doi:10.11887/j.cn.202105014 http://journal.nudt.edu.cn

高速 SpaceFibre 总线节点的系统设计*

祝 平[1,2],朱 岩[1],安军社[1],江源源[1],周 莉[1]
(1. 中国科学院 国家空间科学中心 复杂航天系统电子信息技术重点实验室,北京 100190;
2. 中国科学院大学,北京 100049)

摘 要:为了实现 SpaceFibre 总线节点的高效数据传输,针对网络协议中关键问题和技术提出了一种基于现场可编程门阵列(Field Programmable Gate Array,FPGA)的 SpaceFibre 总线节点系统设计方案。其中,采用了轮询仲裁算法,解决了多路虚拟通道中流量控制字的申请冲突;设计了基于服务质量机制的高效处理状态机,实现了多路虚拟通道的服务质量调度;提出了一种并行的分区存储架构和重发控制算法,实现了基于错误检测隔离恢复机制的错误恢复;采用了不同的数据并行处理方案,实现了多种数据格式的循环冗余校验和伪随机序列的计算。通过 ModelSim 仿真平台对节点系统进行功能仿真,并在 Virtex-6 FPGA 上完成了系统验证。结果表明,该设计实现了 SpaceFibre 总线节点的功能,串行传输速度可达 3.125 Gbit/s,能够满足高速数据传输需求。

关键词:SpaceFibre;SpaceWire;服务质量;错误检测隔离恢复;高速总线
中图分类号:TP336 **文献标志码**:A **文章编号**:1001-2486(2021)05-117-10

System design of high-speed SpaceFibre node

ZHU Ping[1,2], *ZHU Yan*[1], *AN Junshe*[1], *JIANG Yuanyuan*[1], *ZHOU Li*[1]
(1. Key Laboratory of Electronics and Information Technology for Space System, National Space Science Center, Chinese Academy of Sciences, Beijing 100190, China; 2. University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: In order to achieve efficient data transmission of SpaceFibre node, a system design scheme of SpaceFibre node based on FPGA (field programmable gate array) was proposed aiming at the key problems and technologies in network protocols. The polling and arbitration algorithm was adopted to solve the flow control token words application conflict of multiple virtual channels. An efficient processing state machine based on QoS (quality of service) mechanism was designed, which can realize QoS scheduling for multiple virtual channels. A parallel partitioned storage architecture and a resend control algorithm were proposed, which can realize efficient error recovery based on FDIR (fault detection isolation and recovery) mechanism. Different parallel processing schemes were designed for cyclic redundancy check of various data formats and calculation of pseudo-random sequence. A simulation platform was built by ModelSim to test the function simulation of the node system, and the system verification was completed on Virtex-6 FPGA. The results show that the function of SpaceFibre bus node can be realized, and the serial transmission speeds up to 3.125 Gbit/s, which can meet the demand of high-speed data transmission.

Keywords: SpaceFibre; SpaceWire; quality of service; fault detection isolation and recovery; high speed bus

随着空间技术的飞速发展,星上任务的规模和复杂度也相应地提升,面对星上各类载荷设备越来越多的数据量,需要考虑不同数据源的传输需求和载荷设备之间灵活的互联方案。传统的控制器局域网络(Controller Area Network, CAN)总线、1553 B 越来越难以满足空间任务的发展需求。2003 年,欧空局提出的 SpaceWire 总线协议支持全双工点对点和灵活的路由组网,但传输速度最高只有 400 Mbit/s。

针对航天传输任务中的诸多需求,2019 年欧空局在 SpaceWire 总线的基础上提出了面向高速传输的新一代高速互联总线网络 SpaceFibre[1] 的标准。SpaceFibre 支持点对点全双工传输,采用 8 B/10 B 编码,单通道速率最高可达 6.25 Gbit/s,多通道传输速率可达 50 Gbit/s,支持光纤和电缆传输;支持路由网络结构,拓扑灵活,能够实现各类星载设备的互联。同时 SpaceFibre 能够在网络层兼容 SpaceWire,实现已有 SpaceWire 设备的平

* **收稿日期**:2020-02-24
基金项目:中国科学院战略性先导科技专项基金资助项目(XDA15020205)
作者简介:祝平(1993—),女,山东济宁人,博士研究生,E-mail:zhuping9382@163.com;
朱岩(通信作者),男,研究员,博士,博士生导师,E-mail:zhuyan@nssc.ac.cn

稳升级。

SpaceFibre 技术在国外的研究开始较早，逐渐成为国际上星载总线技术的研究热点。英国邓迪大学的 Parks 等[2-4]研发了基于各类抗辐照和商业 FPGA 的 SpaceFibre 节点知识产权（Intellectual Property，IP）核和路由 IP，以及路由设备 SUNRISE 及配套的测试分析仪 STAR Fire 和 STAR Fire MK3[5]，并设计了一款抗辐照的 SpaceFibre 超高速串行接口 VHiSSI[6]，完成了测试和流片。德国比勒费尔德大学的 Jungewelter 等[7]和瑞典 Cobham Gaisler 公司的 Siegle 等[8]也研制了相关的节点 IP。日本电气股份有限公司的 Hiroki 等[9]在研发的新型纳米桥现场可编程门阵列（NanoBridge Filed Programmable Gate Array，NBFPGA）上实现了 SpaceFibre 传输来自光学传感器和合成孔径雷达的高清图像，并在“创新卫星技术验证项目”中进行了验证。

国内对 SpaceWire 技术的研究较成熟，对 SpaceFibre 技术的研究尚处于起步阶段。上海创景计算机系统有限公司的徐曙清等[10]对 SpaceWire 和 SpaceFibre 的研究现状进行了概述。北京航空航天大学的伊小素等[11]针对 SpaceFibre 的服务质量（Quality of Service，QoS）机制进行了仿真研究，张春熹等[12]对 SpaceFibre 的即插即用技术进行了性能分析。

