网络模块对外接口设计

|  |  |  |
| --- | --- | --- |
| 版本 | 日期 | 修订 |
| 2.0 | 2015-6-12 | 黄廉真 |
| 3.0 | 2017-1-17 | 黄廉真 |
| 4.0 | 2017-1-22 | 黄廉真 |

## 修订

2017/1/17 对于user[5:0],它划分为2个定义，des\_ch和src\_ch。而最高位则定义为是否为mailbox通信。

在后面的使用中，由于设计FPGA通道PDU是不带应答的。而FPGA的mailbox才可能进行应答。这导致PC的应用层程序实现困难。

当前约定ID=0表示访问FPGA，其他值则表示访问其他通道，最多支持7个通道。因为

{1'b0，user[5:0]，len[8:8]}这个数据不能为0xFF。

为了让PC的mail也可以直接访问FPGA的PDU。特定定义，des\_ch或者src\_id的bit2作为交换bits。从而实现pdu通道可以访问mail通道。mail通道也可以访问pdu通道。

详见4.3部分。

2017/1/22 对mapping的地址映射重新修改。为了最大限度压缩存储。定义了新的映射方式。

原来的映射是直接存储间接地址。此处修改了起始地址+访问模式+长度。另外也保留原有的模式。那样存储的为起始地址，长度固定为1。同时去掉FMMU\_ADDR和FMMU\_LEM两个寄存器。

所有的映射详见2.2 逻辑映射区域。

port寄存器增加watch\_link标志。当连接一个外部设备，该外部设备发送一个特殊的报文，则该端口进行watch模式。

## 目录

[目录 2](#_Toc421881418)

[第一部分：网络模块对外接口类型 4](#_Toc421881419)

[1.1应用层接口分类 4](#_Toc421881420)

[1.2 DSP多通道实现协议 6](#_Toc421881421)

[周期传输方式 6](#_Toc421881422)

[查询配置传输方式 6](#_Toc421881423)

[1.3 DSP发送DMA接口设计 7](#_Toc421881424)

[TX\_CFG 8](#_Toc421881425)

[TX\_CTRL 9](#_Toc421881426)

[FPGA模块接口 9](#_Toc421881427)

[1.4 DSP接收DMA接口设计 11](#_Toc421881428)

[查询方式 11](#_Toc421881429)

[预使能方式 11](#_Toc421881430)

[RX\_CFG 12](#_Toc421881431)

[RX\_CTRL 13](#_Toc421881432)

[FPGA模块接口 13](#_Toc421881433)

[第二部分 网络数据链路层协议接口 14](#_Toc421881434)

[接收数据协议： 14](#_Toc421881435)

[发送数据协议 15](#_Toc421881436)

[第三部分 FPGA应用层协议接口 16](#_Toc421881437)

[2.1数据区协议 16](#_Toc421881438)

[2.2 三片存储区域 16](#_Toc421881439)

[用户数据区域 17](#_Toc421881440)

[逻辑地址映射区域 17](#_Toc421881441)

[文发送配置区域 17](#_Toc421881442)

[2.3 报文发送配置区定义 18](#_Toc421881443)

[第四部分 DSP应用层协议接口 20](#_Toc421881444)

[4.1 DSP访问站点的DMA通道协议 20](#_Toc421881445)

[4.1.1 DSP发送写请求协议格式 21](#_Toc421881446)

[4.1.2 DSP写请求应答返回协议 22](#_Toc421881447)

[4.1.3 DSP发送读请求协议格式 23](#_Toc421881448)

[4.1.4 DSP读请求应答返回协议 24](#_Toc421881449)

[4.2 DSP与DSP之间访问协议 25](#_Toc421881450)

[第五部分 FPGA网络接口设计 26](#_Toc421881451)

[5.1发送接口设计 26](#_Toc421881452)

[5.2接收接口设计 27](#_Toc421881453)

[5.3 DSP通道访问寄存器接口 28](#_Toc421881454)

[PDU\_TX\_CFG 28](#_Toc421881455)

[PDU\_TX\_CTRL 29](#_Toc421881456)

[MAIL\_TX\_CFG 29](#_Toc421881457)

[MAIL\_TX\_CTRL 31](#_Toc421881458)

[第六部分 工作时间 32](#_Toc421881459)

# 第一部分：网络模块对外接口类型

## 1.1应用层接口分类

针对网络系统本身的应用，它的应用层主要以下几类：

1. 无DSP管理协议。这种情况通常发生在站点仅仅只有一个FPGA。FPGA仅仅作为逻辑计算模块。该应用层仅仅是一个被动模块，而网络模块本身作为一个主动模块。根据以往的经验，为了让FPGA应用程序简单化，可以实现一个DMA模块。该模块的作用是实现网络协议，让网络数据转换为DSP总线命令。

如下图。在网络模块和FPGA模块见，实现一个DMA模块。该模块的作用主要是网络接口转换为EBIU接口进行控制FPGA模块。从而使得FPGA模块的接口最简单化。

这种方式适用于应用层没有DSP处理模，不能实现复杂的网络协议管理；或者DSP不愿因管理网络协议。



Fig1.1 无DSP网络应用接口模式

1. DSP单通道实现协议。这种情况是指DSP内部管理整个应用层的网络协议，FPGA的网络部分仅仅进行数据链路层的数据传输，以及管理同步；同时DSP与FPGA的网络接口通道只有一个。

如下图所示。



Fig1.2 DSP单通道实现协议模式

网络数据链路层的数据仅仅与DSP在一个通道上进行交互。只要DSP能够获得网络数据链路层的状态，就可以根据实际情况进行对网络数据链路层进行控制，实现网络通信。网络的组包与拆包任务在DSP内完成。整个系统的同步实现，则通过Interrupt信号实现。

1. DSP多通道实现协议。这个情况相对比较复杂。DSP每一路都能够和网络数据链路层进行通信。因此，DSP的每一路都能够进行组包和拆包。那么在网络数据链路层与DSP之间必须有一个控制管理器。如下图所示。每一个通道可以实现独立传输。



Fig1.3 DSP多通道实现协议模式

在理论上，该模式涵盖了前两种情况。对于FPGA情况，只要在其中一个通道上，添加一个DMA模块就能涵盖第一种情况。

因此，只要在设计的时候只要实现第三种情况，同时实现DMA模块就能实现兼容所有情况。

## 1.2 DSP多通道实现协议

按照上图Fig1.3描述。考虑资源问题，最多支持的通道数位8个通道。而根据实际的应用情况，可知可能与网络数据链路层进行通信的实际物理通道主要有：SPORT，LINKPORT，EBIU，PCI，PCIE等。

对于以上的接口进行分析，可以知道。PCI，PCIE，EBIU都是数据地址操作类型。SPORT，LINKPORT等属于数据流型。数据地址操作型细分为DMA方式非DMA方式。数据流型本质上类似DMA方式。因此，数据的传输可分为两大类：DMA方式和非DMA方式。

DMA方式的特点是启动传输后不需要CPU干预。传输启动后，一般不再进行干预，因此，它需要预先知道传输数据长度和数据的起始地址。而非DMA方式可以一边传输，一边进行传输形式的修正。由于非DMA方式的灵活性大于DMA方式，因此核心在于设计DMA方式。

针对DMA方式的特点。它的工作有以下几种类型。

### 周期传输方式

第一次配置起始地址，传输数据长度，等待触发信号，然后开始进行传输。传输完成后给出标志位（可用可不用），随后等待下一次的触发信号。以此循环。

这种方式主要出现在与周期数据传输，在正常工作中，这些数据必须传输，长度和地址不会发生任何变化

### 查询配置传输方式

模块通过周期或者非周期查询缓存区状态，根据状态进行选择性传输。

这种数据被认为是非周期性数据，而这种数据的实行性不够。这种方式必须配合寄存器操作。

## 1.3 DSP发送DMA接口设计

DSP往FPGA进行DMA传输时，无论是周期性传输或者非周期性传输，都必须按照以下方式进行。

1. 读取检查FPGA相关通道的DMA状态，用于判断当前DMA是否准备好。它具有两个意义：用于判断当前是否可以进行DMA传输和判断前一次的DMA是否正常执行完成。
2. 根据实际情况进行必然的参数配置。而周期性传输基本可以不用配置。这些配置信号主要用于FPGA判断DMA的执行情况。
3. DSP开始传输至FPGA。
4. 触发传输。DSP将数据传输给FPGA，但是FPGA在何时，根据哪种方式进行网络传输则需要这些触发配置信息。根据实际情况，可以DSP一边往FPGA传输，FPGA一边进行网络传输；也可以等待整个报文DSP与FPGA的DMA传输完成了，立刻发送或者等待特定的触发信号再发送。

