TOE 概要设计
_V1.0_20171222

文件状态	当前版本	*V1.0*
[√] 草稿 [ ] 正式发布	作　　者	马燕涛
	完成日期	
	文档模板	
	密　　级	*内部使用*

分发列表

变更历史

版本	完成日期	变更记录	作者	审核	批准
V1.0	20171222	初稿	马燕涛 李明		

目录

1 系统简介

TCP/IP 协议族作为 Internet 网络协议的实际标准，已经得到广泛应用。在目前的以太网网络通信系统中，多使用软件方式处理 TCPIP 协议栈，而协议栈的处理需要消耗大量的 CPU 资源，尤其在协议报文、链接维护、数据复制和中断处理等方面。

随着数字货币和区块链技术的发展，交易频度也在不断增长，交易数据封装的以太网报文处理速度逐步成为高性能区块链的传输瓶颈。为提高以太网交易报文处理速度，减少以太网通信报文处理中对 CPU 的占用，近年来在网络处理加速领域开展了大量研究工作，其中 TOE 技术是发展较成熟、前景较广阔的一种技术。它将 TCP/IP 协议族的部分或者全部功能转到专门的硬件电路上实现，通过驱动程序与原有操作系统无缝对接，而且能做到与原有网络应用程序完全兼容，能够从系统数据复制、中断处理和协议处理等三个方面减少系统 CPU 的占用。

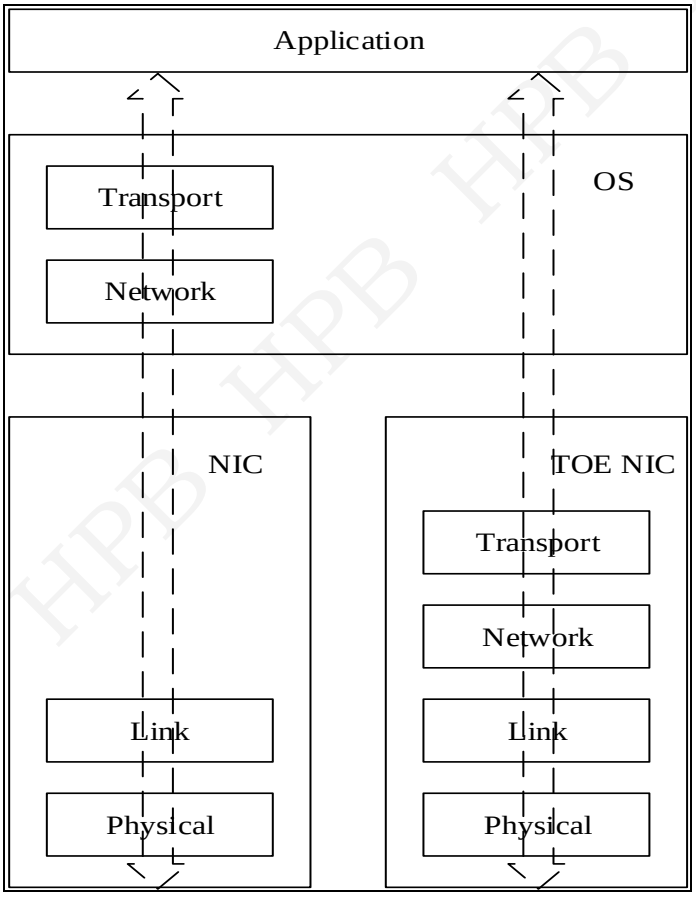

图 1 传统网卡与 TOE 网卡对比

本设计在 FPGA 内实现了一个完整的 TCP/IP 协议栈。包括 TCP 会话的建立、拆除、控制与维护，TCP 数据发送和接收，UDP 数据发送和接收。还包括一些基本辅助功能，如 IP 协议的处理，ARP、ICMP 协议的处理，并在 UDP 协议上实现了 DHCP 客户端功能。

系统用户侧提供 10G 以太网光接口，处理器接口采用高速 PCIE 接口，外挂 2 组 64 位宽 1.2GHz 速率的 DDR4，基本能够满足百万数量级的 TPS 带宽要求。