本文通过研究 SpaceFibre 网络协议，针对 SpaceFibre 网络中的关键问题和技术，提出一种 SpaceFibre 总线节点的系统设计方案。为了解决多路虚拟通道的流量控制申请冲突，提出了轮询仲裁算法；针对 QoS 调度机制中诸多参数和变量的计算比较，设计了一个能高效进行 QoS 参数计算并完成 QoS 调度的状态机；针对复杂的错误检测隔离恢复（Fault Detection Isolation and Recovery，FDIR）机制设计了一个并行的分区存储架构和重发控制算法，能实现备份的清除和重发，降低了处理延迟；针对各类数据的循环冗余校验码和伪随机序列的生成，设计了对应的并行处理方案。最后搭建仿真平台，通过仿真验证和板上测试，结果表明本文设计方案实现了 SpaceFibre 总线节点的功能，串行传输速度可达 3.125 Gbit/s，满足实际高速数据传输需求。

1 SpaceFibre 协议栈

SpaceFibre 协议栈包括物理层、通道层、多通道层、数据链路层、网络层以及管理层，协议栈框架如图 1 所示。

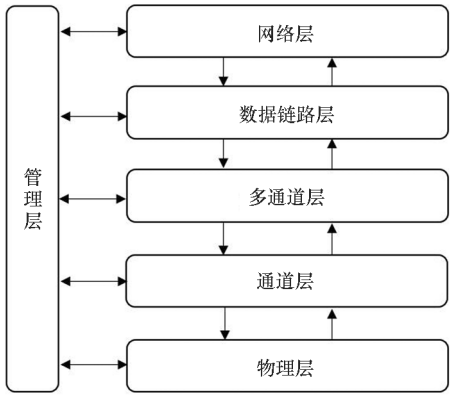


图 1 SpaceFibre 协议栈

Fig. 1 Protocol of SpaceFibre

物理层主要实现数据的串并转换、串行收发以及时钟恢复，传输介质可采用光纤或者电缆；通道层主要通过链路初始化状态机以及通道控制字实现链路的连接，并完成 8 B/10 B 编解码、字符同步字同步、时钟矫正等功能；多通道层主要实现多通道的控制，提高数据传输的吞吐率；数据链路层主要实现对多路数据的管理调度发送和接收，为虚拟通道和广播通道提供流量控制、QoS 服务、FDIR 错误恢复以及数据编号、组帧解帧等功能；网络层主要通过虚拟网络的划分实现路由各个端口虚拟通道之间的数据交换和路由；管理层主要实现各层命令参数的配置和状态的读取。

相比 SpaceWire 总线，SpaceFibre 在传输性能和传输速度上都有很大的提升，其性能对比见表 1。SpaceFibre 采用多路虚拟通道机制，能够实现对多路数据的传输；引入 QoS 机制，为各类数据流提供不同带宽和优先级的管理调度和传输服务；传输过程中为数据流提供确认，传输错误时能实现基于 FDIR 的错误恢复，并提供有保障的高可靠传输服务。由于其优越的性能，SpaceFibre 成为星载互联总线技术的研究热点，有重要的研究意义。

表 1 SpaceWire 和 SpaceFibre 性能对比

Tab. 1 Performance comparison between SpaceWire and SpaceFibre

性能	SpaceWire	SpaceFibre
虚拟通道	不支持	支持，最多 32 个
多通道	不支持	支持，最高达 50 Gbit/s
QoS	不支持	支持
FDIR	不支持	支持
编码方式	DS 编解码	8 B/10 B 编解码
传输速率	2 ~ 400 Mbit/s	最高 6.25 Gbit/s
传输确认	不支持	支持
传输介质	9 针 D 型连接线缆	光口光纤/串口线缆

2　节点系统设计

通过对 SpaceFibre 网络协议的研究,提出了一种基于 FPGA 的单通道 SpaceFibre 总线节点的系统设计方案。其中,数据链路层包括 4 路虚拟通道,通道层为单通道,物理层传输速度可达 3.125 Gbit/s。FPGA 内部提供了丰富的可编程逻辑资源,集成了高速串行接口吉比特收发器(Gigabit Transceiver X, GTX)核,便于编程实现和灵活移植,具备一定的通用性。节点系统涉及数据链路层、通道层、物理层以及管理层,架构如图 2所示。

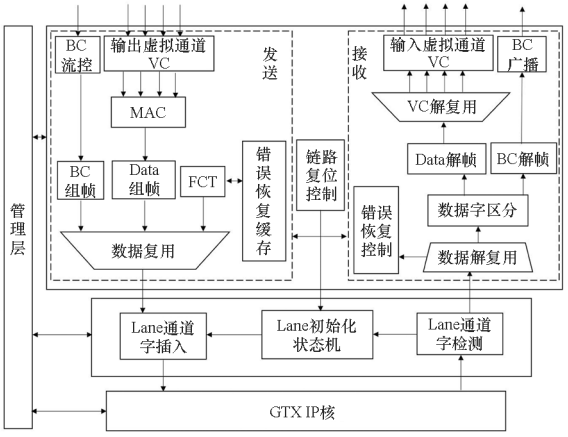


图 2　SpaceFibre 节点系统设计方案

Fig. 2　System design scheme of SpaceFibre node

2.1　物理层

物理层采用 FPGA 的 GTX IP 实现。在配置 GTX IP 时,参考时钟为 125 MHz,数据位宽设置为 32 位,串行速率可设置为 3.125 Gbit/s,对应的 GTX 用户时钟为 78.125 MHz。同时 GTX 实现了通道层的部分功能,如 8 B /10 B 编解码、字符及字同步、时钟矫正等功能。

