小 型 微 型 计 算 机 系 统
Journal of Chinese Computer Systems
DOI:10.20009/j.cnki.21-1106/TP.2025-0139
2026 年 4 月 第 4 期
Vol.47 No.4 2026

100G 网络 UDP 协议的并行实现及可靠性优化

陈晓杰[1],李 斌[2],周清雷[2]

[1](数学工程与先进计算国家重点实验室,郑州 450001)
[2](郑州大学 计算机与人工智能学院,郑州 450001)
E-mail:740248499@qq.com

摘 要: 目前高速信号采集装备的性能处于高速增长阶段,超高采样率的无线信号数据需要通过网络设备传输到后端进行识别检测,基于软件系统的网络协议难以满足高速数据传输,且需要专用设备将数据发送到软件端,增大了研发成本,而在 FPGA 上实现高效的网络传输协议,具有高速、并行化、集成度高等特点,因此,本文基于 FPGA 实现针对于 100G 高速网络的并行 UDP 网络传输协议,同时,协议栈具备 IP 分片、ARP/ICMP 请求应答、巨帧模式等基础功能,提高了应用的灵活性.UDP 本身传输效率较高,但可靠性较弱,设计了基于 DDR 缓存的数据重传机制和多端口并行功能,在特定条件下具备一定的可靠性.通过功能测试和性能测试,分析了不同应用场景采用的策略,100G 网络中的最大有效传输性能为 96.4Gbps,具有较高的传输效率,占用的计算资源较少.

关 键 词: FPGA;100G 网络;UDP 协议栈;IP 分片;数据重传
中图分类号: TP391 **文献标识码:** A **文 章 编 号:** 1000-1220(2026)04-0944-09

Parallel Implementation and Reliability Optimization of UDP Protocol in 100G Network

CHEN Xiaojie[1],LI Bin[2],ZHOU Qinglei[2]

[1](State Key Laboratory of Mathematical Engineering and Advanced Computing,Zhengzhou 450001,China)
[2](School of Computer and Artificial Intelligence,Zhengzhou University,Zhengzhou 450001,China)

Abstract: At present,the performance of high-speed signal acquisition equipment is in a stage of rapid growth,and the wireless signal data with ultra-high sampling rate needs to be transmitted to the back-end through network equipment for identification and detection. The network protocol based on software system is difficult to meet the high-speed data transmission,and special equipment is required to send the data to the software end,which increases the research and development cost. It has the characteristics of high speed,parallelization and high integration. Therefore,this paper realizes the parallel UDP network transmission protocol for 100G high-speed network based on FPGA. Meanwhile,the protocol stack has basic functions such as IP fragmentation,ARP/ICMP request response and giant frame mode,which improves the flexibility of application. UDP itself has high transmission efficiency,but weak reliability. A data retransmission mechanism based on DDR cache and multi-port parallel function are designed,which have certain reliability under certain conditions. Through the functional test and performance test,the maximum effective transmission performance of 100G network is 96.4Gbps,which has a high transmission efficiency and occupies less computing resources.

Keywords: FPGA;100G network;UDP protocol stack;IP fragmentation;data retransmission

0 引 言

随着网络设备带宽的飞速增长,将高速网络作为信号处理中数据传输载体,逐步得到了广泛应用.信号处理装备中,非协作对抗场景,前端设备只作为数据采集,随着数字信号采样率逐步增加,例如 AMD 厂商的信号处理现场可编程门阵列(Field Programmable Gate Array,FPGA)芯片 47DR 集成了 8 通道 ADC(Analog to Digital Converter),单路采样率最大为 5GSPS,采集总速率达到了 640Gbps[1],超高采样率的采集装备的应用也越来越广泛,而采集后的数据需要传输到服务端进行后续的检测、解调、译码等运算.用户数据报协议(User Datagram Protocol,UDP)位于网际互连协议(Internet Protocol,IP)上层,数据通信是无连接的,并且支持广播和组播模式,因此采用 UDP 协议传输高速信号数据,既能提高数据传输的多样性,也能提高网络设备带宽的利用率.

采用传统的 CPU 端软件实现 UDP 数据的传输,一方面,UDP 数据需要在系统底层进行逐级封装,难以并行化处理,导致数据传输的效率较低,另一方面,CPU 从信号处理装备中接收采集数据,再发送到目标端,增大了数据传输延时和成本.随着制造业的迅速发展,FPGA 作为数字加速设备在数字中心[2]、图像加密[3]、实时仿真[4]等领域得到了验证,将 FPGA 作为前端信号采集装备,已成为一个重要的硬件平台[5],在 FPGA 上实现 UDP 协议栈,实现数据的实时发送,既能满足高速的采集信号传输要求,也能减少设备冗余,降低生产

收稿日期:2025-03-19 收修改稿日期:2025-05-06 基金项目:嵩山实验室项目(241110210200)资助;河南省重大科技专项基金项目(221100210600)资助. 作者简介:**陈晓杰**(通信作者),男,1993 年生,博士研究生,研究方向为并行计算;**李 斌**,男,1986 年生,博士,讲师,研究方向为信息安全和高性能计算;**周清雷**,男,1962 年生,博士,教授,CCF 杰出会员,研究方向为信息安全.

成本.

UDP 协议与传输控制协议(Transmission Control Protocol,TCP)协议相比,没有确认应答、超时重传、拥塞控制等机制,虽然能够增大网络带宽的利用率,但是,UDP 是无连接的,且不能保证数据发送的完整性[6],而信号检测装备具有数据完整性要求,任何时刻都可能捕获关键信息,因此,提高 UDP 的可靠性是信号处理平台中高速网络传输的重要指标.