2 特性列表

- 网络侧使用万兆以太网光口，实现以太网报文线速处理能力
- 用户侧使用 PCIE Gen2 ×8 接口
  - 最高速率可达 5G×8=40Gbps
  - 支持 SG-DMA 功能，效率可达 88%
  - 提供 AXI-MM 接口进行高速数据传输
  - 提供 AXI-Lite 接口对加速芯片寄存器进行访问
- 数据链路层
  - DMAC 检查
  - CRC 校验检查
  - 报文长度检查
  - 报文统计
- 网络层
  - 内嵌 ARP SERVER，处理 ARP 请求报文并维护 ARP 表
  - 内嵌 ICMP SERVER，处理 PING 请求报文
  - IP 报文处理
    - 报文成帧与解帧
    - 地址检查
    - 报文长度检查
    - 报文校验生成及检查
    - 报文类型分类处理
- 传输层
  - UDP 报文处理
    - 报文校验生成与检查
    - 报文长度检查
  - TCP 报文处理
    - 实现 10K 条 TCP 连接的管理与实时处理
    - 实现 RFC 793 标准链路状态机
    - 实现滑动窗口数据传输机制
    - 实现超时重传机制
    - 实现乱序报文的处理
    - 实现 ACK 延迟发送机制
    - 实现 RFC 896 中建议的 Nagle 算法
    - 实现慢启动及拥塞控制算法
    - 实现保活机制
- 应用层
  - 具有基本 DHCP 客户端功能，可接受自动分配 IP 地址