2.2　通道层

通道层功能包括通道层控制功能和 GTX 所实现的通道层部分功能。通道层控制功能包括通道初始化状态机、通道字插入以及通道字检测。通道初始化状态机通过 INIT1、INIT2 以及 INIT3 控制字三次握手成功后,在 ACTIVE 状态下进行稳定的数据收发。通道字插入根据状态机指令生成通道控制字,并发送来自数据链路层的数据字。通道字检测根据接收到的通道控制字生成状态机指令,同时把接收的数据字传输到数据链路层。

2.3　数据链路层

数据链路层主要包括发送模块、接收模块,以及链路复位模块。

发送模块发送来自上层的数据和广播信息。由介质访问控制(Media Access Control, MAC)模块实现多路输出虚拟通道(Virtual Channel, VC),基于 QoS 机制的调度,对三类重要数据,即数据帧、广播帧和流量控制字(Flow Control Token, FCT)按照发送顺序依次编号,进行基于循环冗余校验(Cyclic Redundancy Check, CRC)的组帧。在数据发送的同时,备份到错误恢复缓存,以便接收方接收错误时进行重发恢复。数据复用模块按照数据优先级顺序对数据进行抢占式发送,无数据发送时发送空闲帧。

接收模块主要进行数据的接收,并根据接收状态生成确认控制字和非确认控制字(ACKnowledge/Negative ACKnowledge, ACK/NACK)。当广播帧、数据帧以及 FCT 控制字正确接收时,生成 ACK 发送到发送方,以删除发送方错误恢复缓存中已成功接收的备份;当接收时发生 CRC 错误或序列号错误,错误恢复控制模块生成 NACK 发送到发送方,以请求发送方的错误恢复缓存进行备份重发恢复。当接收数据从输入虚拟通道缓存中被读走,向发送方发送 FCT,告知空出的接收缓存空间,实现流量的控制。

链路复位模块通过接收管理层的复位命令,实现了链路复位的控制,完成各层的复位。

2.4　管理层

管理层包括命令配置模块和状态读取模块,分别对各层的命令配置寄存器进行实时的配置,对状态寄存器进行实时的读取。

3　关键技术及实现

3.1　基于轮询仲裁的多虚拟通道流量控制

SpaceFibre 节点的数据链路层支持多路虚拟通道,通过 FCT 控制字实现对每个 VC 的流量控制:每当被从输入 VC 中读取 64 个数据字后,接收方都会向数据发送方发送一个 FCT,每个 FCT 携带对应的 VC 编号;发送方收到 FCT 后,被告知在接收方相应的输入 VC 中空出了缓存空间,可继续发送数据,实现 VC 的流量控制。

由于多路输入 VC 中的数据可以同时被读出,会存在多路输入 VC 同时有空闲并同时向发送方提出 FCT 发送申请,因此会产生 FCT 发送申请冲突。为了解决多路 VC 的 FCT 发送申请冲突,对多路 VC 采用了轮询仲裁算法,轮询仲裁算法按照默认的轮询顺序,对各路 VC 的 FCT 发送

申请冲突依次完成仲裁。FCT 轮询仲裁算法步骤如下所示。

步骤 1:初始化设置默认轮询顺序。

步骤 2:收到多个 VC 的 FCT 申请,按轮询顺序响应第一顺位申请,发送对应的申请 $FCT\_req(i)$。

步骤 3:收到 FCT 已发送确认 $FCT\_ack(i)$,根据轮询顺序置 $i+1$ 为第一顺位。

步骤 4:响应申请直至发送完所有申请。

3.2 基于处理状态机的 QoS 机制

QoS 机制通过配置 QoS 参数实现对各个 VC 中数据的调度,为各路 VC 中的数据提供不同质量的传输服务。QoS 机制主要由 MAC 模块实现,过程如下:在满足发送条件的 VC 中选择优先级最高的 VC 中的数据进行组帧发送,每发送完一段数据帧后,更新各个 VC 的优先级并进行下一轮发送。由于 QoS 机制涉及诸多参数和变量,在每帧数据被发送调度之前需要对各参数变量进行计算和比较,因此参数变量的计算和比较的效率关系到数据帧的传输效率。

若节点中存在 i 个 VC,每个 VC 都会被分配对应的 QoS 参数:一个优先权 $R[i]$,一个带宽预留比例 $BwPer[i]$,以及一个 64 位宽的时间片调度表 $TimeSlot[i]$。优先权 $R[i]=0$ 时优先等级最高;预留带宽比例是为该 VC 所预留的可用带宽比例;时间片调度表中的 01 序列指示了当前时间片中该 VC 是否具有发送权;时间片长度可由上层网络定义。QoS 变量包括带宽信用($BwCredit$)和优先级($Precedence$),其中优先权和优先级的换算公式如式(1)所示;带宽信用和带宽预留比例,以及每次 VC 发送的数据量相关,如式(2)所示;优先级为优先权和带宽信用之和,综合考量了各个 VC 的优先权等级和剩余的带宽信用情况,如式(3)所示,各参数如表 2 所示。

$$Priority[i]=2B(Q-1-R[i])+B \tag{1}$$

$$BwCredit[i]=\sum_{n}\left(AvailableBw[n]-\frac{UsedBw[i]}{BwPer[i]}\right) \tag{2}$$

$$Precedence[i]=Priority[i]+BwCredit[i] \tag{3}$$

为了高效地进行 QoS 调度,设计了一个处理状态机:每次调度前,先选出当前时间片内允许发送数据的 VC 发送申请,比较各 VC 的优先级,并选出优先级最高的 VC 进行组帧发送;在该 VC 发送数据的同时统计所有 VC 链路上的 $AvailableBw[n]$ 和该 VC 发送的数据字 $UsedBw[i]$,数据段发送截止时数据发送量也已计数完毕;最后在发送帧尾