1　相关工作

UDP 是一种无连接的传输层协议,它提供了一种简单的数据传输服务,适用于需要快速传输和实时性要求较高的应用. 将 FPGA 硬件与 UDP 结合,使数据的传输效率获得极大的提高,Sasi 等人[7]在 FPGA 内以数据链路层三态以太网硬核为基础,设计并实现具有基础功能的 UDP/IP 协议,吞吐量 900Mbps,实时性满足声纳系统要求. 网络数据包处理消耗 CPU 大量的计算能力,Batmaz 等人[8]提出了基于 FPGA 的数据包卸载系统,为 CPU 预留了更多的算力,用于解决更复杂的计算任务. 针对 1000BASE-T 和 1000BASE-KX 链路,Födisch 等人[9]在 FPGA 实现了包含 ARP 和因特网控制报文协议(Internet Control Message Protocol,ICMP)的通用 UDP 协议栈,具有较高的带宽利用率. Dong 等人[10]实现了面向 10G 网络的 UDP 协议栈,能够应用于服务器场景,降低传输层协议对处理器的处理需求,提升系统的传输效能. 针对国产化 FPGA,Li 等人[11]实现了包含 ARP 请求、ARP 表查询、ICMP 协议、IP 协议、UDP 协议,具有较高的通用性和良好的性能表现.

实现 UDP 的可靠传输,目前有可靠的用户数据报协议(Reliable UDP,RUDP)、基于 UDP 的数据传输(UDP-based Data Transfer,UDT)、并行加速的用户数据报协议(Parallel Acceleration UDP,PA-UDP),三者的计算复杂度较高,且占用较高的 CPU 利用率[12],在 FPGA 上实现的难度较高. 文 Alberto 等人[13]提出了一种基于 UDP 的网络拥塞控制机制,使用从网络中测量的数据,根据预设策略,修改视频编码器的目标输出比特率,以降低或提高服务器向客户端发送的数据流. QUIC(Quick UDP Internet Connections)[14]由 Google 自研,通过滑动窗口的拥塞控制和顺序的数据保证数据的可靠传输. 在保证 UDP 的可靠传输中,需要保证丢失的数据能够再发送,而重传机制则是保证可靠传输的前提[15],Nie 等人[16]提出了一种基于 FPGA 的丢包重传技术,在无线传感器网络中充分利用 FPGA 的并行处理能力,在传输大量数据时降低丢包率.

现有的研究中对于 UDP 实现的加速技术和可靠性机制大多集中于千兆网中,而高速信号采集数据以 100Gbps 吞吐量为单位,难以满足应用需求,因此,本文的主要工作针对 100G 网络设计实现了 UDP 协议栈,包含 ARP 请求和响应、ICMP 请求和应答、UDP 的组帧和解析的基本功能,并且支持巨帧模式,可兼容千兆网和万兆网,并设计了基于 DDR(Double Data Rate Synchronous Dynamic Random Access Memory)缓存的数据重传机制和多端口并行,提高系统的可靠性.

2　基于 FPGA 的 UDP 协议栈实现

FPGA 是一种可编程逻辑器件,它可以根据用户的需求进行可重构硬件逻辑. 相比于传统的冯·诺依曼架构处理器具有更低的功耗[17]. FPGA 包含查找表、寄存器、DSP、CARRY 等逻辑计算资源,将各种资源通过电路布线连接可实现不同的功能,同时,各个资源具有独立性,通过时钟驱动,完成功能的并行化,具有高度的灵活性和并行计算能力,而 UDP 协议栈中的数据组帧以及关键的 CRC 校验计算,采用 FPGA 进行实现,即能发挥 FPGA 位级运算的特点,也能满足 UDP 设备低能效要求.

在 FPGA 上实现 UDP 数据协议时,为了增加协议栈的可扩展性,实现 ARP 和 ICMP 协议的基本功能,具有重要的工程实践意义. 各种网络协议和数据都要封装在以太网帧格式中,以太网帧通过物理层设备实现数据的接收和发送,以太网帧格式如表 1 所示.

表 1　以太网帧格式

Table 1　Ethernet frame format

字段	长度	值
前导码	7 字节	0x55555555555555
SFD	1 字节	0xd5
目的 MAC 地址	6 字节	目标端 MAC 地址
源 MAC 地址	6 字节	发送端 MAC 地址
类型	2 字节	0x0800/0x0806(IP/ARP)
数据	46~1500 字节	IP/ARP 数据
FCS	4 字节	CRC 码

在 FPGA 上实现 100G 网络数据的传输时,需要经过 FPGA 上专用的 IP(Intellectual Property)核,实现数字端到 01 码的转换,不同厂家的 FPGA 上 100G 硬核略有差异,以 AMD 的 FPGA 为例,其专用 IP 是 CMAC 硬核[18],用户接收和发送的报文包含目的 MAC 地址、源 MAC 地址、类型、数据字段,前导码、SFD 和 FCS 由 CMAC 内部进行解析和填充,而用户端在进行报文组帧和解析时,不需要再次操作,简化了报文实现的难度.

用户从 CMAC 接收数据和发送数据,采用 AXIS 协议,包含 512 位的数据端口 data、数据有效标记 valid、当前一组数据的最后一个 512 位数据 last,基于此端口,实现 ARP、ICMP、UDP 协议的组帧和解析.

2.1　ARP 协议请求和响应实现

ARP 协议封装在以太网帧格式的数据部分,帧格式定义如表 2 所示.

ARP 专有的帧格式中,前 4 个字段对于现有的网络设定为固定值,因此,设计 ARP 请求和应答时序结构如图 1、图 2 所示.

在 FPGA 内进行 ARP 请求时,ARP 请求模块接收到命令后,在一个时钟周期,拼接 64 字节数据,第 2 个时钟周期按照 AXIS 协议,将 valid 和 last 同时置 1,即一次请求发送 64 字节数据. 第 3 个时钟周期复位 AXIS 协议,删除端口占用. 等候 ARP 响应包,并在一个时钟解析目的 MAC 地址,如果超时未收到,则重复操作.

表 2 ARP 帧格式

Table 2 ARP frame format

字段	长度	值
硬件类型	2 字节	0x0001
协议类型	2 字节	0x0800
MAC 地址长度	1 字节	0x06
IP 地址长度	1 字节	0x04
操作码(OP)	2 字节	0x0001/0x0002(请求和应答)
源 MAC 地址	6 字节	发送端 MAC 地址
源 IP 地址	4 字节	发送端 IP 地址
目的 MAC 地址	6 字节	目标端 MAC 地址
目的 IP 地址	4 字节	目标端 IP 地址
填充	18 字节	0

在 FPGA 内实现 ARP 应答时,ARP 应答模块接收到数据包,第 1 个时钟通过帧类型和操作码字段判断当前帧是否为请求包,第 2 个时钟周期,拼接 64 字节的应答数据,第 3、4 个时钟周期用于 AXIS 接口的数据发送和赋值.