3 总体设计

3.1 系统框图

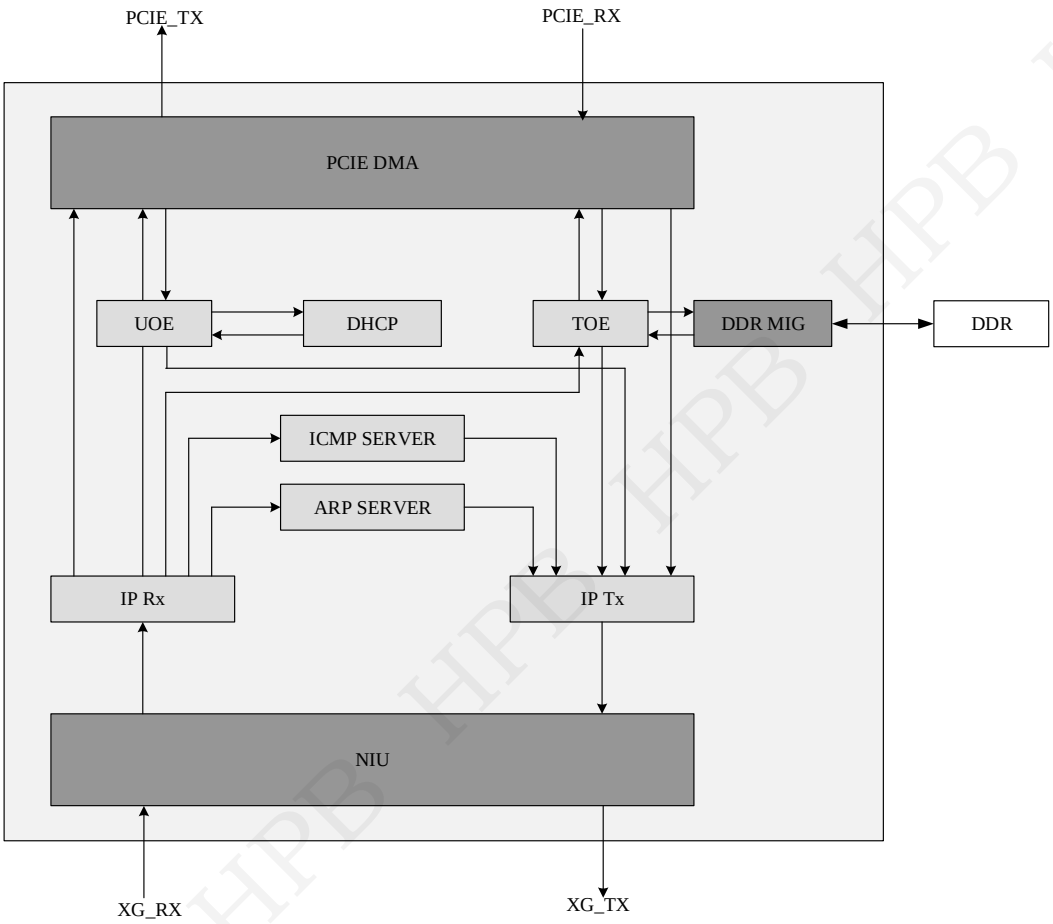

图 3-1 TOE 系统框图

3.2 功能概述

本系统实现了万兆 TOE 以太网卡的万兆功能，从万兆以太网光接口到 TCP/IP 协议栈处理，再到 PCIE 应用层用户接口，如图 3-1 所示。

NIU（网络接口单元）实现了万兆以太网接口的物理层和数据链路层。IP 处理模块则实现网络接口接收报文解析处理，通过以太网报文头部区分协议类型，可解析 ARP、ICMP、TCP/IP 和 UDP/IP 等协议报文，其它报文类型直接通过用户接口送到 PCIE DMA 缓存接口，上报软件处理。另外为支持地址解析协议，内置了 ARP Server，可以通过硬件进行 MAC 地址解析。还内置了 ICMP Server，能够回显 ICMP 请求或产生目的地址不可达等告警信息。UOE 模块处理所有 UDP 报文并将其送入 PCIE DMA 缓存，并触发 DMA 将 UDP 报文传输到应用层。TOE 模块实现 TCP 连接的建立、维护、状态切换和拆除等动作，并将 TCP 报文缓存到片外 DDR RAM 中，通过用户接口连接 PCIE DMA 接口。

3.3 模块说明

3.3.1 网络接口单元 NIU

NIU 模块（网络接口单元）包括 Ten_gig_eth_pcs_pma、 Ten_gig_eth_mac、rx_interface 三个模块，其中前两个为 Xilinx 的 IP，分别实现万兆光接口物理层和以太网 MAC 子层。

实现框图如下：

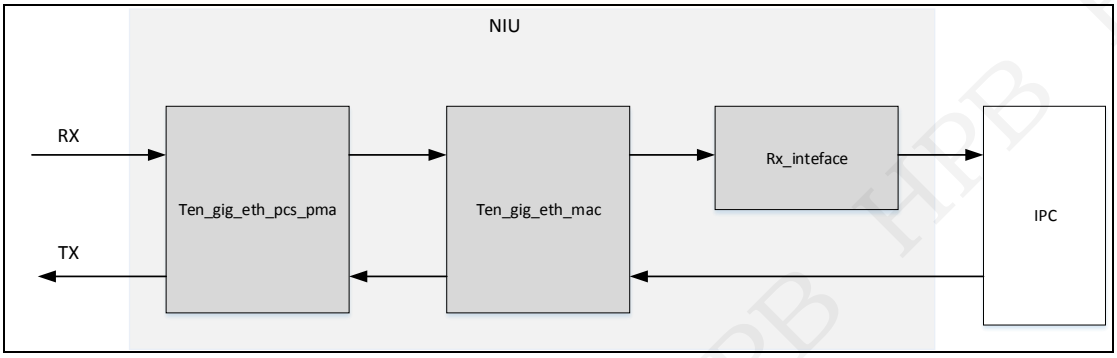

图 3-2 NIU 框图

通过 Vivado 软件产生 Ten_gig_eth_pcs_pma IP 时，可通过选项设置为 10G 以太网光口模式，网络接口线路速率为 10.3125Gbps，采用 64/66 编码。芯片内部接口为 156.25MHz×64bit 总线，有效带宽为 10Gbps。

Ten_gig_eth_pcs_pma 发送方向，使用本地参考时钟将用户数据进行线路编码，填充线路开销及空闲码。然后通过 P2S 模块将数据进行串化，通过差分接口发出。可通过 IP 界面设置发送信号预加重、信号电平幅度等物理层参数，调整发送信号质量，保证传输。

Ten_gig_eth_pcs_pma 接收方向，将光接口收到的差分信号进行时钟恢复，使用恢复时钟采样接收数据，对串行线路码进行解码和块定位，确定线路编码块的边界，然后进行串并处理，恢复并行数据。接收方向同样可以设置物理参数，包括去加重、内置匹配电阻等，以改善线路传输效果。当线路传输出现故障或误码，可通过线路编码校验开销进行检查，并将故障情况送出到用户接口，用户可以根据故障情况进行线路质量监控和排查。

Ten_gig_eth_mac 发送方向，接来自 IPC 模块的报文，进行 FCS32 的计算与插入。然后将报文转换为 XGMII 格式发送给 Ten_gig_eth_pcs_pma 模块。