根据以上的特点，每一个发送通道的接口应该包括以下配置功能。

### TX\_CFG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| 15~13 |  |  |  | 保留 |
| 12~11 | priority | 1 | 可读写 | 该数据通道的优先级。0最高，3最低。 |
| 10~9 | model | 1 | 可读写 | 0=ATUO\_END\_TX。当FPGA处于tx\_ing=0时，只要FPGA接收到len个数据后，自动启动发送  1=WAIT\_ACTIVE。当FPGA处于tx\_ing=0时，并接收到len个报文后，等待active信号，才开始发送报文。  2=WAIT\_SYNC。当FPGA处于tx\_ing=0时，并接收到len个报文后，等待外部sync信号，才开始发送报文。  3=ATUO\_TX。当FPGA处于tx\_ing=0时，只要FPGA开始接收到第一个数据时，就开始启动传输。这个模式必须要求DMA的传输速度必须大于等于网络速度。这个模式只用在要求传输延迟非常小的情况。 |
| 8~0 | len | 100 | 可读写 | 该值的取值范围为1~480。  表示即将要发送数据长度。该长度是DSP往FPGA传输的数据整个长度，以byte为单位。该值取值范围不能超出设定值。否则出错。 |

### TX\_CTRL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| 15 | active |  | 只写 | 写1有效，则启动一次发送。自动清零。 |
| 14 | clr |  | 只写 | 写1清除所有状态位 |
| 13 | data\_valid | 0 |  | 1：数据有效，可以等待发送。FPGA接收len个报文后自动置1。表示报文接收完成  0：FPGA发送完整个报文后，该值自动清零。 |
| 12 | finish | 0 | 只读 | 1：buffer中的数据已经传输完成。  0：buffer中的数据没有被传送完成。禁止DSP往FPGA 进行新的DMA传输。 |
| 11 | ing | 0 | 只读 | 1：buffer中数据正在往外发送。禁止DSP往FPGA进行新的DMA传输。  0：当前没有在传输。可能处于FPGA接收DMA的数据，也可能处于空闲状态。 |
| 10 | err | 0 | 只读 | 1：当数据传送的时候，DSP往FPGA发送新的数据  0：无错 |
| 9 | len\_err | 0 | 只读 | 当发送模式为WAIT\_ACTIVE或者WAIT\_SYNC时，触发信号来后，FPGA接收的报文长度不够len或者超出了len。 |
| 8 | bf\_clr | 0 | 只写 | 写1清除buffer数据。当数据是不是处于发送过程中，可以将buffer清除。如果数据正在传输，清除将失败。 |
| 7 | bf\_clr\_err | 0 | 只读 | 当数据正在发送，清除失败，报错。 |
| 6~0 | 保留 |  |  |  |

### FPGA模块接口

|  |  |  |
| --- | --- | --- |
| IO | 命名 | 备注 |
| parameter | DATA\_WIDTH = 32, | //数据入口宽度 |
| parameter | SIMULINK\_EN = 0 |  |
| input | clk, |  |
| input | rst, |  |
| input | param\_wr, |  |
| input | [3:0]param\_addr, |  |
| input | [15:0]param\_din, |  |
| output | [15:0]param\_dout, |  |
| input | [DATA\_WIDTH-1:0]din |  |
| input | valid, |  |
|  |  |  |
| xxx | xxxx |  |

## 1.4 DSP接收DMA接口设计

DSP通过DMA读取FPGA数据时，如果是周期性数据，可以按照以下方式进行传输：

### 查询方式

1.当读取数据的时刻到后，查询FPGA是否接收到数据。

2.如果接收到数据，则启动传输，否则进入异常处理。

### 预使能方式

1.在每一次读取完数据的时候，立刻启动下一次的传输。

2.FPGA一旦收到数据后就将数据传送给DSP。

3.DSP在特定的时候去读取状态，判断是否读取完成。

4.如果读取完成，则进行计算，同时再次启动下一次传输。否则进入异常处理。

对于启动传输，有两个层面，1个是DSP内部的DMA的启动。另一个是FPGA的启动。对于交互的式的（需要握手），在正常计算开始后，可让FPGA一直使能状态，DSP只要适时启动内部DMA即可。如LINKPORT接口。

对于非交互式的（无握手），则需要DSP对FPGA进行控制，只有在DSP启动DMA后，才能打开FPGA的传输。当传输结束后，DSP需要关闭FPGA的传输，等待下一次传输。如SPORT接口。

当前DSP通过DMA读取FPGA数据为非周期数据时，则需要采用查询握手方式。DSP周期后者不定时查询buffer状态，根据buffer状态进一步决定是否启动传输。

### RX\_CFG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| 15 | active |  | 只写 | 写1有效，则启动一次发送。自动清零。 |
| 14 | clr |  | 只写 | 写1清除所有状态位 |
| 13 | bf\_clr |  | 只写 | 写1清除buffer数据 |
| 10~9 | model | 1 | 可读写 | 0=ATUO\_END\_TX。当FPGA接收完整个报文后，立刻自动启动DMA往DSP发送。  1=WAIT\_ACTIVE。当FPGA接收完整个报文后，等待active信号，才开始启动DMA往DSP发送报文。  2=WAIT\_SYNC。当FPGA接收完整个报文后，等待sync信号，才开始启动DMA往DSP发送报文。  3=ATUO\_TX。只要FPGA开始接收到第一个数据时，就开始启动才开始启动DMA往DSP发送报文。这个模式只用在要求传输延迟非常小的情况。此时要求DSP与FPGA通信速度小于网络速度。 |
| 8~0 | len | 100 | 可读写 | FPGA往DSP发送数据长度1~480。  表示即将要发送数据长度，以byte为单位。该值取值范围不能超出设定值。否则出错。 |

### RX\_CTRL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| 15 | bf\_lose | 0 | 只读 | 当bf中有数据，新的数据无法存储下，则报错。这个报文会完全丢弃，从而保证报文的一致性。 |
| 14 | overflow |  |  | 1：buffer溢出，错误会被锁存，clr才能清除。  0：正常。 |
| 13 | data\_valid | 0 |  | 1：数据有效，可以等待发送。FPGA从网络接收到一个完整报文后，置1。  0：FPGA往DSP发送len个报文后清零。 |
| 12 | finish | 0 | 只读 | 1：buffer中的数据往DSP传送len后置1。在下一次传输激活时，该值被清零。  0：buffer中数据正在往DSP传送。 |
| 11 | ing | 0 | 只读 | 0：FPGA没有往DSP进行传输。  1：buffer中数据正在往DSP传送。 |
| 10 | err | 0 | 只读 | 1：buffer中没有数据，DSP强行读取数据。此时读取无效。错误会被锁存，clr才能清除。  0：无错 |
| 9:0 | bf\_num | 0 | 只读 | 当前buffer中数据长度。 |

### FPGA模块接口

|  |  |  |
| --- | --- | --- |
| IO | 命名 | 备注 |
| parameter | DATA\_WIDTH = 32, | //数据出口宽度 |
| parameter | SIMULINK\_EN = 0 |  |
| input | clk, |  |
| input | rst, |  |
| input | param\_wr, |  |
| input | [3:0]param\_addr, |  |
| input | [15:0]param\_din, |  |
| output | [15:0]param\_dout, |  |
| output | [DATA\_WIDTH-1:0]dout, |  |
| output | valid, |  |
| input | ready, |  |
| xxx | xxxx |  |

# 第二部分 网络数据链路层协议接口

该接口是网络数据层直接的接口。该接口主要用，上层期望仅调用数据链路层，各个站点之间的应用层自定制。

一般在该接口与其他接口之间有一个转换过程。如果不是直接面对数据链路层接口，则可以不管该接口。

**DSP或者CPAC不需要关注该接口协议**。

## 接收数据协议：

|  |  |  |
| --- | --- | --- |
| 序列 | 意义 | 备注 |
| 1 | src\_id | 数据源ID。 |
| 2 | { type，user[5:0]，len1[8:8]} | 数据区1数据长度高1位。type=1:mail |
| 3 | len1[7:0] | 数据区1数据长度低8位。 |
| 4 | data[0] | 数据1 |
| …… |  | 数据1 |
| 4+len1-1 | data[len1-1] | 数据1 |
| 4+len1 | {1'b0，user[5:0]，len2[8:8]} | 数据区2数据长度高1位。 |
| 5+len1 | len2[7:0] | 数据区2数据长度低8位。 |
| 6+len1 | data[0] | 数据2 |
| …… |  | 数据2 |
| 6+len1+len2-1 | data[len-1] | 数据2 |
| …… |  | 其他数据区 |
|  | endflag | ***必须取值0xff*** |