表 2 QoS 公式参数

Tab. 2 Parameter of QoS formula

符号	含义
$Priority[i]$	第 i 个 VC 的优先权换算优先级
$BwCredit[i]$	第 i 个 VC 的带宽信用
$Precedence[i]$	第 i 个 VC 的优先级
$AvailableBw[n]$	更新时间所有 VC 发送的数据字
$UsedBw[i]$	更新时间第 i 个 VC 发送的数据字
$BwPer[i]$	第 i 个 VC 的带宽预留比例
$R[i]$	第 i 个 VC 的优先权
B	最大带宽信用限制
Q	优先权等级的总数
n	数据字的个数

时进行 QoS 变量计算更新,以进行下一轮的优先级比较和调度,实现了与数据发送的并行处理。各 QoS 参数由管理层写入,变量的计算和比较各占一个时钟周期,发生在数据帧帧头和帧尾的发送过程,对数据帧的发送几乎无延迟。QoS 状态机流程如图 3 所示。

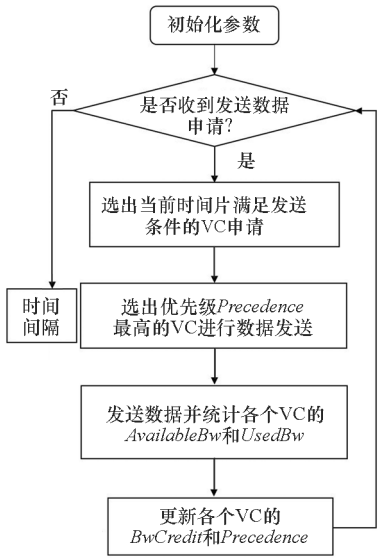


图 3 QoS 状态机流程

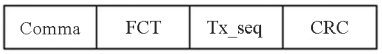
Fig. 3 QoS state machine flow

3.3 基于重发控制算法的 FDIR 机制

FDIR 机制对已发送的重要数据进行备份,以便进行数据的错误恢复。SpaceFibre 中三类重要数据分别为广播帧、FCT 和数据帧,优先级依次降低,如图 4 所示。在发送方发送数据时,对各类数据按照发送序列号 Tx_seq 编号依次备份,按优先级抢占式发送。接收方若成功接收数据,则发送正确确认控制字 ACK;若数据接收错误,则发送

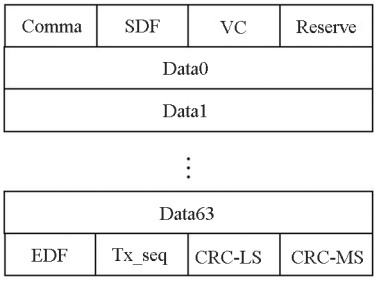
错误确认控制字 NACK。ACK 和 NACK 均携带接收方已成功接收的序列号 *Rx_seq*,发送方收到 ACK(*Rx_seq*)后清除备份中 *Tx_seq* 小于等于 *Rx_seq*的备份,即删除接收方已成功接收的数据备份;收到 NACK(*Rx_seq*)后先清除已成功接收的数据备份,再对未成功接收的数据备份按照优先级进行重发恢复,重发序号从 *Rx_seq* + 1 开始,备份重新进行校验组帧。

由于三类数据格式不同,发送时按照优先级抢占式发送,发送编号混杂,给按优先级重发恢复带来困难;在清除和重发时,需要对比接收序列号 *Rx_seq* 和各备份的发送序列号 *Tx_seq* 的大小,而发送序列号均在各备份的帧尾,直接比较会导致延时过大。因此,需要设计合理的存储结构和重发控制算法。



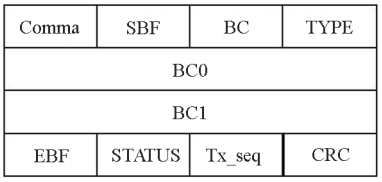
(a) FCT 控制字

(a) FCT control word



(b) 数据帧

(b) Data frame



(c) 广播帧

(c) Broadcast frame

图 4　SpaceFibre 的重要数据格式

Fig. 4　Important data format of SpaceFibre

为了提高 FDIR 的处理效率,本文提出了一种边发边备的并行分区存储架构,按数据类型对备份进行分区存储,在数据发送的同时进行备份。同时,针对该并行分区存储架构,提出了 FDIR 机制的主控状态机和一种高效的存储分区重发控制算法,实现对不同数据类型的备份、清除和快速重发恢复。

边发边备的并行分区存储架构按照数据优先级对三类数据进行分区备份,在数据正常组帧发送的同时备份到各自的存储分区,备份操作对数据发送不会造成延迟。当收到 ACK/NACK 进行备份清除或错误重发恢复时,根据不同数据形式设计了并行分区存储的内部架构,如图 5 所示。

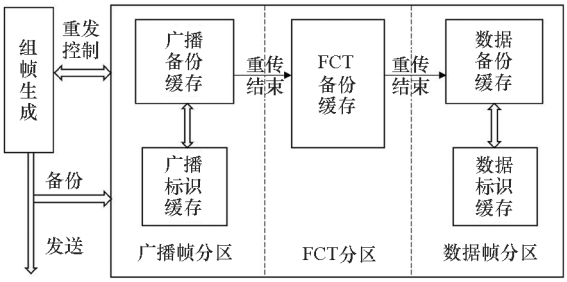


图 5　并行分区存储架构

Fig. 5　Parallel partitioned storage architecture

对于广播帧和数据帧,虽帧长不同但帧格式相似,故存储分区架构相同,分区内部包括备份缓存和标识缓存。其中,备份缓存用来存放数据备份,标识缓存用来存放备份标识。当每帧数据存入备份缓存时,对应的地址指针更新以指示该备份的存储地址,同时该备份对应的备份标识也存入标识缓存。为了高效地识别备份,设计了备份标识结构:{重发标识 + 备份帧尾地址 + 备份发送序列号}。重发标识是为了区分备份缓存未重发的数据和已重发且又再次被备份到缓存中的数据,从而避免多次重发;备份帧尾地址是对帧的存储位置进行标识,当收到 ACK/NACK 进行备份清除时,以便及时地释放备份中的缓存空间;备份发送序列号是为了在备份清除时无须读出完整数据备份,即可进行序列号比较。

