100 | 120 | Ohm |  |
| TERM电阻修调的范围 | 72 |  | 175 | Ohm |  |
| TERM电阻修调的步长 |  | 3 |  | Ohm |  |
| 接收端眼图的眼高 | 400 |  |  | mV | NRZ |
| 133 |  |  | mV | PAM-4 |
| 接收端眼图的眼宽 | 0.5 |  |  | UI | NRZ |
| 0.17 |  |  | UI | PAM-4 |

表9-8 接收端的电学参数

* + 1. 连续时间线性均衡器（CTLE）

对于传输32/64GT/s的数据率时，当合适的TX均衡设置后，在最坏的通道和封装模型下，传输至接收端的眼图仍然满足要求，所以在32/64GT/s中不需要在接收端进行均衡。

但是传输128GT/s的数据时，传输至接收端的眼图已不能满足要求，所以需要在接收端进行均衡，采用的均衡是1阶的CTLE。

CTLE的原理图如下所示：

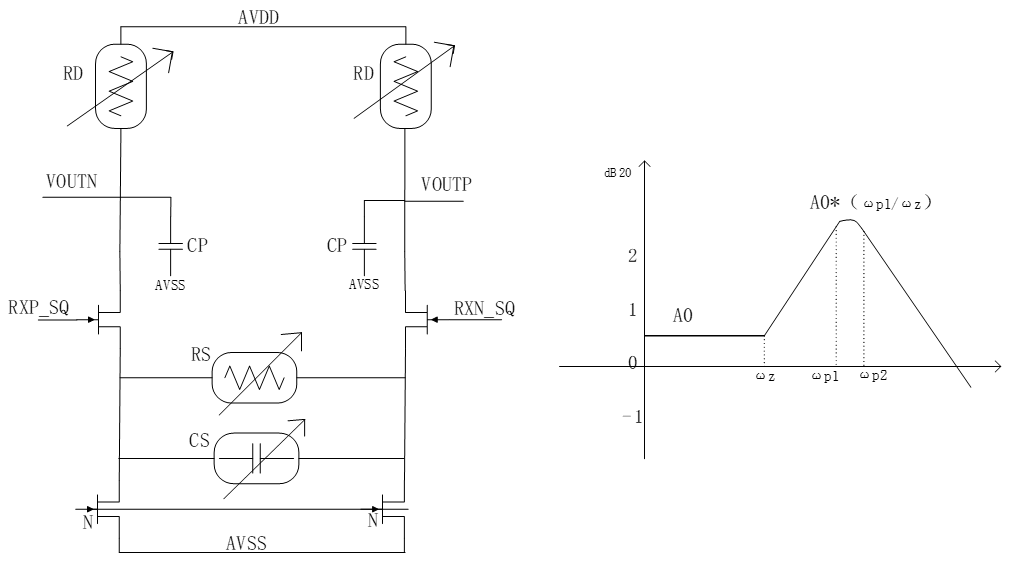
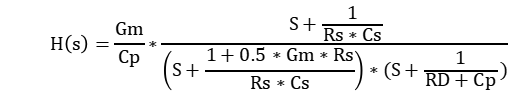


图9-9 CTLE电路原理图

CTLE原理：



直流增益： 

顶点增益：

CTLE的工作原理是直接通过线性模拟高通滤波器拟合信道的衰减，实现信道的补偿。

信道的频率响应等效为一个低通滤波器，而CTLE通过减小低频信号的方式补偿高低频的衰减差，从而补偿信道的衰减。

* + 1. 时钟恢复电路CDR

时钟数据恢复主要完成两个工作，一个是时钟恢复，一个是数据重定时。即将嵌入在数据中的时钟信息提取出来，并通过恢复的时钟将接收到的数据重定时，采样含噪声的数据，从而消除数据在传输过程中引入的抖动，其性能决定了整个串行传输系统的性能。