整个最大的数据总大小不能大于492。

## 发送数据协议

|  |  |  |
| --- | --- | --- |
| 序列 | 意义 | 备注 |
| 1 | des\_id | 需要发送至站点。0xff：所有其他站点都会接收。不能取值0xf0~0xfe。 |
| 2 | {type，user[5:0]，len1[8:8]} | 数据区1数据长度高1位。type=1:mail |
| 3 | len1[7:0] | 数据区1数据长度低8位。 |
| 4 | data[0] | 数据1 |
| …… |  | 数据1 |
| 4+len1-1 | data[len1-1] | 数据1 |
| 4+len1 | {1'b0，user[5:0]，len2[8:8]} | 数据区2数据长度高1位。 |
| 5+len1 | len2[7:0] | 数据区2数据长度低8位。 |
| 6+len1 | data[0] | 数据2 |
| …… |  | 数据2 |
| 6+len1+len2-1 | data[len-1] | 数据2 |
| …… |  | 其他数据区 |
|  | endflag | ***必须取值0xff*** |

整个最大的数据总大小不能大于492。

**详细的物理接口见《ring net网络设计.doc》的“****第五部分：数据链路层应用接口”**

# 第三部分 FPGA应用层协议接口

该协议作用是应用FPGA实现的DMA模块协议。它的作用参见本文档的第一部分的“1.1应用层接口分类”的“无DSP管理协议”一类。如果某一个站点采用FPGA实现DMA模式，那么要通过该DMA获取数据时，必须按照该协议访问。

## 2.1数据区协议

该协议仅支持1个数据区

|  |  |  |
| --- | --- | --- |
| 序列 | 意义 | 备注 |
| 1 | src\_id/des\_id | 数据源ID/目标ID。 |
| 2 | {1'b0，user[5:0]，len[8:8]} | 数据区的数据长度1位。  访问FPGA DMA通道，user必须为0。 |
| 3 | len[7:0] | 数据区的数据长度低8位。 |
| 4+0 | {1’b0,  num[8],  resp[0:0]  resp\_req[0:0]  delta[0:0],  cmd[2:0]  } | cmd=0(LWR):逻辑寻址写  cmd=1(AWR):绝对寻址写  cmd=2(LMWR):逻辑地址映射区寻址写  cmd=3(TCWR):报文发送配置区寻址写  cmd=4(LRD):逻辑寻址读  cmd=5(ARD):绝对寻址读  cmd=6(LMRD):逻辑地址映射区寻址读  cmd=7(TCRD):报文发送配置区寻址读  delta[0:0]DMA访问地址增量值。  0：每次访问同一个地址；  1：每次访问都累加1。  resp\_req =0:不需要返回应答。  resp\_req =1:需要返回应答。当cmd=4,5,6,7,resp\_req必须为1. |
| 4+1 | num[7:0] | DMA寻址次数， |
| 4+2 | start\_addr[7:0] | DMA启动地址低8位（byte 地址）,寻址不支持非对齐模式 |
| 4+3 | start\_addr[15:0] | DMA启动地址高8位（byte 地址） |
| 4+4 |  | 整个数据，数据长度必须为num= len-4。 |
| …… |  |
| 4+len-1 |  |

## 2.2 三片存储区域

在APP层，有三片区域。1片是用户数据区域，1片是逻辑地址映射区域，1片是报文发送配置区域。

其中逻辑地址映射区和报文发送配置区总大小为8\*2048(bit)。其中0x0~0x3FF（byte地址，为DMA CFG，而0x400~0x7FF为MAP区。

### 用户数据区域

通过LWR和AWR可以访问得到，这边地址空间范围是0x0000~0xffff；（byte 地址）

### 逻辑地址映射区域

它是LWR访问需要用的区域。LWR是一个间接寻址的方式。通过配置逻辑地址映射区域的数据，可以修改逻辑寻址的对象。逻辑地址寻址映射区域的访问范围是0x0000~0x01ff（byte 地址）。该区域需要通过LMWR访问。

该区域是一片存储地址的内存。在LWR访问下，APP层根据数据区中的指令，先计算出地址用于访问该区域，获得需要实际访问的逻辑地址，然后再进一步访问用户数据区域的数据。存储的地址为byte地址。

纯地址模式（默认）：

|  |  |  |
| --- | --- | --- |
| 序列 | 意义 | 备注 |
| 1 | addr[7:0] | 起始地址低位(short地址) |
| 2 | addr[15:8] | 起始地址高位(short地址) |
| 3 | addr[7:0] | 起始地址低位(short地址) |
| 4 | addr[15:8] | 起始地址高位(short地址) |
| 4+0 | addr[7:0] | 起始地址低位(byte 地址) |
| 4+1 | addr[15:8] | 起始地址高位(byte 地址) |
| 4+2 | addr[7:0] | 起始地址低位(short地址) |
| 4+3 | addr[15:8] | 起始地址高位(short地址) |
| 4+4 |  |  |
| …… |  |
| 4+len-1 |  |

地址+长度模式，定义下的格式：

|  |  |  |
| --- | --- | --- |
| 序列 | 意义 | 备注 |
| 1 | addr[7:0] | 起始地址低位(byte 地址) |
| 2 | addr[15:8] | 起始地址高位(byte 地址) |
| 3 | word\_num[7:0] | 数据长度(word的个数)低8位。  1 表示1个， 0表示最大 |
| 4 | {6’b0,  enable, delta[0:0]} | enable : 表示是否使能这个地址，在读情况，该值无效。  delta：表示地址递增或者保持不变 |
| 4+0 | addr[7:0] | 起始地址低位(byte 地址) |
| 4+1 | addr[15:8] | 起始地址高位(byte 地址) |
| 4+2 | word\_num[7:0] | 数据长度(word的个数)低8位。  1 表示1个， 0表示最大 |
| 4+3 | {6’b0,  enable, delta[0:0]} | enable : 表示是否使能这个地址，在读情况，该值无效  delta：表示地址递增或者保持不变 |
| 4+4 |  |  |
| …… |  |
| 4+len-1 |  |