Ten_gig_eth_mac 接收方向，接收报文来自 Ten_gig_eth_pcs_pma，为 XGMII 格式，从中提取出以太网报文，并进行长度检查、FCS 校验，进行报文统计。然后将报文送给 Rx_interface 进行处理。

Rx_interface 接收来自 Ten_gig_eth_mac 的报文，查找以太网报文头部，从以太网报文中对应位置取出 MAC 地址，然后比较目的 MAC 地址与本地 MAC 地址是否一致，如果一致则送给 IPC 进行后续处理，否则进行丢弃。

3.3.2 IPC

实现以太网协议类型处理、IP 协议处理、IP 协议类型处理，并将处理后的报文分流。

从 NIU 模块接收到校验后的以太网报文后，根据以太网报文中的协议类型和 IP 报文中的协议类型进行分类。本系统支持对 ARP、ICMP、UDP/IP 和 TCP/IP 等协议的处理，内置 ARP 服务器和 ICMP 服务器，对 ARP 请求报文和 ICMP 请求报文通过硬件直接进行回应，其它协议报文直接输出到 PCIE 缓存接口交给软件处理。

ARP 协议是地址解析协议，可以根据 IP 地址查找 MAC 地址。本地需要维护一张 MAC 地址表，存放 IP 地址和 MAC 地址的对应关系。从本机给某个 IP 地址发送以太网报文时，需要从 MAC 地址表中查找其 MAC 地址才能封装以太网报文并发送，如果表中没有对应表项，需要从本地向网络发出 ARP 请求广播报文，该 IP 地址收到 ARP 报文后发出 ARP 响应报文，报文中携带该设备的 MAC 地址，本机收到后根据此 MAC 地址封装以太网报文实现通信，并在 MAC 地址表中增加一条表项。

ICMP 协议是互联网控制报文协议，用来传递控制信息，控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。

IP 协议位于网络层，是一种无连接协议，不保证数据的正确传递。IP 帧结构如下图所示：

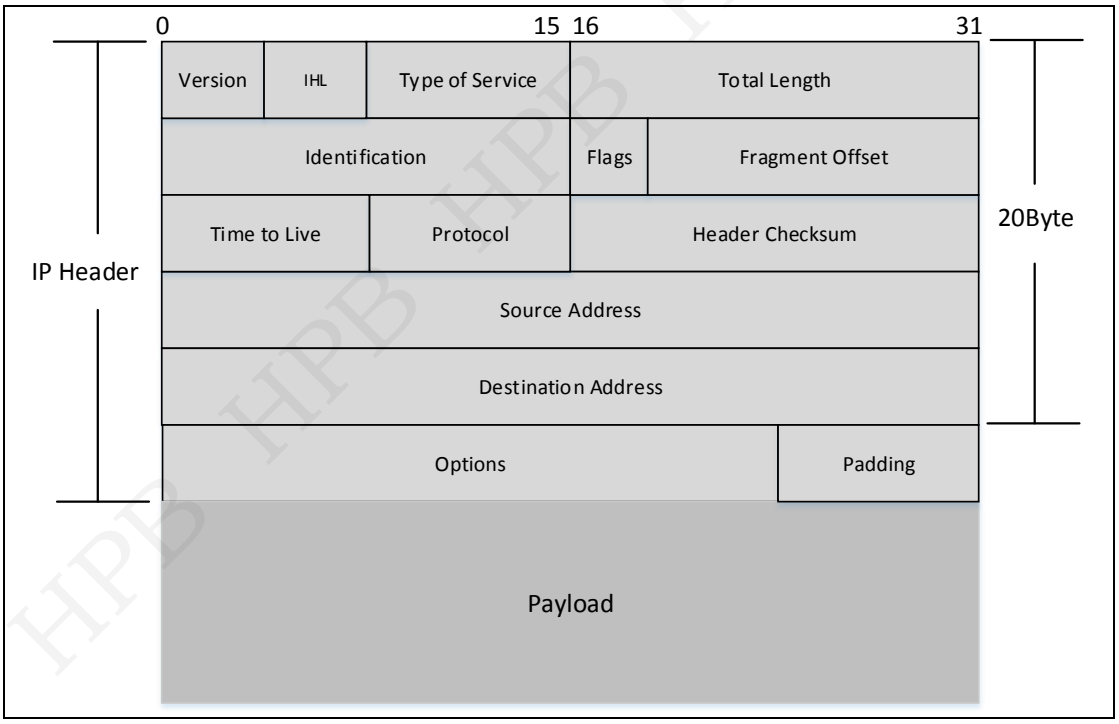

图 3-3 IP 报文格式

IPC 根据高层协议类型分别将 TCP 报文和 UDP 报文送到 TOE 模块和 UOE 模块进行处理。

3.3.3 TOE

TCP（Transmission Control Protocol 传输控制协议）是一种面向连接的、可靠的、基于字节流的传输层通信协议，其规范为 IETF 的 RFC 793。