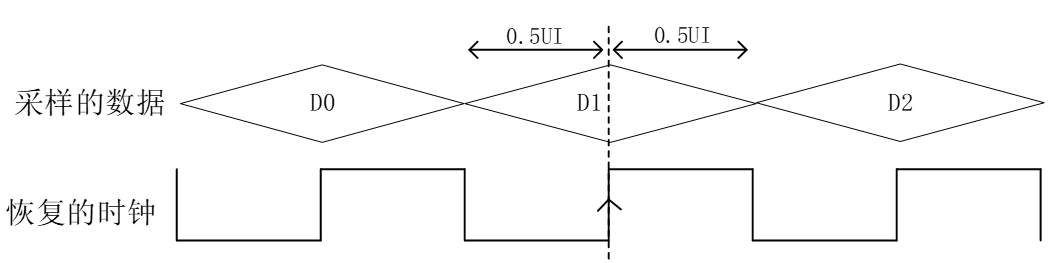


图9-10 CDR回复后的时钟采样图

1. 边带传输 Sideband (Optional)

本标准中边带信号用于近端对远端读写寄存器。

Sideband包含2个引脚，一个是双向的串行时钟线，只由近端发送给远端，另一个是双向的串行数据线，当近端给远端写数据时，信号由近端发送给远端，当远端给近端返回数据时，信号由远端发送给近端。

Sideband串行时钟线速度不低于10MHz，接收端根据时钟上升沿对数据线进行采样，先传输最高位，传输信号示意图如下：

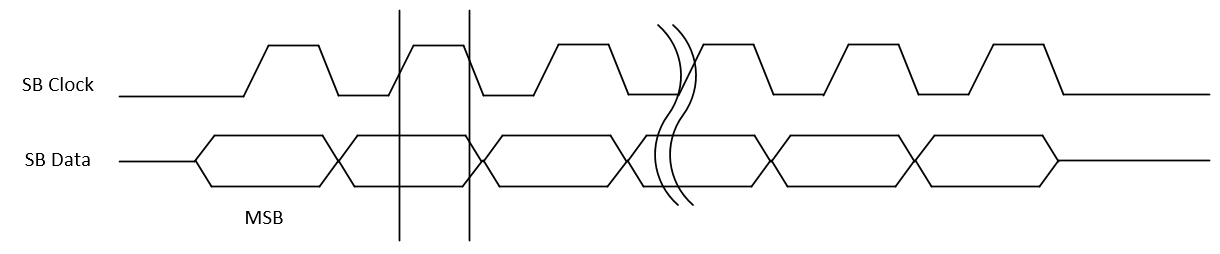


图10-1 Sideband信号

近端将对远端的寄存器访问打包并发送，远端接收到边带传输的协议包后解析并做出响应，如果是写寄存器，则立即执行，如果是读寄存器，则需要8个空拍才可回复寄存器值。

写寄存器包包长为64bit，包格式如下：

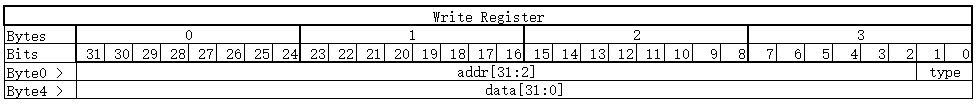


图10-2 写寄存器格式

|  |  |
| --- | --- |
| **位域** | **描述** |
| addr[31:2] | 写寄存器地址的[31:2]，addr[1:0]固定为0，由近端发送给远端。 |
| type[1:0] | 对于写寄存器来说，该值固定为2’b11，由近端发送给远端 |
| data[31:0] | 写寄存器数据[31:0]，由近端发送给远端 |

表10-3 写寄存器格式描述

读寄存器包包长为72bit，包格式如下：

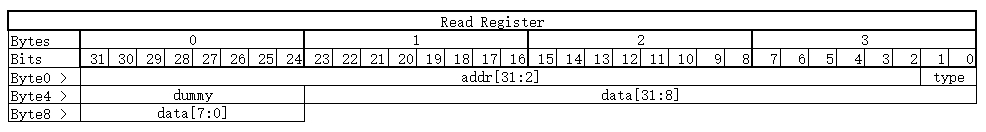


图10-4 读寄存器格式

|  |  |
| --- | --- |
| **位域** | **描述** |
| addr[31:2] | 读寄存器地址的[31:2]，addr[1:0]固定为2’b00，由近端发送给远端 |
| type[1:0] | 对于读寄存器来说，该值固定为2’b00，由近端发送给远端 |
| dummy[7:0] | 空闲拍，近端只发送8拍时钟并释放数据线 |
| data[31:0] | 读寄存器数据[31:0]，有远端发送给近端 |