### 文发送配置区域

该区域用于配置从站需要发送哪些数据到哪些站点，已经发送模式等。该区域需要通过TCWR访问。

## 2.3 PDU DMA发送配置区定义

|  |  |  |
| --- | --- | --- |
| 地址 | 意义 | 备注 |
| 0 | {  bit7 = 1’b0;  bit6 = rd\_num[8]  bit5 = rd\_num\_src[0:0]  bit4=1’b0  bit3:2=rd\_cmd[1:0],  bit1=rd\_delta[0:0]  bit0=rd\_en=1’b1(保留)  } | rd\_cmd=0(LRD):逻辑寻址读（必须为该值）  rd\_cmd=1(ARD):绝对寻址读  rd\_cmd=2(LMRD):逻辑地址映射区寻址读  rd\_cmd=3(TCRD):报文发送配置区寻址读  rd\_delta:DMA访问地址增量值  rd\_num\_src=0:数据长度由rd\_num决定  rd\_num\_src=1:数据长度由外部决定，通过rd\_start\_addr获得rd\_num，同时也决定下面的len。  rd\_en表示这个报文需要发送，该值必须为1，否则解析将就此结束，后面的也不会在解析。 |
| 1 | rd\_num[7:0] | RD\_DMA寻址次数。 |
| 2 | rd\_start\_addr[7:0] | RD\_DMA启动地址低8位（byte addr） |
| 3 | rd\_start\_addr[15:0] | RD\_DMA启动地址高8位（byte addr） |
| 4 | check\_sum[7:0] | sum{0~11} = 0xff |
| 5 | des\_id | 数据源ID/目标ID。 |
| 6 | {1'b0，user[5:0]，len[8:8]} | 数据区的数据长度1位。  user[2:0]表示这个数据要返回到des\_id的那个通道。  user[5:3]保留 |
| 7 | len[7:0] | 数据区的数据长度低8位。~~len由内部计算~~， |
| 8 | {1’b0,  num[8],  resp[0:0]  resp\_req[0:0]  delta[0:0],  1’b0,  cmd[1:0]  } | cmd=0(LWR):逻辑寻址写（必须为该值）  cmd=1(AWR):绝对寻址写  cmd=2(LMWR):逻辑地址映射区寻址写  cmd=3(TCWR):报文发送配置区寻址写  delta[0:0]DMA访问地址增量值。  0：每次访问同一个地址；  1：每次访问都累加1。  resp\_req恒为0。  cmd不能取4,5,6,7； |
| 9 | num[7:0] | DMA寻址次数，由内部计算 |
| 10 | start\_addr[7:0] | DMA启动地址低8位（byte 地址）,寻址不支持非对齐模式 |
| 11 | start\_addr[15:8] | DMA启动地址高8位（byte 地址） |
| 12 | {  rd\_num[8]  rd\_num\_src[0:0]  rd\_cmd[1:0],  rd\_delta[0:0],  rd\_model, rd\_cycle\_num[2:0]  } | rd\_model=1:每一个周期都发送一次，  rd\_model=0:当rd\_cycle\_num!=0,一个周期发送一次,同时修改rd\_cylce\_num--。  rd\_cmd=0(LRD):逻辑寻址读  rd\_cmd=1(ARD):绝对寻址读  rd\_cmd=2(LMRD):逻辑地址映射区寻址读  rd\_cmd=3(TCRD):报文发送配置区寻址读  rd\_delta:DMA访问地址增量值  rd\_num\_src=0:数据长度由rd\_num决定  rd\_num\_src=1:数据长度由外部决定，通过rd\_start\_addr获得rd\_num，同时也决定下面的len。 |
| 13 | rd\_num[7:0] | RD\_DMA寻址次数。 |
| 14 | rd\_start\_addr[7:0] | RD\_DMA启动地址低8位 |
| 15 | rd\_start\_addr[15:8] | RD\_DMA启动地址高8位 |
| 16 |  | 保留 |
| 17 | des\_id | 数据源ID/目标ID。 |
| 18 | {1'b0，user[5:0]，len[8:8]} | 数据区的数据长度1位。  user[2:0]表示这个数据要返回到des\_id的那个通道。  user[5:3]保留 |
| 19 | len[7:0] | 数据区的数据长度低8位。len由内部计算， |
| 20 | {1’b0,  num[8],  resp[0:0]  resp\_req[0:0]  delta[0:0],  cmd[2:0]  } | cmd=0(LWR):逻辑寻址写  cmd=1(AWR):绝对寻址写  cmd=2(LMWR):逻辑地址映射区寻址写  cmd=3(TCWR):报文发送配置区寻址写  delta[0:0]DMA访问地址增量值。  0：每次访问同一个地址；  1：每次访问都累加1。  resp\_req恒为0。  cmd不能取4,5,6,7； |
| 21 | num[7:0] | DMA寻址次数，由内部计算 |
| 22 | start\_addr[7:0] | DMA启动地址低8位（byte 地址）,寻址不支持非对齐模式 |
| 23 | start\_addr[15:8] | DMA启动地址高8位（byte 地址） |
|  | 8’b0 | 解析到这里，结束。 |

# 第四部分 DSP应用层协议接口

## 4.1 DSP访问站点的DMA通道协议

当DSP需要访问FPGA实现的DMA通道，那么就必须按照参照第三部分的协议。但是在数据结构略做调整。由于第三部分的协议是按照byte进行计算，而DSP的总线可能是16位或者32位。为了便于DSP的字节对齐，需要填充一个无效byte。如下表

因此，无论发送和接收，DSP在访问DMA通道的时，后者DMA往DSP返回数据，数据格式都是按照以下。

|  |  |  |
| --- | --- | --- |
| 序列 | 意义 | 备注 |
| 0 | 4’b0 | 无效扩充 |
| 1 | src\_id/des\_id | 数据源ID/目标ID。 |
| 2 | {1'b0, user[5:0]，len[8:8]} | 数据区的数据长度1位。  访问FPGA DMA通道，user[2:0] = des\_ch必须为0。  user[5:3]=src\_ch |
| 3 | len[7:0] | 数据区的数据长度低8位。 |
| 4+0 | {1’b0,  num[8],  resp[0:0]  resp\_req[0:0]  delta[0:0],  cmd[2:0]  } | cmd=0(LWR):逻辑寻址写  cmd=1(AWR):绝对寻址写  cmd=2(LMWR):逻辑地址映射区寻址写  cmd=3(TCWR):报文发送配置区寻址写  cmd=4(LRD):逻辑寻址读  cmd=5(ARD):绝对寻址读  cmd=6(LMRD):逻辑地址映射区寻址读  cmd=7(TCRD):报文发送配置区寻址读  delta[0:0]DMA访问地址增量值。  0：每次访问同一个地址；  1：每次访问都累加1。  resp\_req =0:不需要返回应答。  resp\_req =1:需要返回应答。当cmd=4,5,6,7,resp\_req必须为1. |
| 4+1 | num[7:0] | DMA寻址次数， |
| 4+2 | start\_addr[7:0] | DMA启动地址低8位（byte 地址）,寻址不支持非对齐模式 |
| 4+3 | start\_addr[15:0] | DMA启动地址高8位（byte 地址） |
| 4+4 |  | 整个数据，数据长度必须为num= len-4(不再需要)。 |
| …… |  |
| 4+len-1 |  |

### 4.1.1 DSP发送写请求协议格式

如下表所示，cmd根据实际情况选择哪种访问，user[2:0]必须取值为0~6，用于访问对应的DMA通道，如果对应站点没有该DMA通道，则不会有任何响应。然后根据实际情况选择是否需要应答。

注意：如果需要应答，resp\_req取1，**此时user[5:3]表示报文返回后选择送到的DSP通道，该值必须正确，否则该应答报文可能去到别的通道，导致应答失败**。如不需要应答，resp\_req取0，user[5:3]可取任意值。

数据区的数据为将要写入的数据。此时len和num有对应关系，len=num+4。

|  |  |  |
| --- | --- | --- |
| 序列 | 意义 | 备注 |
| 0 | 8’b0 | 无效扩充 |
| 1 | src\_id/des\_id | 数据源ID/目标ID。 |
| 2 | {1'b0，user[5:0]，len[8:8]} | 数据区的数据长度1位。  访问FPGA DMA通道，user[2:0]必须为0。  user[5:3]根据DSP通道决定。 |
| 3 | len[7:0] | 数据区的数据长度低8位。 |
| 4+0 | {1’b0,  num[8],  resp[0:0]  resp\_req[0:0]  delta[0:0],  cmd[2:0]  } | cmd=0(LWR):逻辑寻址写  cmd=1(AWR):绝对寻址写  cmd=2(LMWR):逻辑地址映射区寻址写  cmd=3(TCWR):报文发送配置区寻址写  cmd=4(LRD):逻辑寻址读  cmd=5(ARD):绝对寻址读  cmd=6(LMRD):逻辑地址映射区寻址读  cmd=7(TCRD):报文发送配置区寻址读  delta[0:0]DMA访问地址增量值。  0：每次访问同一个地址；  1：每次访问都累加1。  resp\_req =0:不需要返回应答。  resp\_req =1:需要返回应答。当cmd=4,5,6,7,resp\_req必须为1. |
| 4+1 | num[7:0] | DMA寻址次数， |
| 4+2 | start\_addr[7:0] | DMA启动地址低8位（byte 地址）,寻址不支持非对齐模式 |
| 4+3 | start\_addr[15:0] | DMA启动地址高8位（byte 地址） |
| 4+4 |  | 整个数据，数据长度必须为num= len-4(不再需要)。 |
| …… |  |
| 4+len-1 |  |

### 4.1.2 DSP写请求应答返回协议

如下表所示，返回的报文中，resp\_req=1, resp=1，表示属于应答报文。此时len=4。也即没有数据区的数据。

|  |  |  |
| --- | --- | --- |
| 序列 | 意义 | 备注 |
| 0 | 4’b0 | 无效扩充 |
| 1 | src\_id/des\_id | 数据源ID/目标ID。 |
| 2 | {1'b0，user[5:0]，len[8:8]} | 数据区的数据长度1位。  访问FPGA DMA通道，user[2:0]将和user[5:3]交换。 |
| 3 | len[7:0] | 数据区的数据长度低8位。 |
| 4+0 | {resp\_ok,  num[8],  resp[0:0]  resp\_req[0:0]  delta[0:0],  cmd[2:0]  } | cmd=0(LWR):逻辑寻址写  cmd=1(AWR):绝对寻址写  cmd=2(LMWR):逻辑地址映射区寻址写  cmd=3(TCWR):报文发送配置区寻址写  cmd=4(LRD):逻辑寻址读  cmd=5(ARD):绝对寻址读  cmd=6(LMRD):逻辑地址映射区寻址读  cmd=7(TCRD):报文发送配置区寻址读  delta[0:0]DMA访问地址增量值。  0：每次访问同一个地址；  1：每次访问都累加1。  resp\_req =0:不需要返回应答。  resp\_req =1:需要返回应答。当cmd=4,5,6,7,resp\_req必须为1. |
| 4+1 | num[7:0] | DMA寻址次数， |
| 4+2 | start\_addr[7:0] | DMA启动地址低8位（byte 地址）,寻址不支持非对齐模式 |
| 4+3 | start\_addr[15:0] | DMA启动地址高8位（byte 地址） |