对于 FCT 控制字备份,由于仅有一个数据字,序列号便于读取比较,故为了节省存储资源,仅设置一个 FCT 缓存,同时为了区分已重发数据和未重发数据,FCT 备份数据的高位加入重发标识。

针对该并行分区存储架构,设计了 FDIR 机制主控状态机,状态机如图 6 所示。针对各备份分区内部的存储架构,提出了重发控制算法,实现了各个分区的正确备份清除和错误备份重发恢复。重发控制算法步骤如下所示。

步骤 1:收到 ACK/NACK,得到 *Rx_seq*,与各分区中备份的 *Tx_seq* 比较,各分区并行执行备份删除:对数据帧和广播帧分区中的标识缓存,若 $Tx\_seq \leqslant Rx\_seq$,则删除该备份的标识,同时更新该备份数据缓存中的地址指针以释放缓存空间;对于 FCT 分区,直接比较备份缓存中的 *Tx_seq* 和

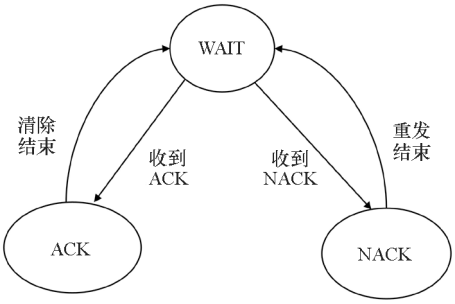


图6 FDIR 主控状态机

Fig. 6 Main state machine of FDIR

Rx_seq，若 $Tx\_seq \leqslant Rx\_seq$，则直接删除备份。当分区的备份数据 $Tx\_seq > Rx\_seq$ 时，无须删除备份。

步骤 2：数据备份清除完毕后，若主控状态是ACK，则返回正常备份状态；若主控状态是NACK，则进入数据重发状态。

步骤 3：在数据重发时，先翻转重发标识，更新发送序列号 $Tx\_seq = Rx\_seq$。首先根据优先级重发广播帧，重发的同时备份重发数据和备份标识，当备份标识中的重发标识均为翻转后的重发标识时说明重发结束，缓存中均为重发数据；然后重发 FCT，重发的同时备份 FCT，当备份标识中的重发标识均为翻转后的重发标识说明重发结束；最后重发数据帧，重发过程同广播帧。

步骤 4：重发结束后，返回数据正常备份状态。

通过重发控制算法，在清除备份时无须从缓存中依次读出备份，仅需删除标识缓存中的备份标识，并更新备份的地址指针，即可释放备份的缓存空间，操作简单高效，提高了备份缓存的利用率。同时，各分区在接收正常数据备份的过程中，能够并行地执行备份清除操作，对数据的正常发送无延迟影响。在错误重发时，通过重发控制算法能够对各类数据按照优先级依次重发，并可以对重发后的数据再次备份，利用重发标识能有效地区分原有备份和重发备份。针对不同的数据格式设计了不同的备份分区存储架构和重发控制算法，提升了 FDIR 的数据处理效率。

3.4 数据并行处理方案

3.4.1 并行 CRC

SpaceFibre 中数据帧、广播帧、FCT 及其他控制字均需要进行 CRC 校验，其中数据帧 CRC 校验生成式为 $G(X) = X^{16} + X^{12} + X^5 + 1$，原理如图 7 所示，CRC 校验范围包括 32 位的帧头、32 位的数据以及高 16 位的帧尾。广播帧的 CRC 生成式为 $G(X) = X^8 + X^2 + X + 1$，CRC 校验范围包括 32 位的帧头、32 位的数据以及高 24 位的帧尾。FCT 及其他控制字为一个 32 位的数据字，CRC 校验范围均为高 24 位，CRC 生成式与广播帧相同。

由于串行 CRC 算法需要占用多个时钟周期才能得到多位数据的 CRC，若通过对串行 CRC 校验进行时钟倍频，会因频率过高导致系统不稳定；采用查表法则会占用过多的存储资源，通过对 CRC 算法的比较[13-15]，选择采用基于公式递推法的并行 CRC 算法，能够在一个时钟下实现多位数据的 CRC 计算。以数据帧为例，若 32 位的数据为 $D[i]$ $(i = 0, \cdots, 31)$，在第 1 个时钟时对 $D[0]$ 进行校验，可以计算出该时钟下各 CRC 寄存器的表达式 $C[i]$ $(i = 0, \cdots, 15)$；以此类推，即可得到 32 个时钟后，对 32 位数据进行 CRC 校验后的各 CRC 寄存器表达式 $C[i]$。故根据 CRC 寄存器的初始值和各表达式，即可完成在一个时钟下 32 位数据的并行 CRC 计算。