在因特网协议族（Internet protocol suite）中，TCP 层是位于 IP 层之上，应用层之下的中间层。为不同主机的应用层之间提供可靠的、像管道一样的连接。

其基本原理如下:为了保证可靠传输，发送端给每个包一个序号，序号保证了接收端可以按序接收。然后接收端对已成功收到的包发回一个相应的确认（ACK）；如果发送端在合理的往返时延（RTT）内未收到确认，那么对应的数据包就被假设为已丢失，将会被进行重传。TCP 用一个校验和函数来检验数据是否有错误；在发送和接收时都要计算校验和。

其帧结构如下图所示

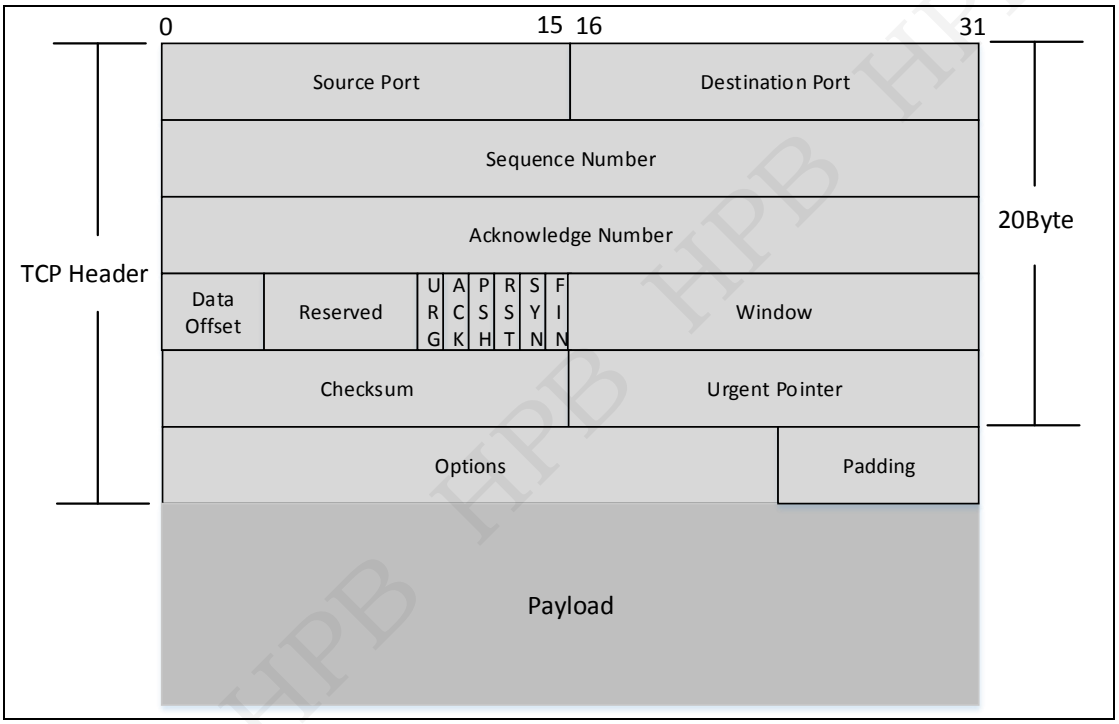

图 3-4 TCP 报文格式

为了提供一个可靠的连接通道，TCP 实现了多种控制机制，是一个非常复杂的协议，略述如下：

- 为实现连接可靠的建立与终止，采用了三次握手建立连接，四次握手终止连接
- 为实现可靠传输，采用了确认机制、超时重传，乱序处理、
- 为实现高效传输，采用了滑动窗口机制
- 为实现拥塞控制，采用了慢启动、拥塞避免、快速恢复等算法
- 为减少小分组报文的产生，采用了数据捎带 ACK、Nagle 等方法
- 为增强可靠性，提供了窗口探查机制和保活机制