2.2 ICMP 协议请求和应答实现

ICMP 协议属于网络层协议,封装在 IP 帧格式中,IP 帧格式如表 3 所示.

ICMP 实现验证通信双方是否能够连接,故 IP 头字段中的标识、标志、片偏移可提前设置为 0,IP 帧格式中的数据部分包含 ICMP 协议,详细 ICMP 字段如表 4 所示.

ICMP 数据包的请求和应答的填充字段通常为 32 字节(a ~ w,a ~ i),因此,一个完整的ICMP报文组帧后包含两个

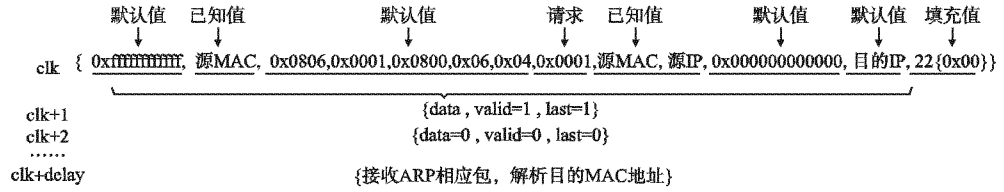

图 1 ARP 请求时序

Fig. 1 ARP request timing

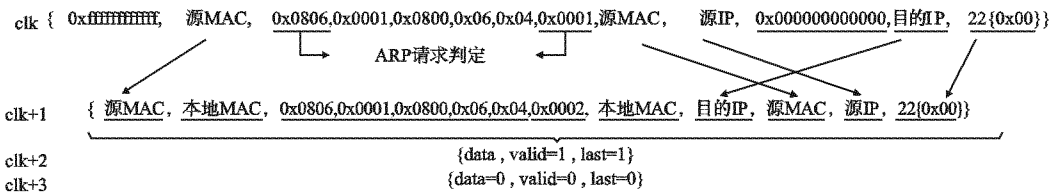

图 2 ARP 应答时序

Fig. 2 ARP reply timing

512 位数据,请求时序如图 3 所示.

表 3 IP 帧格式

Table 3 IP frame format

字段	长度(位)	值
版本	4	0x4/0x6
首部长度	4	5(word)
服务类型	8	0x00
总长度	16	IP 帧头 + IP 数据
标识	16	包计数
标志	3	分片标记
片偏移	13	当前 IP 数据的位置
生存时间	8	0x80
协议	8	0x01/0x11(ICMP/UDP)
首部校验和	16	计算获得
源 IP 地址	32	发送端 IP 地址
目的 IP 地址	32	目标端 IP 地址
数据	208 ~ 11776	

ICMP 请求操作包含两部分,第 1 部分采用 20 个时钟实现,第 1 个时钟拼接 IP 帧头和 ICMP 数据,第 2 个时钟将帧头数据发送到校验和计算模块,根据校验和计算方法分别计算出 IP 首部校验和和 ICMP 校验和,由于加法运算在 FPGA 内时序要求较高,而 ICMP 报文对性能要求较低,故采用串行的方式,每次计算两个 16 位数据,共消耗 16 个时钟. 第 17 个时钟将计算的校验和填充到对应位置. 第 18 个时钟拼接并发送第 1 个 512 位数据. 第 19 个时钟拼接并发送第 2 个 512 位数据. 第 20 个时钟复位 AXIS 端口. 第 2 个部分是等候 *delay* 个时钟后,收到响应包,并按规则解析. 再重复以上 3 次,将链路是否正常的结果返回给上层.

表 4 ICMP 帧格式

Table 4 ICMP frame format

字段	长度(位)	值
类型	8	0/8(应答/请求)
代码	8	0/0(应答/请求)
校验和	16	-
标识符	16	标识所属会话
序列号	16	当前会话的序列
填充	256	固定值

ICMP 响应模块实时等待接收数据,接收到数据后 8 个时钟完成响应,时序如图 4 所示.

ICMP 的响应操作第 1 个时钟是判定接收到的数据是否为 ICMP 报文,如果不是,则直接丢弃,如果是,进行校验和计算,判定当前接收到的数据包是否完整,如果通过校验,则进行组帧发送,结束后,重新进入等待状态.

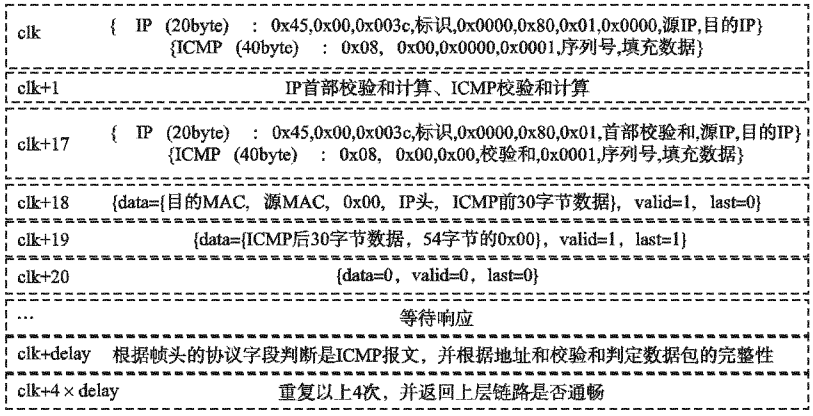

图 3　ICMP 请求时序

Fig. 3　ICMP request timing

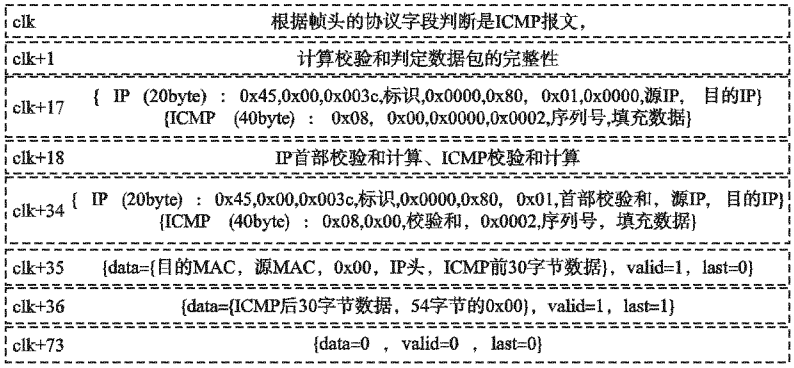

图 4　ICMP 响应时序

Fig. 4　ICMP reply timing

2.3　UDP 协议组帧和解析实现

UDP 协议的字段包含源端口、目的端口、UDP 长度和校验和 4 种字段,共 8 个字节,为了增大网络带宽的利用率,UDP 有效载荷数据越大,传输效率越高,最大可设置为 65507 字节,而 65507 字节超过了以太网帧的 MTU (Maximum Transmission Unit),故需要进行分片,而分片需要在 IP 首部进行计算,故设计的 UDP 实现结构如图 5 所示.