表10-5 读寄存器格式描述

1. 接口定义



图11-1 接口概述

如上图所示，本标准中将D2D接口划分为4个层次，分别为：

* 协议层（Protocol Layer）；
* 链路层（Link Layer）；
* 数字物理层（Digital Physic Layer）；
* 电气物理层（Electrical Physic Layer）。

在本标准中对于D2D的接口定义了如上4个层次之间的接口，分别为：

* 协议层与链路层之间的接口，即PL-LL Interface（PLI）；
* 链路层与数字物理层之间的接口，即LL-DPL Interface（LDI）；
* 数字物理层与电气物理层之间的接口，即DPL-EPL Interface（DEL）；
* D2D近端与远端的接口，通过近端与远端的EPL相连，即EPL-EPL Interface（EEI）。
  1. **PL-LL Interface（PLI）**

PLI为协议层与链路层之间的接口。对于发送方向，协议层将来自上层系统的数据流与控制流打包，然后将其传输到链路层；而接收方向，链路层通过CRC检测，剥离链路层包等操作，将其中的协议层数据向上传输至协议层。具体接口如下：

|  |  |
| --- | --- |
| **接口名** | **接口描述** |
| prot2link\_vld[0:0] | 发送侧，协议层向链路层指示prot2link\_data/prot2link\_tail上的数据有效 |
| link2prot\_rdy[0:0] | 发送侧，链路层向协议层指示可以接收数据，使用valid-rdy握手机制传递数据 |
| prot2link\_data[1023:0] | 发送侧，协议层向链路层传递的协议层包的其中1拍数据 |
| prot2link\_tail[0:0] | 发送侧，协议层向链路层传递的协议层包的尾标记 |
| link2prot\_vld[0:0] | 接收侧，链路层向协议层指示link2prot\_data/link2prot\_tail上的数据有效 |
| prot2link\_rdy[0:0] | 接收侧，协议层向链路层指示可以接收数据，使用valid-rdy握手机制传递数据 |
| link2prot\_data[1023:0] | 接收侧，链路层向协议层传递的协议层包的其中1拍数据 |
| link2prot\_tail[0:0] | 接收侧，链路层向协议层传递的协议层包的尾标记 |

表11-2 协议层与链路层接口

* 1. **LL-DPL Interface（LDI）**

本节定义了链路层与数字物理层之间的接口，在发送方向，链路层在接收到协议层发送的数据之后，对其进行添加STP，END等字符，进行ID编号，CRC校验，插入链路层包等操作后将数据按照多个通道的方式发送到数字物理层，在接收方向上，数字物理层将电气物理层发送的数据进行定帧，解码，解扰等操作后按照多通道的方式将数据发送到链路层，具体接口定义如下：

|  |  |
| --- | --- |
| 接口名 | 接口描述 |
| link2phy\_valid[0:0] | 发送侧，链路层向物理层指示link2phy\_data/link2phy\_dk上的数据有效 |
| phy2link\_rdy[0:0] | 发送侧，物理层向链路层指示可以接收数据，使用valid-rdy握手机制传递数据 |
| link2phy\_data[1023:0] | 发送侧，链路层向物理层传递的数据流，每个通道128bit，共8个通道 |
| link2phy\_dk[7:0] | 发送侧，链路层向物理层传递的数据流的控制字符/数据字符标记，每个通道1bit，共8个通道  1'b0：控制字符  1'b1：数据字符 |
| phy2link\_valid[0:0] | 接收侧，物理层向链路层指示phy2link\_data/phy2link\_dk上的数据有效 |
| phy2link\_data[1023:0] | 接收侧，物理层向链路层传递的数据流，每个通道128bit，共8个通道 |
| phy2link\_dk[7:0] | 接收侧，物理层向链路层传递的数据流的控制字符/数据字符标记，每个通道1bit，共8个通道  1'b0：控制字符  1'b1：数据字符 |