传统上，都是用软件的方式来处理 TCP 协议，这在网络负载较轻的时候运转良好。但近年来，随着网络负载越来越高，用软件来处理 TCP 给 CPU 造成了很大的负担。这时候，产生了一种基于硬件处理 TCP 协议的技术，即 TOE（TCP OFFLOAD ENGINE）

TOE 的主要功能就是将原来由软件处理的 TCP 和 UDP 协议从软件协议栈剥离，用硬件来实现，比如 FPGA 和 ASIC 等。

在 TOE 实现过程中，会遇到一系列的设计挑战

- 复杂的状态控制：连接控制、拥塞控制、报文数量控制
- 复杂的数据处理：分段、聚合、确认机制、重传机制
- 10K+连接状态维护
- 多单元之间的异步通信

其实现原理框图如下：

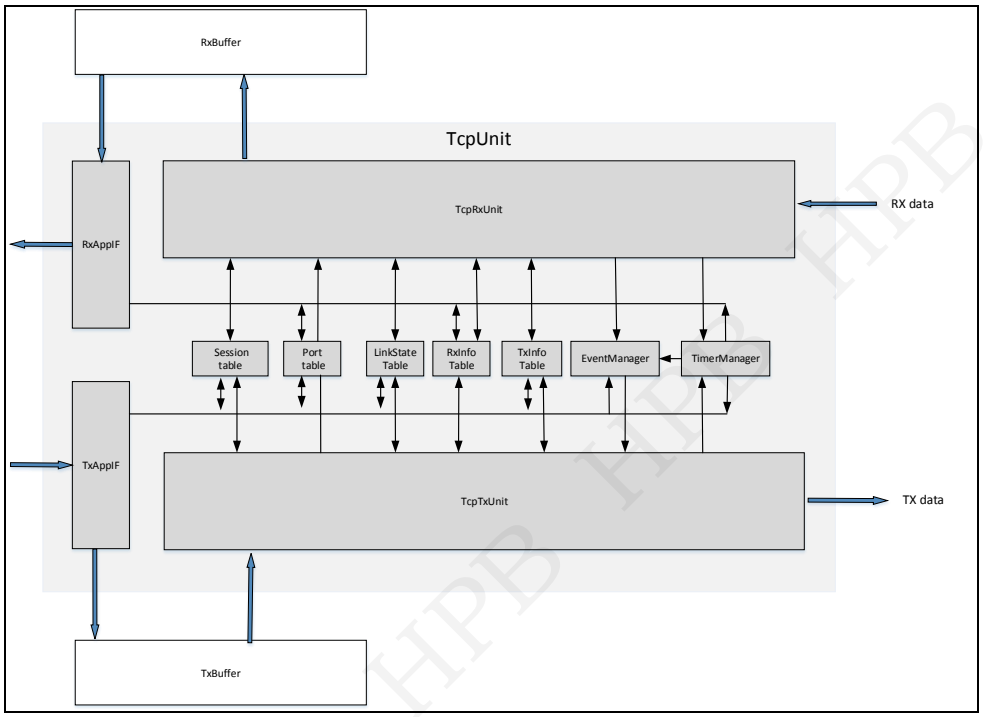

图 3-5 TOE 系统框图

TcpTxUnit、TcpRxUnit 分别实现了主要的报文发送功能和报文接收功能，TxAppIF、RxAppIF 负责和应用程序接口。在收发单元之间是共用的数据表项，列举如下：

- 会话表：实现“IP 地址+端口号”到内部编址的转换
- 端口表：存储每个端口的配置和状态信息
- 链路状态表：存储每个连接的状态机（由 RFC 793 定义）
- 接收信息表：存储每个连接的接收控制信息，如接收序号、应用层读取指针
- 发送信息表：存储每个连接的发送控制信息，包括接收到的反向控制信息、本地发送控制信息
- 事件管理器：负责维护给发送侧的事件
- 定时管理器：负责维护一系列的计数器

收发单元通过上述的表项进行读取和处理，来实现 TCP 定义的各种功能。

3.3.4 UOE

UDP （User Datagram Protocol，数据报协议），是一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务，其规范为 IETF 的 RFC 768。

和 TCP 相比，UDP 不属于连接型协议，不提供可靠性保证、顺序保证和流量控制字段等，可靠性较差。但 UDP 资源消耗小，处理速度快，传输延迟小，传输效率高，所以通常音频、视频等数据在传送时使用 UDP 较多，因为即使偶尔丢失一两个数据包，也不会对接收结果产生太大影响。