### 4.1.3 DSP发送读请求协议格式

如下表所示，cmd根据实际情况选择哪种访问，user[2:0]必须取值为0~6，用于访问对应的DMA通道，如果对应站点没有DMA通道，则不会有任何响应。然后根据实际情况选择是否需要应答。

注意：**对于读请求，不管resp-req一定会有应答，此时user[5:3]表示报文返回后选择送到的DSP通道，该值必须正确，否则该应答报文可能去到别的通道，导致应答失败**。

数据区没有数据len=4。

|  |  |  |
| --- | --- | --- |
| 序列 | 意义 | 备注 |
| 0 | 4’b0 | 无效扩充 |
| 1 | src\_id/des\_id | 数据源ID/目标ID。 |
| 2 | {1'b0，user[5:0]，len[8:8]} | 数据区的数据长度1位。  访问FPGA DMA通道，user[2:0]必须为0。  user[5:3]根据DSP通道决定。 |
| 3 | len[7:0] | 数据区的数据长度低8位。 |
| 4+0 | {1’b0,  num[8],  resp[0:0]  resp\_req[0:0]  delta[0:0],  cmd[2:0]  } | cmd=0(LWR):逻辑寻址写  cmd=1(AWR):绝对寻址写  cmd=2(LMWR):逻辑地址映射区寻址写  cmd=3(TCWR):报文发送配置区寻址写  cmd=4(LRD):逻辑寻址读  cmd=5(ARD):绝对寻址读  cmd=6(LMRD):逻辑地址映射区寻址读  cmd=7(TCRD):报文发送配置区寻址读  delta[0:0]DMA访问地址增量值。  0：每次访问同一个地址；  1：每次访问都累加1。  resp\_req =0:不需要返回应答。  resp\_req =1:需要返回应答。当cmd=4,5,6,7,resp\_req必须为1. |
| 4+1 | num[7:0] | DMA寻址次数， |
| 4+2 | start\_addr[7:0] | DMA启动地址低8位（byte 地址）,寻址不支持非对齐模式 |
| 4+3 | start\_addr[15:0] | DMA启动地址高8位（byte 地址） |

### 4.1.4 DSP读请求应答返回协议

如下表所示，返回的报文中，resp\_req=1, resp=1，表示属于应答报文。此时len=num+4。也即数据区的数据为读取的数据。

|  |  |  |
| --- | --- | --- |
| 序列 | 意义 | 备注 |
| 0 | 4’b0 | 无效扩充 |
| 1 | src\_id/des\_id | 数据源ID/目标ID。 |
| 2 | {1'b0，user[5:0]，len[8:8]} | 数据区的数据长度1位。  访问FPGA DMA通道，user[2:0]将和user[5:3]交换。 |
| 3 | len[7:0] | 数据区的数据长度低8位。 |
| 4+0 | {resp\_ok,  num[8],  resp[0:0]  resp\_req[0:0]  delta[0:0],  cmd[2:0]  } | cmd=0(LWR):逻辑寻址写  cmd=1(AWR):绝对寻址写  cmd=2(LMWR):逻辑地址映射区寻址写  cmd=3(TCWR):报文发送配置区寻址写  cmd=4(LRD):逻辑寻址读  cmd=5(ARD):绝对寻址读  cmd=6(LMRD):逻辑地址映射区寻址读  cmd=7(TCRD):报文发送配置区寻址读  delta[0:0]DMA访问地址增量值。  0：每次访问同一个地址；  1：每次访问都累加1。  resp\_req =0:不需要返回应答。  resp\_req =1:需要返回应答。当cmd=4,5,6,7,resp\_req必须为1. |
| 4+1 | num[7:0] | DMA寻址次数， |
| 4+2 | start\_addr[7:0] | DMA启动地址低8位（byte 地址）,寻址不支持非对齐模式 |
| 4+3 | start\_addr[15:0] | DMA启动地址高8位（byte 地址） |
| 4+4 |  | 整个数据，数据长度必须为num= len-4。 |
| …… |  |
| 4+len-1 |  |

## 4.2 DSP与DSP之间访问协议

由于DSP 通道与DSP通道交互，因此跟更高级的内部定义有DSP之间相互协商。而往网络上传输的数据必须满足一下的通信格式。网络只负责将这些数据搬运到指定的站点，和指定的user[2:0]通道，此时user[2:0]不能为0。

|  |  |  |
| --- | --- | --- |
| 序列 | 意义 | 备注 |
| 0 | 4’b0 | 无效扩充 |
| 1 | src\_id/des\_id | 数据源ID/目标ID。 |
| 2 | {1'b0，user[5:0]，len[8:8]} | 数据区的数据长度1位。  访问其他DSP通道，user[2:0]一定不能为0。 |
| 3 | len[7:0] | 数据区的数据长度低8位。 |
| 4 |  | DSP与DSP之间的数据 |
| 5 |  |
| …… |  |
| 4+len-1 |  |

1’b1, src[2:0], des[2:0], 1’b1, des[2:0], src[2:0]

## 备注：FPGA端暂不支持delta设为0的模式。(暂不支持)

## 4.3 CH\_ID定义

对于byte[2], 定义为{1'b0，user[5:0]，len[8:8]}。实际上，在数据链路层，定义为

{is\_mail，src\_ch[2:0], des\_ch[2:0]，len[8:8]}。

在发送端，src\_ch为自动填写。des\_ch为应用层填写。

在接收端，系统会根据des\_ch选择分发给不同的对象。

在原有的系统，is\_mail通道由系统自动补充。而且，mail通道只能与mail通道通信，pdu通道只能与pdu通道通信。pdu与mail通道之间不能交互同行。

而由于在FPGA的通道设计之处，就定义。PDU不带应答，而MAIL才能带应答传输。在不同的应用中，发现这个严重制约应用层程序的开发。因此，需要一种机制，可以让FPGA的PDU通道与MAIL通道直接交叉通信，甚至其他的通道也可以交互通信。因此，特做以下扩展。而以下的扩展，兼容以往的定义。

因此，对byte[2]的详细定义为：

在应用层定义的发送端定义为： {1'b0，rvs[2:0], switch, des\_ch[1:0], len[8:8]}

它表示，当switch为1时，访问的pdu，mail通道需要交互。否则保持原有不变。

在数据链路层定义的发送端定义：{is\_mail，1’b0，src\_ch[1:0], switch, des\_ch[1:0], len[8:8]}

它表示该数据是否来自mail的通道，且该数据需要需要switch到pdu或者mail通道。由于描述通道只有两位，因此最多支持的通道数位4个。

在数据链路层定义的接收端定义：{is\_mail，1’b0，src\_ch[1:0], switch, des\_ch[1:0], len[8:8]}

在接收到这个数据后，系统会根据is\_mail和switch，des\_ch信号选择发给对应的通道。转变为。

{1’b0，switch，src\_ch[1:0], 1’b0, des\_ch[1:0], len[8:8]}。

在应用层定义的接收端定义为：{1’b0，switch，src\_ch[1:0], 1’b0, des\_ch[1:0], len[8:8]}。此时应用层直接知道数据的源src\_ch，其数据是否switch了。然后如果程序需要应答则，直接交换得到数据{1’b0，rvs[2:0] switch, des\_ch[1:0], len[8:8]}。

# 第五部分 FPGA网络接口设计

FPGA网络接口包括多个层面，它下行端面向的是网络输入的入口，也即是网络数据链路层；上行端面向的最多8个通道的数据通道，其中通道0为FPGA DMA通道。其他通道为DSP通道。因此，在数据下发情况时，它是一个mux模块。它需要根据优先级将报文发送至网络。

数据优先级分类两大类：PDU数据，Mailbox数据。这两大类数据再按照通道设置的优先级进行传输。

对于接收的数据，它会经过decode模块，然后分别发送到各个通道。数据在发送至通道的时候会附带该信息是PDU数据或者Mailbox数据。然后上次进一步决定如何处理。

## 5.1发送接口设计

它的结构图如下，每一个通道都一个独立的buffer，用于缓存。该通道主要用于传输周期性数据，或者实时性比较高的数据。对于DSP通道，DSP每一次只能先发一个报文，再发下一个报文。