表11-3 链路层与数字物理层接口

* 1. **DPL-EPL Interface（DEI）**

本节定义了数字物理层与电气物理层之间的接口，在发送方向，数字物理层在接收到链路层发送到数字物理层各个通道上的数据进行加扰编码等操作后，按照一定的方式发送电气物理层，而接收方向上，电气物理层将对端发送的数据经过处理后按照一定的方式发送数字物理层，具体接口描述如下：

|  |  |
| --- | --- |
| 接口名 | 接口描述 |
| epl2dpl\_tx\_clk[7:0] | 电气物理层发送数字物理层的发送时钟，数字物理层需要将数据同步到该时钟下发送到电气物理层 |
| dpl2epl\_tx\_dat[127:0][7:0] | 数字物理层发送到电气物理层的数据接口，分为8个通道，每个通道128bit |
| epl2dpl\_rx\_clk[7:0] | 电气物理层发送数字物理层的接收时钟，电气物理层需要通过该时钟对数据进行接收 |
| dpl2dpl\_rx\_dat[127:0][7:0] | 电气物理层发送到数字物理层的数据接口，分为8个通道，每个通道128bit |

表11-4 数字物理层与电气物理层接口

* 1. **EPL-EPL Interface（EEI）**

本节定义D2D近端与远端的接口，即近端电气物理层与远端电气物理层的接口，这部分接口分为数据接口与边带传输接口，而边带传输接口为可选项（如第十章所述），对于数据接口，发送与接收方向均为高速差分串行接口；边带传输信号（sideband）为sb\_clock与sb\_data。具体接口描述如下：

|  |  |
| --- | --- |
| 接口名 | 接口描述 |
| tx\_dat\_p[7:0] | 发送端串行数据线p分量 |
| tx\_dat\_n[7:0] | 发送端串行数据线n分量 |
| rx\_dat\_p[7:0] | 接收端串行数据线p分量 |
| rx\_dat\_n[7:0] | 接收端串行数据线n分量 |
| sb\_clock | 边带传输时钟信号（可选） |
| sb\_data | 边带传输数据信号（可选） |

表11-5 电气物理层与电气物理层接口

1. 调试追踪 Debug&Trace

本章描述各个层次提供的调试追踪功能，目的是方便从外部通过寄存器接口全方位了解各个层次内部的运行状态。主要包括如下6个功能类别。

* 统计计数功能；
* 注入错误、暂停发包功能；
* 测试序列发送、检测和校验功能；
* 环回功能；
* 告警/状态上报功能；
* 跟踪功能；
  1. **测试序列发送、检测和校验功能**

发送端支持发送如下测试码流，用于接收端校验所接收数据的正确性：

* 发送端数字物理层支持发送PRBS7、PRBS9、PRBS15、PRBS23、PRBS31序列；
* 发送端数字物理层支持循环发送用户自定义的长度为2048bit的固定序列；

接收端支持检测如下测试码流，用于接收端校验所接收数据的正确性：

* 接收端数字物理层支持校验PRBS7、PRBS9、PRBS15、PRBS23、PRBS31序列，并单独上报每个通道的校验结果；
* 接收端数字物理层支持检测用户自定义的长度为2048bit的固定序列，并单独上报每个通道的校验结果；

每条通道皆需具备测试序列的发送和检测校验功能，接收端的错误校验器通过对比参考PRBS生成器生成的序列数据和接收到的测试序列数据（或是用户定义序列），来判断数据在通道中是否发生了错误。若发生了错误，则将错误信息上报。如图12-1 PRBS生成和检测示意图所示。