UDP 报文格式如下：

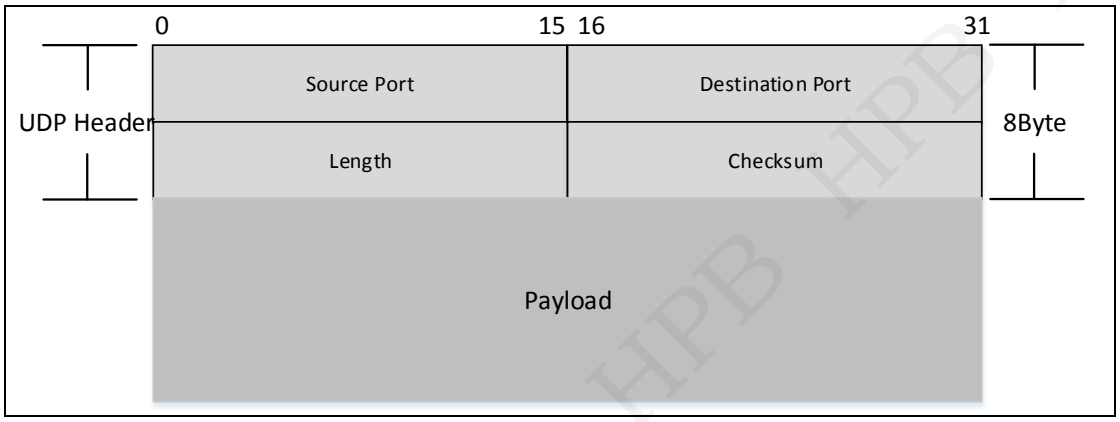

图 3-6 UDP 报文格式

3.3.5 PCIE

作为用户接口最多支持 40Gbps 带宽，内置 DMA 控制器提供 AXI4 接口和 AXI-lite 接口，分别用作收发报文数据传输接口和用户控制接口。

发送 TCP 报文前，需要先通过 PCIE 控制接口写入目的 IP、目的端口号，产生创建 TCP 连接信号，并等待连接建立成功状态。连接建立后，通过 DMA 将用户数据写入缓存，根据链路状态进行发送。

接收 TCP 报文前，需要先通过 PCIE 控制接口创建监听端口，由客户端发起 TCP 连接请求，请求连接后进行数据传输。接收到的 TCP 报文到达接收阈值后，触发 DMA 传输请求，DMA 根据预设源地址、目的地址进行数据搬移，传输完成后通知处理器。

UDP 报文的收发则简单一些，不需要建立连接。接收方向直接将从 UOE 模块收到的报文进行解封装、校验无误后写入缓存，触发 DMA 传输请求。

另外，从 IPC 模块输出的其它协议类型报文，直接写入缓存并触发 DMA 传输请求，将报文转交 CPU 处理。

3.3.6 MIG

DDR4 控制器，主要功能是对外部 DDR 颗粒进行初始化，将 DDR 接口的上下时钟沿转换成单时钟沿的用户数据，同时还可以产生周期性的刷新指令，完成对 DDR 存储器的动态刷新，从而保证数据的完整性。

本设计中由于最多需要支持上万条并发 TCP 连接，每个方向每条 TCP 连接的数据报文最大 64KB，最多需要 1.3GB 数据需要缓存，仅靠片内存储器无法实现，因此采用片外 DDR 来实现。

4 管脚列表

全局信号

信号名称	电平方向	信号描述	备注
resetni	LVCMOS18 I	系统复位 低电平有效，复位所有逻辑，异步复位，维持低电平至少 1us	
mclki	LVCMOS18 I	系统时钟 系统工作钟，由 100MHz 晶振提供	
swi[3:0]	LVCMOS18 I	调试开关输入	
ledo[3:0]	LVCMOS18 O	状态指示输出	

网络接口

信号名称	电平方向	信号描述	备注
xg_refcpi	LVPECL18 I	参考时钟 参考时钟 P 端输入，100MHz 晶振产生	
xg_refcni	LVPECL18 I	参考时钟 参考时钟 N 端输入，100MHz 晶振产生	
xg_rxpi	LVPECL18 I	光接收 差分数据 P 端输入	
xg_rxni	LVPECL18 I	光接收 差分数据 N 端输入	
xg_txpo	LVPECL18 O	光发送 差分数据 P 端输出	
xg_txno	LVPECL18 O	光发送 差分数据 N 端输出	