UDP组帧模块接收的数据每个时钟为64字节,将接收

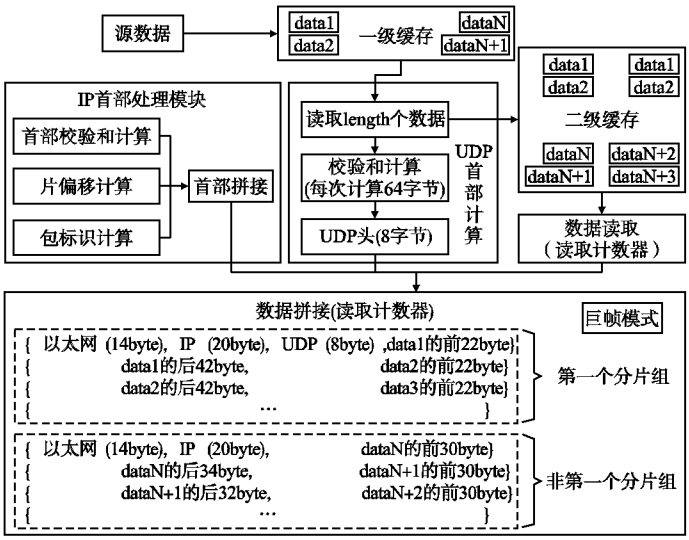

图 5　UDP 组帧实现结构

Fig. 5　UDP frame implementation structure

到的数据存储在第 1 级缓存中. 假设 UDP 的有效载荷为　*length* × 64,共分为 K 个子片,每片长度为 $N \times 64$,当进行组帧

时,首先,从第 1 级缓存中读取 *length* 组 64 字节数据,并行两组分别传输到二级缓存和校验和计算模块,读取到最后一个 64 字节数据后,将校验和拼接到 UDP 头中,与此同时,IP 首部处理模块计算出首部校验和和片偏移,从发送任务开始,每发送一个完整的 UDP 包,包标识序号累计加 1,根据每次发送的包序号在接收端可判断数据是否有丢失. 片偏移的计算通过每次累加 $N\times64$ 计算得出;然后,将计算好的 IP 帧头和 UDP 帧头发送到数据拼接模块,数据拼接模块检测到帧头计算完毕,则通过数据读取模块获取 N 组数据,进行拼接,发送一次拼接后的以太网帧,共发送 K 次,实现一个完整的 UDP 数据包.

数据拼接模块在发送一次完整的 UDP 数据报文时,共运行 K 次,第 1 次,帧头包含 UDP 帧头,共 8 个字节,而 FPGA 的 CMAC 接受的以太网帧的每个字节必须是连续的,故第 1 个 64 字节数据包含 14 + 20 + 8 的帧头,还需要拼接第 1 个 *data*1 的前 22 个字节数据,第 2 个 64 字节数据包含 *data*1 的后 42 字节和 *data*2 的前 22 个字节数据,以此类推,而后续的分片不包含 UDP 头,即第 1 个 64 字节数据包含 14 + 2 的帧头和 *dataN* 的前 30 个字节数据.

在 FPGA 上实现 UDP 协议的组帧,当数据开始输入时,第 1 组待处理的数据需要计算校验和,消耗 *length* 个时钟周期,开始组帧后,需要消耗 *length* + K 个时钟周期,同时,第 2 组待处理的数据计算对应的校验和,形成模块级流水线,假设,计算频率为 f_{UDP},则 UDP 组帧的吞吐量如公式(1)所示:

$$\text{Throughput}_{UDP}=\frac{length}{length+K}\times f_{UDP} \tag{1}$$

从公式(1)中可以看出分片越小,吞吐量越高,巨帧模式能够突破 MTU 的限制,以太网帧最大可设置为 9014 字节,通过巨帧模式,可以有效减少重复的帧头数据在网络上的占用,减少分片. UDP 的实现中,可配置为巨帧模式,此时,只需要更改数据读取长度和数据拼接长度,通过巨帧模式,能够提高数据的传输效率.

当 FPGA 作为 UDP 数据报文的接收端时,需要对 UDP 报文进行解析,UDP 解析模块通过网络接口接收数据,首先判断 MAC 地址是否为本设备,如果是再进行判断 IP 地址协议类型,如果满足是 UDP 报文,然后检测分片标识,提取有效载荷数据,并存储到缓存中,如果分片标识达到最后一包,计算输出 UDP 校验和,如果数据完整,则从缓存中读取 UDP 载荷数据发送到上层应用.

本文分别实现了 ARP、ICMP、UDP 报文,通过上层逻辑控制的整合,实现较完整的 UDP 协议栈,整体结构如图 6 所示.

ARP 请求、ICMP 请求以及 UDP 组帧模块由控制模块下发命令,发送对应的数据帧;帧解析预处理模块对接收到的数据帧进行初步过滤,对于不符合本设备地址以及本设备中没有的协议直接丢弃,并检测协议号和协议类型,将数据帧发送到对应模块;ARP 响应和请求、ICMP 应答和请求以及 UDP 组帧模块需要输出报文,故将 5 个模块的数据存储在符合 CMAC 端口的 AXIS 类型的缓存中,再通过轮询模块实时的将数据输出;UDP 解析后的数据直接存储在 FIFO 中,并输出到上层应用.