由于不同数据格式的 CRC 校验范围不同，如数据帧既需要对 32 位数据进行 CRC 计算，又需

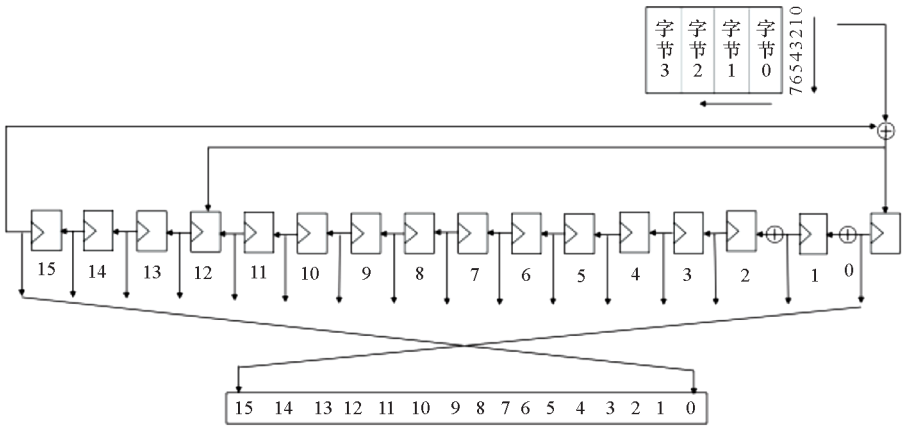


图7 CRC16 计算原理

Fig. 7 Principle of CRC16

要对帧尾的 16 位数据进行 CRC 计算,仅采用 32 位并行 CRC 表达式不能满足校验需求。故针对不同数据特征设计了不同的 CRC 校验方案:对数据帧采用了基于 CRC16 的 32 位数据和 16 位数据的组合并行 CRC 算法;对广播帧采用了基于 CRC8 的 32 位数据和 24 位数据的组合并行 CRC 算法;对 FCT 等控制字采用基于 CRC8 的 24 位数据并行 CRC。采用不同的组合并行 CRC 算法,一个时钟即可对多位数据进行 CRC 校验,校验时能应对数据校验长度的变化,提高了节点的数据处理效率。

3.4.2　*并行伪随机码*

在 SpaceFibre 中,空闲帧的内部由 32 位宽的伪随机数组成,伪随机数的生成式为 $G(X) = X^{16} + X^5 + X^4 + X^3 + 1$。不同于 CRC 校验,伪随机码的生成无须数据输入,仅需伪随机寄存器的串行输出序列。但若串行计算伪随机码则会占用过多的时钟周期,采用倍频会导致高速系统不稳定,故借鉴 CRC 校验中的方法,采用并行的公式递推法。由于没有外部输入,因此 32 位伪随机序列的表达式可以只由伪随机序列的初始值 $I(i)(i = 0,\cdots,15)$ 表示。16 位的伪随机序列寄存器表达式为 $L(i)(i = 0,\cdots,15)$,输出 32 位的伪随机序列表达式依次为 $P(i)(i = 0,\cdots,31)$。

伪随机码计算原理如图 8 所示,输出伪随机码 $P(i)$ 即为每个时钟伪随机序列寄存器 $L(15)$。以一个字节的伪随机码为例,第 1 个时钟后 $P(0) = L(15) = I(15)$;第 2 个时钟后 $P(1) = L(15) = I(14),\cdots,P(8) = L(15) = I(8)$。通过递推可以看出,在已知伪随机序列的初始值 $I(i)$ 的情况下,只需依次求得 8 个时钟下所有 $L(15)$ 的表达式,即可一次性求得一个并行的 8 位并行伪随机码。经过 8 个时钟后,伪随机序列寄存器的表达式如下:

$$\begin{cases} L(0) = I(8) \\ L(1) = I(9) \\ L(2) = I(10) \\ L(3) = I(11) \oplus I(8) \\ L(4) = I(12) \oplus I(9) \oplus I(8) \\ L(5) = I(13) \oplus I(10) \oplus I(9) \oplus I(8) \\ L(6) = I(14) \oplus I(11) \oplus I(10) \oplus I(9) \\ L(7) = I(15) \oplus I(12) \oplus I(11) \oplus I(10) \\ L(8) = I(0) \oplus I(13) \oplus I(12) \oplus I(11) \\ L(9) = I(1) \oplus I(14) \oplus I(13) \oplus I(12) \\ L(10) = I(2) \oplus I(15) \oplus I(14) \oplus I(13) \\ L(11) = I(3) \oplus I(15) \oplus I(14) \\ L(12) = I(4) \oplus I(15) \\ L(13) = I(5) \\ L(14) = I(6) \\ L(15) = I(7) \end{cases} \tag{4}$$

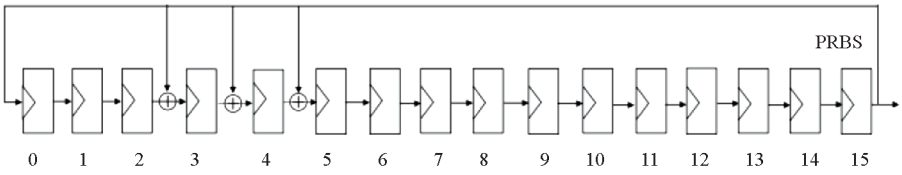


图 8　PRBS 计算原理

Fig. 8　Principle of PRBS

由于需要计算一组 32 位的伪随机序列码,为了便于进行公式推导,32 位伪随机序列的表达式可以分为 4 个字节的伪随机序列。在第 2 个字节的伪随机码计算时,以上述第 1 个字节经过 8 个时钟后的伪随机序列寄存器的值作为第 2 个字节推导时的初始值,再次对第 2 个字节进行如第 1 个字节的推导过程,求出 8 个时钟下 $P(i)(i = 8,\cdots,15)$ 的值。通过重复上述迭代推导,最终可以获得 32 个时钟内,每个时钟下各个输出的伪随机数表达式 $P(0) \sim P(31)$。因此,在一个时钟下,通过计算 32 位伪随机码的表达式,可以一次得出并行的 32 位伪随机码,提高了空闲帧中伪随机码的计算效率。此外,32 个时钟周期后的伪随机序列寄存器的值可以作为下一个 32 位伪随机码计算的初始值 $I(i)(i = 0,\cdots,15)$,将其直接代入伪随机码表达式 $P(i)$ 即可计算下一个 32 位伪随机码序列。迭代计算简单,表达式利用初始值寄存器和逻辑门表示,节省了时钟资源。