除了各自独有buffer通道，还有一个Mail\_buffer用户发送mailbox数据。该通道被所有通道共享。它的优先级最低，只有当各个通道都没有周期数据后，才开始发送这个数据。

Mux会根据优先级进行仲裁发。当所有报文的优先级一样时，则会采用轮询方式。

Flow\_Ctrl为流量控制模块。主要用于周期性传输，要求PDU数据在要求的时刻达到目的地。那么就需要限定每一个站点的数据流。通过牺牲带宽换取数据通道的畅通。



## 5.2接收接口设计

如下图所示，网络中收到的数据，会经过Decode模块，分发到各个通道的的buffer区。由各个通道去读。Decode的分发会直接发送给buffer，而无视buffer是否空。因此，如果buffer不能及时取走数据，则会丢失数据。报文在传输的时候，各个buffer必须保证报文的完整性。也即是，如果一个存储在buffer中报文正在被读取，则不能将该报文覆盖。如果该报没有被读取，则可以被覆盖。



## 5.3 DSP通道访问寄存器接口

### PDU\_TX\_CTRL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| 15~14 | model | 1 | 只写 | 0=ATUO\_END\_TX。当FPGA处于tx\_ing=0时，只要FPGA接收到len个数据后，自动启动发送  1=WAIT\_ACTIVE。当FPGA处于tx\_ing=0时，并接收到len个报文后，等待active信号，才开始发送报文。  2=WAIT\_SYNC。当FPGA处于tx\_ing=0时，并接收到len个报文后，等待外部sync信号，才开始发送报文。  3=ATUO\_TX。当FPGA处于tx\_ing=0时，只要FPGA开始接收到第一个数据时，就开始启动传输。这个模式必须要求DMA的传输速度必须大于等于网络速度。这个模式只用在要求传输延迟非常小的情况。 |
| 13 | data\_valid | 0 | 只写 | 写1：一个完整报文发已经送至FPGA。 |
| 12 | active |  | 只写 | 用于WAIT\_ACTIVE模式，写1有效，则启动一次发送，将buffer中所有完整的报文都发送出去。直接buffer空才停止。 |
| 11~1 |  |  |  |  |
| 0 | clr |  | 只写 | 写1清除PDU\_TX\_STATUS所有状态位。  同时可以清除本通道的所有buffer数据。只能在PDU传输没有进行的时候。  在以下情况下，该功能有效，将一个PDU报文发送至buffer。在该buffer没有启动传输时，写该指令，可以将该PDU报文清除。需要谨慎使用。 |

### PDU\_TX\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| 15~14 | model | 1 | 只读 | 0=ATUO\_END\_TX。当FPGA处于tx\_ing=0时，只要FPGA接收到len个数据后，自动启动发送  1=WAIT\_ACTIVE。当FPGA处于tx\_ing=0时，并接收到len个报文后，等待active信号，才开始发送报文。  2=WAIT\_SYNC。当FPGA处于tx\_ing=0时，并接收到len个报文后，等待外部sync信号，才开始发送报文。  3=ATUO\_TX。当FPGA处于tx\_ing=0时，只要FPGA开始接收到第一个数据时，就开始启动传输。这个模式必须要求DMA的传输速度必须大于等于网络速度。这个模式只用在要求传输延迟非常小的情况。 |
| 13 | overflow | 0 | 只读 | 1：buffer overflow，clr清除 |
| 12 | underflow | 0 | 只读 | 1：报文往网络上送的时候出现错误 |
| 11 |  |  |  |  |
| 10~0 | bf\_cnt |  | 只读 | 当前buffer的使用状态 |

### MAIL\_TX\_CTRL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| 15~14 |  |  |  |  |
| 13 | data\_valid | 0 | 只写 | 写1：一个完整报文发已经送至FPGA。 |
| 12 |  |  |  |  |
| 11 | mail\_req |  | 只写 | 写1表示，请求发送一次mailbox数据。 |
| 10~1 |  |  |  |  |
| 0 | clr |  | 只写 | 写1清除所有MAIL\_TX\_STATUS状态位。  同时可以清除本通道的所有buffer数据。只能在PDU传输没有进行的时候。  在以下情况下，该功能有效，将一个PDU报文发送至buffer。在该buffer没有启动传输时，写该指令，可以将该PDU报文清除。需要谨慎使用。 |

### MAIL\_TX\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| ~~15~14~~ | ~~rx\_mail\_bus\_arb~~ |  |  | ~~当前rx mailbox总线请求者~~ |
| 13 | overflow | 0 | 只读 | 1：buffer overflow，clr清除 |
| 12 | underflow | 0 | 只读 | 1：报文往网络上送的时候出现错误 |
| 11 | mail\_ready | 0 | 只读 | 1：获得mailbox权限，可以PFGA发送收。 |
| 10 | mail\_over | 1 | 只读 | 1：该报文已经发送至网络。 |
| 9 | mail\_req\_ing |  | 只读 | 1：当前正在获取等待总等待线，或者正在发送数据 |
| ~~8~7~~ | ~~tx\_mail\_bus\_arb~~ |  | ~~只读~~ | ~~当前mailbox总线所有者~~ |
| ~~6~5~~ | ~~tx\_bus\_arb~~ |  | ~~只读~~ | ~~当前bus总线所有者~~ |
| 4 |  |  |  | 保留 |
| ~~3~0~~ | ~~rx\_bus\_arb~~ |  |  | ~~当前rx bus的总线请求者。当rx\_bus\_arb最高位为1，是表示当前请求的是mailbox。否则为pdu。正常情况下，rx\_bus\_arb[2]恒为0。~~ |

### PDU\_RX\_CTRL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| 15~14 |  |  |  |  |
| 13 |  |  |  |  |
| 12 |  |  |  |  |
| 11~2 |  |  |  |  |
| 1 | rd\_over |  | 只写 | 写1表示完成1次完整报文的读取。该位已无效 |
| 0 | clr |  | 只写 | 写1清除所有状态位，另外，如果buffer中有**残余**报文，则同时清除该报文。（该报文仅被读一部分） |

### PDU\_RX\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| 15~11 | packet\_cnt |  | 只读 | 当前buffer中有完整的完整报文个数 |
| 10~0 | bf\_cnt |  | 只读 | 当前buffer的使用状态 |

### MAIL\_RX\_CTRL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| 15~14 |  |  |  |  |
| 13 |  |  |  |  |
| 12 |  |  |  |  |
| 11~2 |  |  |  |  |
| 1 | rd\_over |  | 只写 | 写1表示完成1次完整报文的读取。此时mailbox的访问权才会被释放。该位已无效，当读取的个数达到，则访问权自动释放。 |
| 0 | clr |  | 只写 | 写1清除所有状态位，另外，如果buffer中有**残余**报文，则同时清除该报文。（该报文仅被读一部分） |

### MAIL\_RX\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写性 | 备注 |
| 15~11 |  |  |  |  |
| 10~1 | dll\_rx\_num[9:0] |  |  | 当前dll上的缓冲数据个数（short），最高位为1表示已满 |
| 0 | rx\_mail\_valid |  |  | 指示是否有报文等待接收 |

## 5.3 数据通信关于user数据的定义

在发送的时候，发起端不需要填写user[5:3]，但是需要填写user[2:0]。user[2:0]用于指示访问的对应站点的通道ID。

对于user[5:3]，程序会自动填写。程序最多支持7个通道。

而由于支持7路。因此，user[5:0]不可能出现全1情况。因此，最高位1’b0,会被使用如果访问的是PDU通道，最高位依旧会保持1’b0，而访问MAIL通道时，该值会被赋值为1’b1。

## 5.4 Common Net Status寄存器

详细的功能寄存器地址列表如下.