图12-1 PRBS生成和检测示意图

* 1. **环回功能**

环回功能主要用于测试过程中对链路通讯进行确认，对于D2D接口的环回功能，本标准定义了不同的环回点进行环回测试，如图12-2所示。



图12-2 环回路径示意图

本标准中定义的环回点如下：

* 环回点0：从近端数字物理层的发送端环回到近端数字物理层的接收端；
* 环回点1：从近端电气物理层的发送端环回到近端电气物理层的接收端；
* 环回点2：从远端链路层的接收端环回到远端链路层的发送端。

1. 配置参数 Configuration and Parameters

本标准中提出D2D必须的寄存器配置，配置寄存器如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **寄存器名称** | **位宽** | **初始值** | **寄存器描述** |
| code\_stp | 7:0 | 0xFB | STP符号 |
| code\_sdp | 7:0 | 0x5C | SDP符号 |
| code\_end | 7:0 | 0xFD | END符号 |
| code\_com | 31:0 | 0xBCBCBC7d | COM符号 |
| code\_idl | 7:0 | 0xDC | IDL符号 |
| code\_pad | 7:0 | 0x0 | PAD符号 |
| idle | 0:0 | 0x0 | 通过该配置可以将数据链路状态切换至idle状态，高有效 |
| train\_link\_en | 0:0 | 0x0 | 通过该配置信号向controller发起建链的请求，高有效 |
| train\_rate | 1:0 | 0x3 | EPL通讯速率：  2’b00：32GT  2’b01：64GT  2’b10：128GT  2’b11：reserve |
| lane\_enable | 7:0 | 0xFF | 配置需要使用到的通道，8bit对应8条通道，对应bit为1表示使能该通道 |
| lane\_mode | 1:0 | 0x3 | 通道模式选择： 2’b00：1通道模式 2’b01：2通道模式 2’b10：4通道模式 2’b11：8通道模式 |
| lane\_link | 23:0 | 0xFAC688 | 8条通道的连接方式，对其配置可以实现各通道的全交叉连接，从低到高每3bit表示一条通道的交叉连接方式，例如[2:0]=0x7，则表示第0条通道交叉连接到第7条通道 |
| loopback | 10 | 0x0 | loopback模式： 3’b00：非loopback mode 3’b01：loopback0 mode，即本端DPL的发送与接收环回 3’b10：loopback1 mode，即本端EPL的发送与接收环回 3’b11：loopback2 mode，即从远端链路层的接收与发送端环回 |
| data\_sca\_bypass | 0:0 | 0x0 | 数据加扰bypass 使能信号，为1时bypass掉加扰 |
| training\_time | 4:0 | 0x2 | training状态下发生超时的时间限制，当超过该时间限制还未training完成则认为超时，每增加1则增加500us，例如当training\_time为5'd10时，则时间为5ms |
| null\_send\_len | 15:0 | 0x3FF | TX LINK链路训练一次使能最少发送的NULL序列个数，发送个数为配置值加1，16'd1023 |
| acknak\_lantency\_time | 15:0 | 0xff | 在接收到TLP的情况下，每隔多少Cycle产生ACK或NAK |
| wait\_expect\_id\_time | 15:0 | 0x1ff | 在产生NAK情况下，应该在多少个Cycle内收到重发的包 |
| crc\_check\_bypass | 0:0 | 0x0 | 不检测CRC校验错误，高电平有效 |
| null\_det\_len | 15:0 | 0x10 | RX LINK检测NULL码个数 |
| tx\_dpl\_polar\_reverse | 7:0 | 0x0 | DPL发送端8条通道的极性反转使能 |
| rx\_dpl\_polar\_reverse | 7:0 | 0x0 | DPL接收端8条通道的极性反转使能 |

表13-1 寄存器列表

如上寄存器需要在建链之前，需要D2D近端与远端进行配置统一，然后分别配置D2D近端与远端的如下使能信号：

|  |  |  |  |
| --- | --- | --- | --- |
| epl\_pll\_pu | 0:0 | 0x0 | EPL的pll使能 |
| epl\_tx\_pu | 7:0 | 0x0 | EPL 8条通道发送端的使能信号 |
| epl\_rx\_pu | 7:0 | 0x0 | EPL 8条通道接收端的使能信号 |