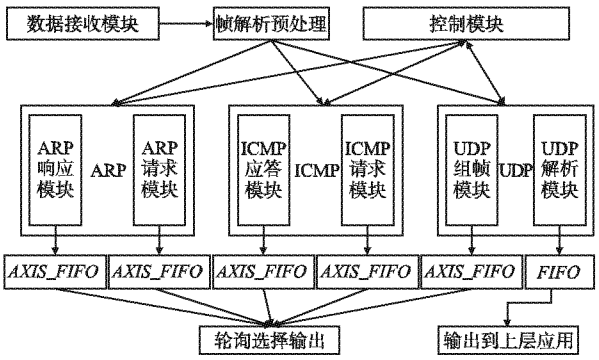

图 6 UDP 协议栈整体实现结构

Fig. 6 Overall implementation structure of UDP protocol stack

2.4 不同速率网络的兼容性设计

在边缘传感器网络应用中,需要网络传输的数据量较少,且硬件接口随着传输性能的增加而增加,本文所设计方案以 100G 网络传输要求进行构建,保证每个时钟可处理 512 位数据,而受到物理限速及宽度的影响,在 FPGA 内部,千兆网、万兆网、40G 网络在每个时钟处理的位宽分别为 8 位、64 位、128 位,因此,为了提高协议栈在不同速率网络的兼容性,增加位宽转换模块,实现结构如图 7 所示.

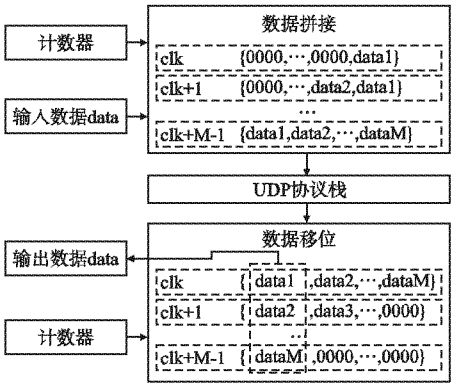

图 7 不同带宽网络转换结构

Fig. 7 Different bandwidth network conversion structure

假设输入 UDP 协议栈的输入位宽是 *width*,512/*width* = *M*,计数器值得范围为 0 ~ *M* - 1,当输入数据时,数据拼接模块将第 1 个数据放入 512 位的低位,第 2 个数据输入时,将 512 位数据左移 *width* 位,并将第 2 个数据放入到低位,当 *N* 个数据输入完毕后,512 位数据拼接完成输出到 UDP 模块.

当 UDP 协议栈输出数据时,数据移位模块每次读取一个 512 位数据,然后截取 512 位数据的高 *width* 位,作为第 1 个数据输出,同时,将低 512-*width* 位数据左移 *width* 位,低位补 0,重复 *N* 次,输出 *N* 个数据,完成数据的移位输出.

通过增加数据拼接和数据移位两个模块,可根据网络应用需要,只需要修改参数 *width*,即可完成协议栈的移植,提高算法的灵活性.

3　基于 FPGA 的 UDP 协议栈可靠性优化

3.1　基于 DDR 缓存的重传机制

数据重传机制用于目的端检测到数据不完整,出现丢包的情况时,目的端反馈信息到源端,源端再次重发丢失的数据包,从而保证目的端接收到完整的数据. 传统的数据重传机制是建立在数据传输速度要求较低的场景下,待重传数据能够存储在源端较长时间,而信号处理装备中,数据采集的速率比较高,需要构建一种新的重传机制,以适应高速采集数据的丢包重传.

随着 DDR 的生产成本的降低,选用大容量 DDR 作为板载存储载体,成为了硬件设计阶段的首选. 以 FPGA 板卡包含两组 DDR,每组 DDR 容量包含 8GB 为硬件基础,设计重传机制,数据重传机制如图 8 所示.

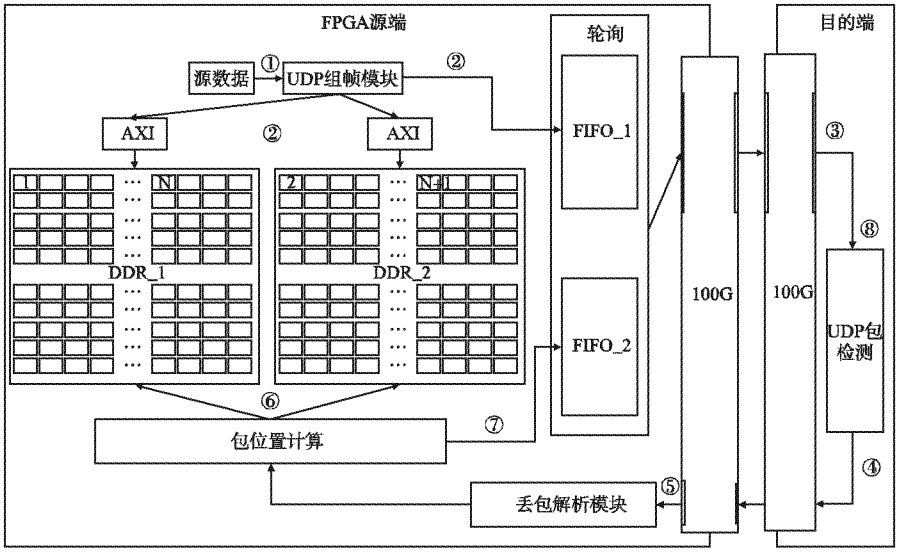

图 8　数据重传流程

Fig. 8　Data retransmission process

基于 DDR 缓存的数据重传机制时序包含 8 个步骤:

第 1 步. UDP 组帧模块接收源数据,并将数据按照 UDP_IP 协议格式进行组帧. 完整的 UDP 协议包最大字节长度小于 65536.

第 2 步. 组帧后的数据复制两份,一份发送到 FIFO_1 缓存模块,用于 100G 网直接输出,另一份数据,通过乒乓缓存机制存储到 DDR 中,奇数包序号存储到第 1 组 DDR 中,偶数包序号存储到第 2 组 DDR 中.

一个完整的 UDP 协议包最大为 65536 字节,FPGA 通过 AXI 接口传输数据,一次突发数据字节长度可设置为 64、128、256、512、1024、2048、4096、8192、16384,为了简化地址的计算复杂度,设置一个 UDP 协议包数据长度小于 65536,采用 4 次突发,每次突发长度为 16384,则每个 UDP 协议包存储的起始地址都为 65536 的整数倍,每组 8GB 的 DDR 容量,最大包序号为 262143.

第 3 步. 目的端接收到 UDP 数据包,解析并判断是否丢包.