|  |  |  |
| --- | --- | --- |
| Addr | Register | Comment |
| 0x0 | PORT\_STATUS |  |
| 0x1 | NET\_RUN\_STATUS |  |
| 0x2 | SYNC\_CYCLE |  |
| 0x3 | ONLINE\_MSG |  |
| 0x4 | CRC\_OK\_CNT\_A[15:0] |  |
| 0x5 | CRC\_OK\_CNT\_A[31:16] |  |
| 0x6 | CRC\_ERR\_CNT\_A[15:0] |  |
| 0x7 | STATION\_MSG | 描述应用层 |
| 0x8 | CRC\_OK\_CNT\_B[15:0] |  |
| 0x9 | CRC\_OK\_CNT\_B[31:16] |  |
| 0xA | CRC\_ERR\_CNT\_B[15:0] |  |
| 0xB | STATION\_ID |  |
| 0xC | DMA\_MAIN\_CTRL\_STATUS | 对于FPGA DMA的配置，有一个CFG RAM存储配置信息。它的配置通过0x11和0x12实现。详细的见《网络模块对外接口设计.doc》 |
| 0xD | DMA\_CFG\_SET\_ADDR |
| 0xE | DMA\_CFG\_OP\_DATA |
| 0xF | NET\_VERSION | 网络版本信息 |
| 0x10 | MODAL\_CTRL | 模态切换 |
| 0x11 | SYS\_TIME\_LOW |
| 0x12 | SYS\_TIME\_HIGH |
| 0x13 | FAST\_NEXT |
| 0x14 | NORMAL\_NEXT |
| 0x15 | CHANGE\_TIME\_L |
| 0x16 | CHANGE\_TIME\_H |
| 0x17 | MODAL\_NOW |
| ~~0x18~~ | ~~FMMU\_ADDR~~ | ~~（byte addr）用于实现1个报文多个站接收~~ |
| ~~0x19~~ | ~~FMMU\_LEN~~ | ~~（byte len）~~ |
|  |  |  |
| 0x1C | phy\_tx\_busy\_cnt\_a | portA在一个周期内出于忙状态计算 |
| 0x1D | phy\_tx\_busy\_cnt\_b | portB在一个周期内出于忙状态计算 |
|  |  |  |
| 0x1F | INTERRUP\_STATUS |  |
| 0x20 | PDU\_TX\_CTRL/ PDU\_TX\_STATUS | 该寄存器在不同的通道会不同一样。pci和dsp采用 |
| 0x21 | MAIL\_TX\_CTRL/MAIL\_TX\_STATUS |
| 0x22 | PDU\_RX\_CTRL/ PDU\_RX\_STATUS |
| 0x23 | MAIL\_RX\_CTRL/MAIL\_RX\_STATUS |
|  |  |  |
| 0x30 | TAP0 | 测试变量 |
| 0x31 | TAP1 | 测试变量 |
| 0x32 | TAP2 |  |
| 0x33 | TAP3 |  |
| 0x34 | TAP4 |  |
| 0x35 | TAP5 |  |
|  |  |  |

### PORT\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bits | 变量 | 读写性 | 复位值 | 备注 |
| 15~8 | port\_status\_b, | 只读 | 0 | portB的状态  status[0] = link\_up; 链路通道1：连接；0：断开  status[1] = OFFSET\_VALID;1：偏移量OK  status[3:2] = clock\_speed; 2'b10 : 千兆；2'b01:百兆；2‘b11:485  status[4] = duplex\_status;1：双工；0：单工  status[5] = link\_status;物理通道1：连接0：断开  status[6]=overflow,数据溢出标志,该值需要clr才能清除。  status[7:7] = watch\_link; |
| 7~0 | port\_status\_a | 只读 | 0 | portA的状态  status[0] = link\_up; 链路通道1：连接；0：断开  status[1] = OFFSET\_VALID;1：偏移量OK  status[3:2] = clock\_speed; 2'b10 : 千兆；2'b01:百兆；2‘b11:485  status[4] = duplex\_status;1：双工；0：单工  status[5] = link\_status;物理通道1：连接0：断开  status[6]=overflow,数据溢出标志。该值需要clr才能清除。  status[7:7] = watch\_link; |

### NET\_RUN\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15 | master\_flag |  | 只读 | 1：表示该站点为主站，否则为从站 |
| 14 | station\_ready |  | 读写 | 该值仅本地可写。上电默认为0，表示本地没有准备好。  本地在所有初始化完成后，该值置1。 |
| 13 | mac\_tx\_packet\_err\_b |  | 只读 | 1:mac port\_b发生过”组包"错误情况，clr清除 |
| 12 | mac\_tx\_underflow\_b |  | 只读 | 1:macport\_b发生过读取fifo underflow错误情况，clr清除 |
| 11 | mac\_tx\_overflow\_b |  | 只读 | 1:macport\_b发生过读取fifo oveflow错误情况，clr清除 |
| 10 | rollback\_valid |  | 读写 | 该值仅本地可写。上电默认为0，表示rollback的数据会被消亡  1：表示rollback的user数据会被保留。 |
| 9 |  |  |  |  |
| 8 | app\_rst |  | 只写 | 1：复位应用层程序。仅在本地使用。主要用于本地异常操作导致网络应用层的总线总裁进入死锁。或者其他异常进入程序死锁时，进行本地复位。 |
| 7 | offset\_calc\_rst |  | 只写 | 清除并重新计算本地的port offset和delay参数，主站和从站都可写。 |
| 6 | local\_id\_src | 0 | 读写 | 该值仅能在本地修改，且在网络没有构建前。0自动分配，1本地分配 |
| 5 | mac\_tx\_packet\_err\_a |  | 只读 | 1:mac port\_a发生过”组包"错误情况，clr清除 |
| 4 | mac\_tx\_underflow\_a |  | 只读 | 1:macport\_a发生过读取fifo underflow错误情况，clr清除 |
| 3 | mac\_tx\_overflow\_a |  | 只读 | 1:macport\_a发生过读取fifo oveflow错误情况，clr清除 |
| 2 | sync\_on |  | 读写 | 该值只有主站可以操作有效，其他站点修改该值无意义。主站写1使能系统的同步功能。  1：同步正常  0：没有同步 |
| 1~0 | dc\_status |  | 只读 | 00：关闭状态  01：正常状态  10：偏差过大  11：长期错误 |

### SYNC\_CYCLE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15:0 | global\_sync\_cycle | 125\*16 |  | 只能主站可以修改，其他站点只读。 |

### ONLINE\_MSG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15~8 | device\_num |  | 只读 | 线上站点个数（包括自己）。 |
| 7:0 | device\_id |  | 读写 | 当local\_id\_src为自动分配时，该值自动分配值，否则该值可写。  当自动分配时，auto\_id = 8'hf0;表示还没有分配。 |

### CRC\_OK\_CNT\_A[15:0]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bits | 变量 | 读写 | 复位 | 备注 |
| 15~0 | crc\_ok\_cnt\_a[15:0] |  |  | 内部不保证crc\_ok\_cnt\_a[15:0]和  crc\_ok\_cnt\_a[31:16]的一致性。 |

### CRC\_OK\_CNT\_A[31:16]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bits | 变量 | 读写 | 复位 | 备注 |
| 15~0 | crc\_ok\_cnt\_a[31:16] |  |  | 内部不保证crc\_ok\_cnt\_a[15:0]和  crc\_ok\_cnt\_a[31:16]的一致性。 |

### CRC\_ERR\_CNT\_A[15:0]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bits | 变量 | 读写 | 复位 | 备注 |
| 15~0 | crc\_err\_cnt\_a[15:0] |  |  | 该值记录上电后的错误累计，不能清除 |

### STATION\_MSG[15:0]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bits | 变量 | 读写 | 复位 | 备注 |
| 15~0 | station\_msg | 只读 |  | 用于描述应用层，由用户决定。  通过读取该值，可以知道应用层的作用。 |

### CRC\_OK\_CNT\_B[15:0]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bits | 变量 | 读写 | 复位值 | 备注 |
| 15~0 | crc\_ok\_cnt\_b[15:0] |  |  | 内部不保证crc\_ok\_cnt\_b[15:0]和  crc\_ok\_cnt\_b[31:16]的一致性。 |

### CRC\_OK\_CNT\_B[31:16]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bits | 变量 | 读写 | 复位值 | 备注 |
| 15~0 | crc\_ok\_cnt\_b[31:16] |  |  | 内部不保证crc\_ok\_cnt\_b[15:0]和  crc\_ok\_cnt\_b[31:16]的一致性。 |

### CRC\_ERR\_CNT\_B[15:0]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bits | 变量 | 读写 | 复位 | 备注 |
| 15~0 | crc\_err\_cnt\_b[15:0] |  |  | 该值记录上电后的错误累计，不能清除 |

### STATION\_ID[15:0]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bits | 变量 | 读写 | 复位 | 备注 |
| 15~8 | set\_id | 只读 |  |  |
| 7~0 | auto\_id | 只读 |  |  |