表13-2 EPL使能配置

如表14-1与14-2中的配置，本标准提出两种配置方式：边带传输配置方式与自定义传输配置：

* 边带传输方式见第10章；
* 自定义传输配置为D2D近端与远端本身就具有除了D2D EPL接口之外的其他可靠通讯接口，即可复用该接口进行表14-1的寄存器配置统一以及表14-2的使能操作。

1. 封装 Package
   1. **封装类型**

应使用bump pitch不大于153um的Flip-Chip（FC）或2.5D封装形式，以便于实现高密度的高速信号互联；过大的bump pitch会导致D2D接口扇面过大，浪费die面积的同时也降低了PKG层面的走线能力。



图14-1 封装类型

* 1. **Bump方案**

Bump次序应如下图所示排列，对接端Bump次序旋转180°，即可实现对应TX、RX的对接，信号线不会产生交叉，有利于PKG绕线和信号质量。

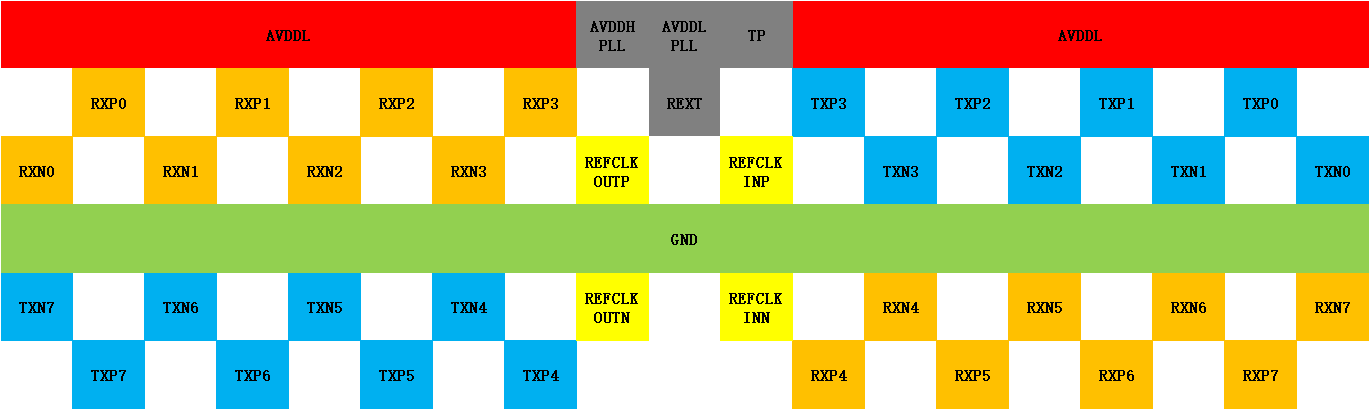


图14-2 Bump方案

* 1. **绕线方案**

参考绕线方案如下图所示，L1、L3层为TX、RX信号走线层，L2为参考地层，一般还需在L4层做参考地保护L3层的信号；因此至少需要用到4层无core层的PKG layer。

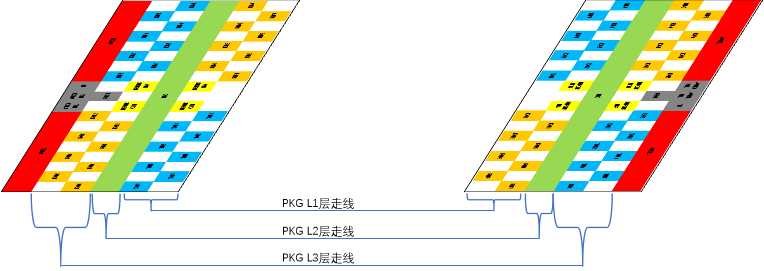


图14-3 绕线方案

* 1. **信号规范**

封装信号走线应参照下表规范：

|  |  |  |  |
| --- | --- | --- | --- |
|  | **最小值** | **典型值** | **最大值** |
| PN skew |  |  | 10um |
| Pair skew |  |  | 100um |
| Length |  |  | 10mm |
| Impedance | 90Ω | 100Ω | 110Ω |