第 4 步. 如果存在丢包,目的端通过 100G 光口发送 UDP 协议包,包含丢包的序号信息.

第 5 步. FPGA 端实时接收带有丢包信息的数据包,解析后,将丢包的序号发送到包位置计算模块. 如果存在丢包,目的端通过 100G 光口发送 UDP 协议包,包数据中含有丢包的序号信息.

第 6 步. 包位置计算模块读取丢包的序号,计算丢失的 UDP 包存储位置,假设丢包的序号为 x,首先判断是奇数还是偶数,如果是奇数,则 DDR 地址为 $(x-1)/2\times65536$,如果是偶数,则 DDR 地址为 $x/2\times65536$.

第 7 步. 根据计算所得丢失包的位置,通过 AXI 接口读取数据并发送到 FIFO_2 存储,通过光纤接口再次发送到目的端.

第 8 步. 目的端检测接收到的重传数据,补全到丢失位置.

数据重传过程中,各个模块的吞吐量决定了数据重传效率,假设 UDP 组帧的吞吐量为 $\mathrm{Throughput}_{\mathrm{UDP}}$,丢包的吞吐量为 $\mathrm{Throughput}_{\mathrm{lose}}$,轮询模块的吞吐量为 $\mathrm{Throughput}_{\mathrm{poll}}$,则应满足公式(2)所示:

$$\mathrm{Throughput}_{\mathrm{UDP}}+\mathrm{Throughput}_{\mathrm{lose}}<\mathrm{Throughput}_{\mathrm{poll}}<100\mathrm{Gbps} \tag{2}$$

通过数据重传中的 8 个步骤,当数据发送速率达到 100Gbps,通过 DDR 缓存机制,能够重传 1 秒内丢失的数据包,且不需要额外的存储设备,提高设备利用率.

3.2　多端口并行设计

CPU 端软件的并行化处理主要通过进程中的多线程实现,当 CPU 端作为高速网络的目的端时,为了尽可能的接收更多数据,可将接收数据的主任务,分成多个子任务并行处理,而不同的子任务,可通过 UDP 协议的端口号进行识别,从而提高 CPU 的处理能力,为了实现这一目的,FPGA 作为发送端,需要实现多端口的 UDP 报文组帧.

FPGA 具有资源并行化的特点,而 UDP 协议栈的实现占用较少资源,随着 FPGA 的制造工艺的提升,FPGA 的关键资源以达到百万级,因此,一些高性能的 FPGA 具备足够资源实现多端口并行的 UDP 协议.

实现多端口 UDP,主要包含并行的 UDP 实现、数据的分发以及多对一的数据选择模块,假设待发送数据 data_1、data_2、data_3、…、data_end,端口数为 N,UDP 协议的有效载荷长度为 $length \times 64$,由于 FPGA 内部的 MAC 层 100G CMAC IP 核的输入位宽为 512 位,因此,在每个待发送数据的位宽为 512.

数据计数器模块用于数据的分发,判断 FIFO 内的有效数据是否达到 *length*,如果达到,则从 FIFO 内读取 *length* 个数据作为一组输出分发到对应的 UDP 模块中,分组的部分结果如公式(3)~公式(7)所示:

$$\text{Data\_Group\_1} = \text{FIFO}(\text{data\_1、data\_2、}\cdots\text{、data\_}length) \tag{3}$$

$$\text{Data\_Group\_2} = \text{FIFO}(\text{data\_}(length+1)\text{、}\cdots\text{、data\_}(length\times 2)) \tag{4}$$

$$\text{Data\_Group\_}N = \text{FIFO}(\text{data\_}(length\times(N-1)+1)\text{、}\cdots\text{、data\_}(length\times N) \tag{5}$$

$$\text{Data\_Group\_}(N+1) = \text{FIFO}(\text{data\_}(length\times N+1)\text{、}\cdots\text{、data\_}(length\times(N+1))) \tag{6}$$

$$\text{Data\_Group\_}(\text{end/length}) = \text{FIFO}(\text{data\_}(\text{end}-\text{length}+1)\text{、}\cdots\text{、data\_}(\text{end}-1)\text{、data\_end}) \tag{7}$$

配置的端口数量和 UDP 模块数量相同,每个 UDP 模块可独立配置目的 IP 地址、目的 MAC 地址、目的端口号等信息,当 UDP 数据包发送到同一个主机,则配置相同的 IP\MAC 地址,再单独配置端口号,不同 UDP 模块的输入如公式(8)~公式(10)所示:

$$\text{UDP\_1}(\text{IP\_1、MAC\_1、Port\_1}) = \{\text{Data\_Group\_1、Data\_Group\_}(N+1)\text{、}\cdots\} \tag{8}$$

$$\text{UDP\_2}(\text{IP\_2、MAC\_2、Port\_2}) = \{\text{Data\_Group\_2、Data\_Group\_}(N+2)\text{、}\cdots\} \tag{9}$$

$$\text{UDP\_}N(\text{IP\_}N\text{、MAC\_}N\text{、Port\_}N) = \{\text{Data\_Group\_}N\text{、Data\_Group\_}(N+N)\text{、}\cdots\} \tag{10}$$

顺序读取模块按顺序从 N 个 UDP 模块获取组帧后的数据,通信协议采用 AXIS 协议,包含有效数据 data、当前数据有效标记 valid、当前包的最后一包 last、UDP 包组帧完成标记 ready,顺序读取模块只需要判断这 4 个信号,就可以完整的读取 UDP 帧数据,即使每个 UDP 模块的组帧长度不同,也不影响数据的读取,减少各模块的耦合度,增加功能的灵活性.

多端口并行设计中,每个时钟的输入为 512 位数据,假设输入的时钟频率为 f_{inclk},最大的吞吐量为 $f_{inclk}\times 512$,当进行高速的多端口网络数据传输时,每个 UDP 模块的吞吐量应满足公式(11)所示,否则将导致数据丢失.

$$\text{Throughput}_{\text{UDP}} \times N \geqslant f_{inclk} \times 512 \tag{11}$$

同时,对于低速的接收网络,可通过降低 UDP 协议栈的处理位宽和计算频率,减少 FPGA 的资源占用和运行能效,以适用不同的场景.