### 5.4.1 FPGA DMA专用寄存器定义

#### DMA\_MAIN\_CTRL\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15~8 | pdu\_num | 1 |  | 表示每一个周期需要发送的PDU报文个数  当remote\_cfg\_en = 1时，该值可读可写，否者该值仅可读。 |
| 7 | pdu\_rx\_map\_model | 1 |  | 1：表示解析map存储数据按照纯模式，  0：表示解析map存储数据按照地址+长度的模式 |
| 6 | pdu\_tx\_map\_model | 1 |  | 1：表示解析map存储数据按照纯模式，  0：表示解析map存储数据按照地址+长度的模式 |
| 5 | pdu\_packet\_size\_zero | 0 | 只读 | 1：有请求发送PDU报文数据长度为0，该报文不会被发送。该值会被锁存。写clr |
| 4 | pdu\_check\_sum\_err | 0 | 只读 | 1：有请求发送PDU报文数据指令校验和错误，该报文不会被发送。该值会被锁存。写clr |
| 3 | pdu\_active\_err | 0 | 只读 | 1：表示上一个周期DMA每一执行完成，又发起新的请求。该值会被锁存。在clr写1清除。 |
| 2 | clr | 0 |  | 当remote\_cfg\_en = 1,该值可读可写，否者仅可读。该值读为0，自清除。 |
| 1 | remote\_cfg\_en | 1 |  | 该值仅本地配置 |
| 0 | pdu\_en | 0 |  | 当remote\_cfg\_en = 1时，该值可读可写，否者该值仅可读。 |

#### DMA\_CFG\_SET\_ADDR

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15~0 | cfg\_addr | 0 | 读写 | 当需要访问dma的cfg区时，该值未设定的地址初始值。而每一次  操作DMA\_CFG\_OP\_DATA时，该地址会自动加1。 |

#### DMA\_CFG\_OP\_DATA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15~0 | cfg\_data |  | 读写 | 每一次读或者写操作DMA\_CFG\_OP\_DATA时，cfg\_addr会自加1。 |

备注:FPGA DMA寄存器的DMA\_CFG\_SET\_ADDR和DMA\_CFG\_OP\_DATA只有在remote\_cfg\_en为0时才有效。而对于具有处理器的站点。可以通过控制remote\_cfg\_en，进一步修改cfg中的数据。

这些变量的修改可以是DSP或者PCI或者其他通道。但是对于特定的系统，它是固定的。

#### CFG RAM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 0~255 | pdu\_tx\_cfg | 0 |  | 该内容用于配置各个FPGA DMA需要发的数据，  详见2.3报文发送配置区定义 |
| 256~511 | pdu\_mapping |  |  | 所有pdu接收和发送都是间接寻址，此处存映射表。  详见2.2 三片存储区域（short 地址） |

### 5.4.2 同步模态专用寄存器定义

模态切换是网络向外提供的一个16bit的寄存器。利用该模块，可以让整个系统同步进入某一指定状态。这个16bit寄存器为instr寄存器。

切换控制有两类：本地直接配置切换和远程配置切换。一般切换有一个发起方，发起方需要将该配置为本地模式，而其他相应方则配置为远程模式。一般发起方都为名义上的主站，而其他则为名义上的从站。

切换过程有两种方式，快速方式。它是直接修改instr相关寄存器，然后在当下一个计算同步时钟（该中断不一定等同于通信同步时钟），instr生效。

另一种切换模式，正常切换模式。这种切换模式用于保证同步的准确性和可靠性，详见《基于网络同步的多轴同步系统设计》

#### MODAL\_CTRL\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15~9 |  |  |  |  |
| 10 | lock\_type\_mask |  |  | 当该值为1，lock\_type才会被修改 |
| 9 | lock\_type | 1 | 读写 | 0:锁存的sys\_time为  0：LOCK\_NOW 该模式下，锁存的是当前的系统时间  1：LOCK\_SYNC 该模式下，锁存的是前一次有sync触发的系统时间，该模式用于计算合理的CHANGE\_TIME |
| 8 | lock\_sys\_time |  | 只写 | 写1锁存当前系统时间，存于sys\_time. |
| 7 | change\_stop |  | 只写 | 当采用normal模式时，且status出于WILL\_CHANGE或者WAIT\_TIME时都可以停止切换，其他时刻无效。写1有效，该值自动清除。 |
| 6~4 | state |  | 只读 | 切换状态：  IDLE = 0, WILL\_CHANGE = 1,  WAIT\_TIME = 2, NORMAL\_WAIT\_SYNC = 3,  FAST\_WAIT\_SYNC = 4 STOP = 5, OVER = 6; |
| 3 | change\_en\_mask |  | 只写 | 当该值为1，change\_en才会被修改 |
| 2 | change\_en | 1 | 读写 | 当该值为1，模态切换才会有效。当state处于IDLE才有效。它不能停止切换 |
| 1 | ctrl\_src\_mask |  | 只写 | 该值仅本地配置，只有该值为1时，ctrl\_src才会被修改 |
| 0 | ctrl\_src | 1 | 读写 | 该值仅本地配置。0：表示所有状态由本地址控制，1：表示所有状态由远程控制。 |

#### SYS\_TIME[15:0]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15~0 | sys\_time[15:0] |  | 只读 |  |

#### SYS\_TIME[31:16]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15~0 | sys\_time[31:16] |  | 只读 |  |

#### FAST\_NEXT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15:0 | fast\_next | 0 | 只写 | 修改该值后，在下一个计算同步时钟时，modal\_now= modal\_fast\_next |

#### NORMAL\_NEXT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15:0 | normal\_next | 0 | 只写 | 当status出于IDLE状态时，修改该值则进入WILL\_CHANGE状态 |

#### CHANGE\_TIME[15:0]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15:0 | change\_time[15:0] | 0 | 读写 |  |

#### CHANGE\_TIME[31:16]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15:0 | change\_time[31:16] | 0 | 读写 | 当status出于WILL\_CHANGE状态时，修改该值则进入WAIT\_TIME状态 |

#### MODAL\_NOW

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15:0 | modal\_now | 0 | 只读 | 当前模态 |

### 5.5 测试信号专用寄存器

#### TAP0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15 | TAP\_EN |  |  | 当该值为1时，以下数据有效 |
| 14 | dma\_tx\_mail\_bus\_req\_ing |  |  | fpga mail正在申请mail总线 CH0 |
| 13 | pci\_tx\_mail\_bus\_req\_ing |  |  | pci mail正在申请mail总线 CH1 |
| 12 | dsp0\_tx\_mail\_bus\_req\_ing |  |  | dsp0 mail正在申请mail总线 CH2 |
| 11~9 | tp\_mail\_rx\_status |  |  | fpga mail 接收状态机 |
| 8~6 | tp\_mail\_tx\_status |  |  | fpga mail发送状态机 |
| 5~3 | tp\_dma\_pdu\_rx\_status |  |  | fpga mail 接收状态机 |
| 2~0 | tp\_dma\_pdu\_tx\_status |  |  | fpga mail 发送状态机 |

#### TAP1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15~12 | rx\_bus\_mail\_arb\_src |  |  | 当前接收MAIL总线归属权{CH2，CH1，CH0 } |
| 11~8 | rx\_bus\_arb\_src |  |  | 当前接收总线归属权 { MAIL, CH2，CH1，CH0 } |
| 7~4 | tx\_bus\_mail\_arb\_src |  |  | 当前发送MAIL总线归属权{CH2，CH1，CH1 } |
| 3~0 | tx\_bus\_arb\_src |  |  | 当前发送总线归属权 {MAIL ，CH2，CH1，CH0 } |

#### TAP2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15 | dll\_wr\_ready\_lock |  |  | 当前dll是否允许接收用户数据用于发送 |
|  |  |  |  |  |
|  |  |  |  |  |
| 10~0 | tp\_dll\_tx\_fifo\_num\_a |  |  | 当前portA 用户数据dll tx fifo的使用情况，按byte计算 |

#### TAP3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 10~0 | tp\_dll\_tx\_fifo\_num\_b |  |  | 当前portB 用户数据dll tx fifo的使用情况，按byte计算 |

#### TAP4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
|  |  |  |  |  |
| 9~0 | dll\_rd\_num\_lock |  |  | 当前dll上的缓冲数据个数（short），最高位为1表示已满 |

#### TAP5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit | 命名 | 初值 | 读写 | 备注 |
| 15~12 | tp\_dll\_rx\_err\_cnt |  |  | 从上电开始，发送用户数据解析错误次数，最大为15. |
| 11~8 | tp\_dll\_flush\_cnt |  |  | 发生数据拥塞导致数据丢失次数，最大值为15 |
| 7~0 | tp\_dll\_flush\_flag\_lock |  |  | {1’b0, CH2\_MAIL, CH1\_MAIL, CH0\_MAIL, 1’b0, CH2\_PDU, CH1\_PDU, CH0\_PDU}发生数据拥塞导致丢失发生情况 |

# 第六部分 工作时间

详细的工作模块要求：

1、实现第五部分的网络接口模型。同时给出和DSP通信的寄存器工作寄存器

2、实现第三部分的应用协议接口。

规划时间为1个月。