用户接口

信号名称	电平方向	信号描述	备注
pcie_rstni	LVCMOS18 I	模块复位 低电平有效，复位 PCIE 接口逻辑，异步复位	
pcie_refpi	LVPECL18 I	参考时钟 参考时钟 P 端输入，来自 PCIE 背板接口	
pcie_refni	LVPECL18 I	参考时钟 参考时钟 N 端输入，来自 PCIE 背板接口	
pcie_rxpi[7:0]	LVPECL18 I	接收数据 差分数据 P 端输入，共 8 个通道，每个通道 5Gbps	
pcie_rxni[7:0]	LVPECL18 I	接收数据 差分数据 N 端输入，共 8 个通道，每个通道 5Gbps	
pcie_txpo[7:0]	LVPECL18 O	发送数据 差分数据 P 端输出，共 8 个通道，每个通道 5Gbps	
pcie_txno[7:0]	LVPECL18 O	发送数据 差分数据 N 端输出，共 8 个通道，每个通道 5Gbps	

C0_DDR 接口

信号名称	电平方向	信号描述	备注
c0_resetn	LVCMOS12 O	复位输出	
c0_refcpi	DIFF_SSTL12 I	参考时钟 P 端输入，100MHz 差分晶振产生	
c0_refcni	DIFF_SSTL12 I	参考时钟 N 端输入，100MHz 差分晶振产生	
c0_actn	POD12_DCI O	激活命令 低电平时 c0_adr[16:14]为激活行地址，高电平时 c0_adr[16:14]分别对应行选通、列选通和写使能	
c0_odt	POD12_DCI O	片上匹配 高电平表示启用片上终端匹配电阻	
c0_csn	POD12_DCI O	片选信号 低电平有效	
c0_ckp	DIFF_SSTL12_DCI O	时钟输出 操作 DDR 差分时钟，1.2GHz	
c0_ckn	DIFF_SSTL12_DCI O		
c0_cke	POD12_DCI O	时钟使能 高电平有效，可正常访问 DDR 数据。低电平时用于 DDR 自刷新	
c0_bg[1:0]	POD12_DCI O	块选择 x16 模式下 bg[1]无效	
c0_ba[1:0]	POD12_DCI O	块地址 选通待激活块	
c0_adr[17:0]	POD12_DCI O	地址输出 用于选择行列地址	
c0_dm[7:0]	POD12_DCI O	数据选通 每个比特对应写命令数据总线的一个字节，高电平表示数据有效	
c0_dqs[7:0]	POD12_DCI IO	数据选通	
c0_dq[63:0]	POD12_DCI IO	读写数据	

C1_DDR 接口

信号名称	电平方向	信号描述	备注
c1_resetn	LVCMOS12 O	复位输出	
c1_refcpi	DIFF_SSTL12 I	参考时钟 P 端输入，100MHz 差分晶振产生	
c1_refcni	DIFF_SSTL12 I	参考时钟 N 端输入，100MHz 差分晶振产生	
c1_actn	POD12_DCI O	激活命令 低电平时 c0_adr[16:14]为激活行地址，高电平时 c0_adr[16:14]分别对应行选通、列选通和写使能	
c1_odt	POD12_DCI O	片上匹配 高电平表示启用片上终端匹配电阻	
c1_csn	POD12_DCI O	片选信号 低电平有效	
c1_ckp	DIFF_SSTL12_DCI O	时钟输出 操作 DDR 差分时钟，1.2GHz	
c1_ckn	DIFF_SSTL12_DCI O		
c1_cke	POD12_DCI O	时钟使能 高电平有效，可正常访问 DDR 数据。低电平时用于 DDR 自刷新	
c1_bg[1:0]	POD12_DCI O	块选择 x16 模式下 bg[1]无效	
c1_ba[1:0]	POD12_DCI O	块地址 选通待激活块	
c1_adr[17:0]	POD12_DCI O	地址输出 用于选择行列地址	
c1_dm[7:0]	POD12_DCI O	数据选通 每个比特对应写命令数据总线的一个字节，高电平表示数据有效	
c1_dqs[7:0]	POD12_DCI IO	数据选通	
c1_dq[63:0]	POD12_DCI IO	读写数据	