4　仿真分析

为了验证节点系统的设计方案,采用自顶向下的方法使用 Verilog 硬件描述语言完成了节点的设计,以 ModelSim10.5 作为仿真平台,对节点间的数据传输、FCT 申请的轮询仲裁、虚拟通道的 QoS 调度、FDIR 机制以及数据的并行处理等关键功能进行了仿真分析。

在数据传输时例化两个节点,并为发送方的

4 个虚拟通道构造 4 路数据源，搭建一个双节点的收发系统。将 4 路数据源分别写入 4 个对应的输出虚拟通道，经过发送方 QoS 调度、数据编号组帧以及 GTX 发送，在接收方通过 GTX 接收、数据校验解帧以及写入对应的输入虚拟通道等操作后，从接收方对应的输入虚拟通道读出。数据收发如图 9 所示，在两个节点之间实现了对应虚拟通道之间的数据传输功能。

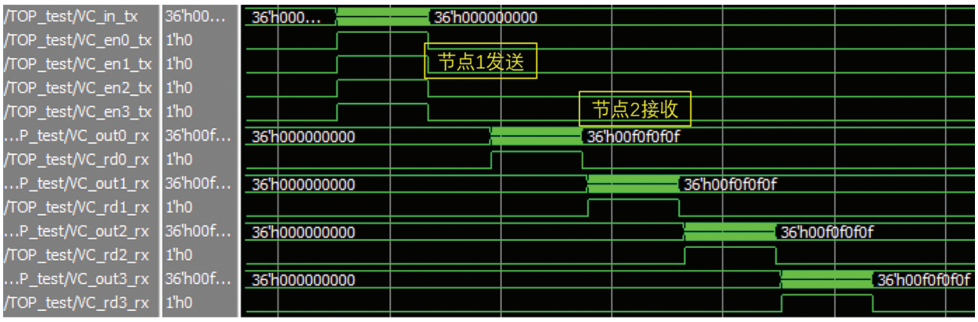


图 9 数据的发送和接收

Fig. 9 Sending and receiving of data

在输入虚拟通道中，当各路缓存中的数据同时被读走后产生了 FCT 申请冲突，为了解决该冲突，对 FCT 申请进行轮询仲裁。FCT 轮询仲裁如图 10 所示，当 4 路 VC 同时出现 FCT 申请冲突时，依次受理了 VC0、VC1、VC2、VC3 的 FCT 申请，实现了轮询仲裁功能，处理了 FCT 的发送申请冲突。

在基于 QoS 调度过程中，为了给不同数据流提供不同质量的传输服务，需要为各路 VC 配置不同的 QoS 参数。给出一种 QoS 参数配置方案：VC0、VC1、VC2 和 VC3 的优先权分别为 $R[0]=0$，$R[1]=1$，$R[2]=2$，$R[3]=3$；带宽百分比 $BwPer[0]=10\%$，$BwPer[1]=20\%$，$BwPer[2]=20\%$，$BwPer[3]=25\%$，VC 的调度表 $TimeSlot[i]$ 均设置为全 1，带宽信用限制 $B=10\ 000$。QoS 调度仿真如图 11 所示，MAC 模块先选择最高优先级 VC0 中的数据发送，并按照优先级依次发送 VC1、VC2、VC3 中的数据。数据发送后各 VC 对应的带宽信用 *BwCredit* 降低，而各 VC 的优先级 *Precedence* 也随之降低；数据不发送时，带宽信用和优先级缓慢增长。各 QoS 参数变化规律符合式(2)和式(3)，满足 QoS 的调度规则，同时 QoS 参数计算和比较对每帧数据的发送几乎无延迟影响，为各路数据源的传输提供了 QoS 服务。

在基于 FDIR 机制进行错误重发时，结合重发控制算法，该分区存储架构能实现高效的重发恢复。当发送方发送 *Tx_seq* 分别为 2、3、4、5、6、7 号数据帧的同时，进行了数据备份。接收方在成功接收 2、3、4 号后产生接收错误，并发送错误确认 NACK(*Rx_seq* = 4)，即已正确接收序号为前 4 的数据。发送方通过重发控制算法比较 *Tx_seq* 和 *Rx_seq*，删除了标识缓存中的标识释放备份缓存，并重发了备份缓存中未成功接收的 5、6、7 号数据帧。通过与普通重发控制算法的对比，如图 12所示，在对序号为 2、3、4 的数据备份进行清除时，仅用 3 个时钟即可实现备份的删除，并成功重发了后续备份，降低了 3 个数据帧长的延时，实现了高效的错误重发恢复。

在数据的并行处理中，通过计算 CRC 的 32 位并行表达式，实现了一个时钟内完成 32 位数据的 CRC 校验，其他 CRC 计算同理。通过计算推导出的并行 32 位伪随机序列寄存器和并行伪随机序列的输出表达式，实现了一个时钟即可计算出 32 位并行的伪随机数。并行数据计算如图 13 所示，并行的数据处理方案提高了节点的数据处理效率。