4 实现结果与分析

本文采用的硬件平台为 AMD 的 FPGA,芯片型号为 Virtex UltraScale + 系列的 XCVU9P,其关键计算组件 LUT Elements 资源 1182240 个,FlipFlops 资源 2364480 个,Block RAM 资源 2160,可满足多种应用场景下的并行加速、高速通信需求.

本文设计并实现了 UDP 协议栈的 RTL(Register Transfer Level)级代码,通过 Vivado 软件进行仿真、综合、布线,资源占用如表 5 所示.

表 5 资源占用

Table 5 Resource occupation

各模块	CLB LUTs	CLB Registers	Block RAM
ICMP	467	521	1
ARP	384	465	1
UDP	3154	5261	30
预处理模块	482	569	7.5
控制模块	626	834	0
重传控制模块	3526	7312	15
总计	8539	14762	54.5

通过资源分析,所设计的 UDP 协议栈关键计算单元占用总资源的 7.23%,具有较少的资源占用.

为了验证 UDP 传输的极限性能,搭建的硬件环境包含两块相同性能的 FPGA 卡,板卡上包含 100G 光模块,并使用光纤线进行直接连接,由于没有经过任何的网络设备,故不会出现数据报乱序的现象,其中一块作为发送端,一块作为接收端,通过逐步增大数据源的发送速度,并且设置不同的 UDP 包长度,单位为字节,经过 UDP 组帧后在接收端统计是否丢包,实现结果如表 6 所示.

表 6 极限性能测试

Table 6 Ultimate performance test

发送速率	不分片 512	不分片 1024	分片 20 25600	分片 40 51200	分片 6 51200
10Gbps	√	√	√	√	√
80Gbps	√	√	√	√	√
85Gbps	√	√	√	√	√
88Gbps	×	√	√	√	√
92Gbps	×	×	√	√	√
94Gbps	×	×	×	√	√
94.7Gbps	×	×	×	×	√
96.4Gbps	×	×	×	×	×

在表 6 中,验证了不同 UDP 长度和不同的发送速率的接收情况,在不分片时,且 UDP 的有效载荷为 512 字节,由于以太网帧中包含了以太网帧头、IP 帧头和 UDP 帧头,100G 网中的带宽大量的被帧头占用,导致有效载荷的速率为 88Gbps 时即出现了丢包.

当 UDP 中的有效载荷增大一倍时,在不进行分片时,可传输速率提高 6%,当进一步提高 UDP 载荷时,设置 IP 层进行分片,由于传输中减少了 UDP 帧头的发送,故传输速率有效数据传输速率能够进一步提高,在巨帧模式下,有效数据的发送速率达到 96.4Gbps 出现丢包,性能提升了 9.54%,具有较高的 100G 网络带宽利用率和传输性能.

以 FPGA 作为发送端,主机上插入 100G 网卡作为接收端,CPU 型号为 i7-7700k,在 FPGA 端配置不同的发送速度和

UDP 有效载荷长度,主机端通过检测 UDP 数据报文中的序号来判断是否丢包,实现结果如表 7 所示.

表 7　丢包测试

Table 7　Packet loss test

发送速率	不分片 512	不分片 1024	分片 20 25600	分片 40 51200	分片 6 51200	分片 6 51200/3 端口
1Gbps	√	√	√	√	√	√
1.45Gbps	×	√	√	√	√	√
1.67Gbps	×	×	√	√	√	√
1.8Gbps	×	×	×	×	√	√
4.6Gbps	×	×	×	×	×	√
4.92Gbps	×	×	×	×	×	×

由于 CPU 的系统层解析数据报文时,需要按照不同协议逐层解析,当接收到大量的网络包时,需要进行多次软中断与网卡进行交互,从而导致 CPU 接收性能低于 FPGA,故测试的发送速率在 1.45Gbps 时就出现了丢包.

虽然 CPU 的接收能力较弱,但通过调整发送端 FPGA 的分片功能和巨帧模式,使 CPU 的接收速率提升了 2.17 倍,且通过多端口策略,使 CPU 可通过多线程并行,CPU 的接收能力进一步提高了 6.96%,因此,在 CPU 作为网络的接收端时,FPGA 发送端应调整为巨帧模式,且尽可能增大 UDP 有效数据载荷.

数据重传机制能够在一定条件下实现数据传输的可靠性,即数据的发送速率应低于数据接收方的接收性能,如果发送速率较高,造成了数据丢包,此时,再进行数据重传,会占用原本的接收带宽,可能造成更多的数据报丢失.

搭建测试环境,以 FPGA 作为发送端,UDP 有效载荷设置为 51200 字节,CPU 作为接收端,且 CPU 端通过 IP 帧头的序号字段判断丢失的数据包,并反馈给 FPGA,实现结果如表 8 所示.

表 8　重传测试

Table 8　Retransmission test

发送速率	丢包数	丢包率	重传接收数	重传接收率	重传过程中的丢包率
1.4Gbps	—	——	5	100%	0%
1.5Gbps	—	——	5	100%	0%
1.6Gbps	—	——	5	75%	0%
1.7Gbps	—	——	5	50%	0.02409%
1.8Gbps	1	0.02277%	1	100%	0.02277%
1.9Gbps	1	0.02156%	1	100%	0.02156%
2.2Gbps	89	1.65704%	16	18%	3.07205%

数据传输速率小于 1.7Gbps 时,CPU 接收端能够完全接收到数据,不会造成丢包,故在测试中每秒从接收到数据包中选择 5 个序号,模拟丢包,通过测试,数据包能够正常重传,可验证重传功能的正确性,且随着发送速率的增大,到 1.8Gbps 时,出现了丢包,然后按照实际的丢包重传,接收端能够正确接收到数据包.

当发送速率增加到 2.2Gbps,出现了大量的丢包,此时,根据实际的丢包数进行重传,此时数据链路上的传输的速率大于 2.2Gbps,所以 CPU 接收数据的丢包率提高了将近 2 倍,也验证了数据重传机制需要在特定条件下使用.

文献[11]在国产化 FPGA 上实现了 UDP 协议栈,资源消耗对比如表 9 所示.

表 9　资源对比

Table 9　Resource comparison

实现方案	CLB LUTs	CLB Registers	Block RAM
文献[11]	5715	8755	56
本文	5013	7450	39.5

由于增加了数据重传功能,故在对比时,仅对比 UDP 协议相关资源,通过资源对比,本文资源占用较少,验证本文实现方案的可行性.