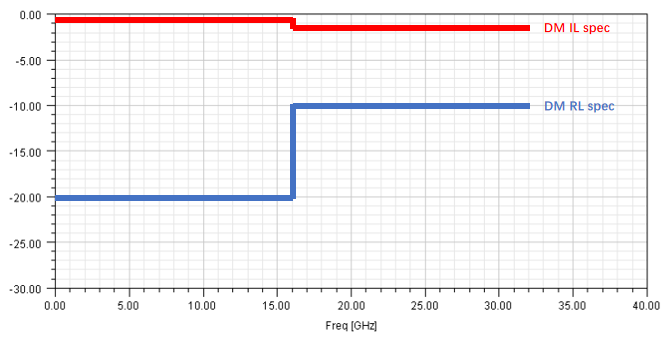
表14-4 封装信号走线规范

PKG信号走线损耗及隔离度应满足下表规范：

|  |  |  |
| --- | --- | --- |
|  | **0-16GHz** | **16-32GHz** |
| Insertion loss | <1.5dB | <3dB |
| Return loss | <-20dB | <-10dB |
| Diff isolation | <-30dB | |
| Comm isolation | <-15dB | |

表14-5 PKG信号走线规范

PKG信号走线损耗的spec mask如下图所示：



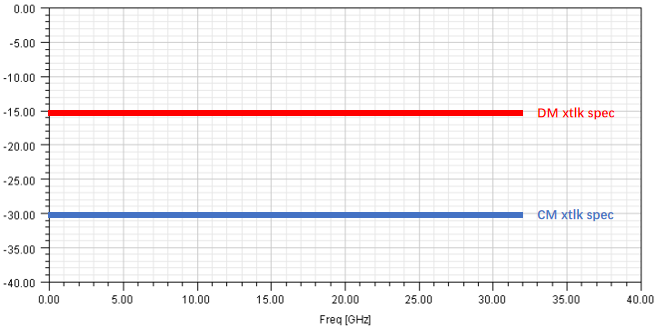


图14-6 PKG信号走线损耗示意图

* 1. **时钟方案**

为使系统能够简化时钟方案，D2D phy需兼容下图所示的时钟并联和时钟串联方案，因此必须有clkin和clkout，clkout输出时钟信号与clkin输入的时钟信号同频同相。

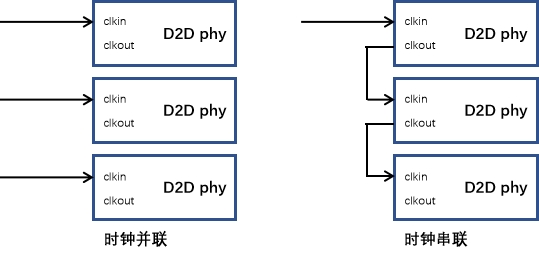


图14-7 时钟方案

* 1. **信号完整性**

为保证信号传输稳定可靠，RX端的1e-12误码率眼图需满足下表中的指标：

|  |  |  |
| --- | --- | --- |
|  | **NRZ** | **PAM4** |
| 接收端最小眼宽 | 0.5UI | 0.17UI |
| 接收端最小眼高 | 400mV | 133mV |

表14-8 信号完整性指标

NRZ眼图如下图所示，菱形区域为eyemask，信号眼图的眼宽眼高应大于eyemask区域：

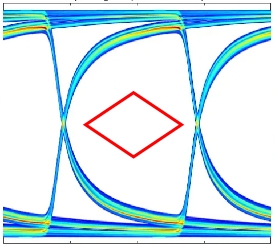


图14-9 NRZ眼图

PAM4眼图如下图所示，3个菱形区域为eyemask，信号眼图的眼宽眼高应大于eyemask区域：



图14-10 PAM4眼图