通过多种实验验证和对比分析,验证了本文方案的可行性,且具有较高的性能和较低的资源占用,同时,UDP 协议栈的可靠性和兼容性方面也具有一定的优势.

5　总结与展望

针对高速信号采集装备中要求高效的数据网络传输要求,文中设计实现了 UDP 协议栈,包含 ARP 请求和应答、ICMP 请求和响应、以及并行的 UDP 组帧和解析功能,通过实验验证,具有较高的传输性能,并且通过参数配置,使协议栈能够移植到千兆网、万兆网、100G 网的不同带宽应用需求;然后,设计了数据重传机制,能够满足一定的可靠性要求;最后,通过 FPGA 并行化特点,实现多端口 UDP 并行,在软件端通过线程并行提高接收性能.

面向 100G 网络的 UDP 协议在 FPGA 上进行实现,即可以作为高速数据的发送端,也可以在特定的高速网路中,作为 UDP 协议的接收端,将有效载荷数据上传,能够减少系统软件的计算开销,提高设备利用率. FPGA 在高速信号的采集、分析设备中发挥着重要作用,而 UDP 协议栈占用资源较少,在设备的研发上,可将 UDP 协议栈进行内嵌,进一步压缩系统装备中的硬件成本.

References:

[1] Zynq UltraScale + RFSoC Product Selection Guide(XMP105). [EB/OL]. https://docs. amd. com/v/u/en-US/zynq-usp-rfsoc-product-selection-guide,2023-02-28.

[2] WANG Y W,LI R G,XU R,et al. Data center heterogeneous acceleration software-hardware system-level platform based on reconfigurable architecture[J]. Journal of Computer Research and Development,2025,62(4):963-977.

[3] Kong X,Yu F,Yao W,et al. Memristor-induced hyperchaos,multi-scroll and extreme multistability in fractional-order HNN: image encryption and FPGA implementation[J]. Neural Networks,2024,171:85-103,doi:10.1016/j.neunet.2023.12.008.

[4] Xu H,Zheng J,Zeng Y,et al. Topology-aware matrix partitioning method for FPGA real-time simulation of power electronics systems[J]. IEEE Transactions on Industrial Electronics,2024,71(7):7158-7168.

[5] LI G F,NAN G Y,PAN D Y,et al. Research on signal acquisition and processing of lidar wind measurement system[J]. Infrared and

Laser Engineering,2021,50(S2):188-194.
[6] Lu Q,Li J,Yuan K,et al. UDP-RT:a UDP-based reliable transmission scheme for power WAPS[J]. Computer networks,2023,236:1-13,doi:10.1016/j.comnet.2023.110012.
[7] Sasi A,Saravanan S,Pandian S R,et al,UDP/IP stack in FPGA for hard real-time communication of Sonar sensor data[C]// IEEE International Symposium on Ocean Electronics(SYMPOL),2013:1-6.
[8] Batmaz B,Doan A. UDP/IP protocol stack with PCIe interface on FPGA[C]//Embedded Systems and Applications(ESA),2015:49-53.
[9] Födisch P,Lange B,Sandmann J,et al. A synchronous gigabit ethernet protocol stack for high-throughput UDP/IP applications[J]. Journal of Instrumentation,2015,11(1):1-19.
[10] DONG Y J,WANG Y,YUAN Z. Design and implementation of 10G ethernet UDP/IP hardware protocol stack based on FPGA [J]. Application Research of Computers, 2022, 39 (8): 2465-2468.
[11] LI S,TANG J,YU Q. Design and implementation of UDP protocol stack IP core based on domestic FPGA[J]. Journal of Air & Space Early Waring Research,2024,38(5):347-352+363.
[12] Masirap M,Amaran M H,Yussoff Y M,et al. Evaluation of reliable UDP-based transport protocols for Internet of Things(IoT)[C]//IEEE Symposium on Computer Applications & Industrial Electronics(ISCAIE),Penang,Malaysia,2016:200-205.
[13] Alberto Alós,Francisco Morún,Carballeira P,et al. Congestion control for cloud gaming over UDP based on round-trip video latency[J]. IEEE Access,2019,7:78882-78897,doi:10.1109/ACCESS.2019.2923294.
[14] Carlucci G,Cicco L D,Mascolo S. HTTP over UDP:an experimental investigation of QUIC[C]//SAC'15:Proceedings of the 30th Annual ACM Symposium on Applied Computing,Association for Computing Machinery,2015:609-614.
[15] Aziz S M,Pham D M. Energy efficient image transmission in wireless multimedia sensor networks[J]. IEEE Communications Letters,2013,17(6):1084-1087.
[16] Nie Y,Song P,Yang C,et al. Lost packet retransmission mechanism based on hardware acceleration in wireless sensor networks [C]//2nd International Conference on Applied Mathematics, Modelling and Statistics Application(AMMSA),2018:321-324.
[17] YU S H,YI M J,WU Z,et al. Neuromorphic computing:from spiking neural network to edge deployment[J]. Journal of Software, 2025,36(4):1758-1795.
[18] UltraScale devices integrated 100G ethernet subsystem product guide(PG165)[EB/OL]. https:// docs. amd. com/r/en-US/pg165-cmac,2024-07-12.

附中文参考文献:

[2] 王彦伟,李仁刚,徐 冉,等. 基于可重构架构的数据中心异构加速软硬件系统级平台[J]. 计算机研究与发展,2025,62(4):963-977.
[5] 李光福,南钢洋,潘冬阳,等. 激光雷达测风系统信号采集处理究[J]. 红外与激光工程,2021,50(S2):188-194.
[10] 董永吉,王 钰,袁 征. 基于 FPGA 的万兆以太网 UDP_IP 硬件协议栈设计与实现[J]. 计算机应用研究,2022,39(8):2465-2468.
[11] 李 森,唐 建,袁 强. 基于国产 FPGA 的 UDP 协议栈 IP 核设计与实现[J]. 空天预警研究学报,2024,38(5):347-352+363.
[17] 俞诗航,易梦军,吴 洲,等. 神经形态计算:从脉冲神经网络到边缘部署[J]. 软件学报,2025,36(4):1758-1795.