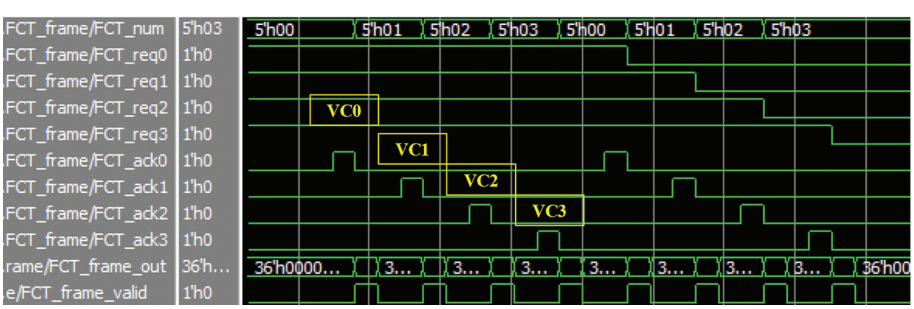


图 10 FCT 申请轮询仲裁

Fig. 10 Polling arbitration of FCT request

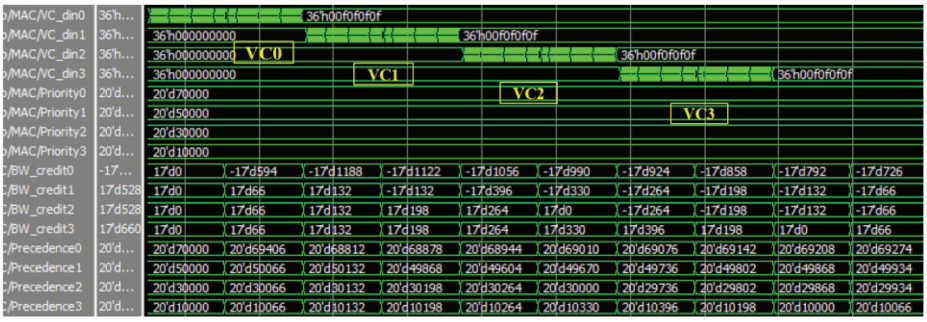
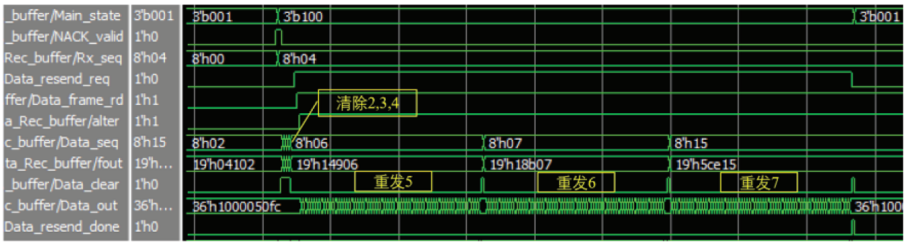


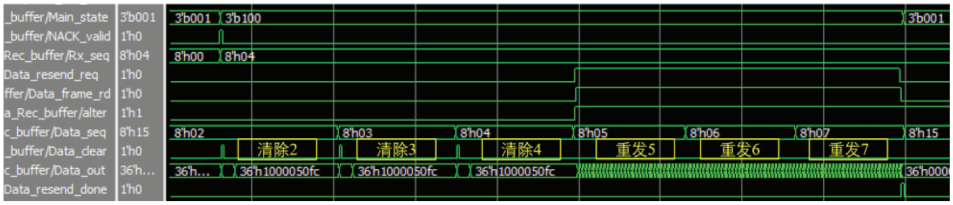
图 11　QoS 调度仿真

Fig. 11　Simulation of QoS scheduling



(a) 重发控制算法

(a) Retransmission control algorithm

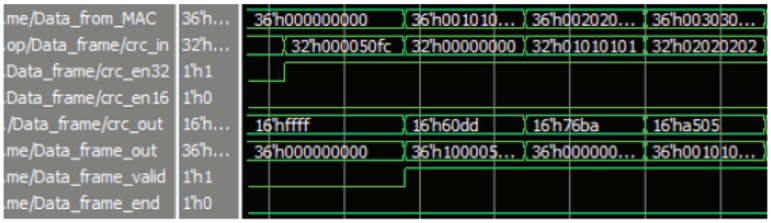


(b) 普通重发控制

(b) General retransmission control

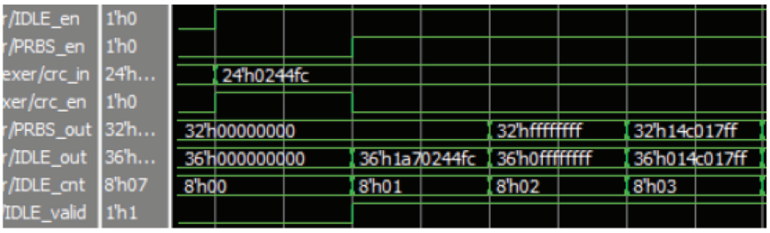
图 12　基于 FDIR 的错误重发恢复

Fig. 12　Error resend recovery based on FDIR



(a) 并行 CRC 计算

(a) CRC parallel computation



(b) 并行 PRBS 计算

(b) PRBS parallel computation

图 13　并行的数据处理

Fig. 13　Process of parallel data process

最后，在 Xilinx XC6VLX240T – 2FFG1156 FPGA 上实现了 SpaceFibre 节点系统的设计方案。通过搭建测试平台，把节点 IP 分别例化到两块 Virtex – 6 官方评估板的 FPGA 上，利用串行光纤实现了节点间的高速数据传输，板上主时钟频率为 78.125 MHz，串行传输速率可达 3.125 Gbit/s。该节点传输性能可以满足在大容量存储等高速数传项目中的传输要求。其资源占用情况如表 3 所示。

表 3 Virtex – 6 XC6VLX240T **资源占用**

Tab. 3 Resource utilization of Virtex – 6 XC6VLX240T

资源类型	总资源数	资源占用数	占用比例/%
Slice Registers	301 440	5 191	1.7
Slice LUTs	150 720	5 259	3.4
Block RAMs (36 Kbit)	416	8	1.9

5 结论

SpaceFibre 总线网络能够提供基于 QoS 机制和 FDIR 机制的高速数据传输，网络拓扑灵活，成了未来航天高速总线网络的研究热点。本文在对 SpaceFibre 协议研究的基础上提出了一种基于 FPGA 的 SpaceFibre 总线终端节点的系统设计方案，并针对节点中的关键问题和技术，设计实现了 FCT 申请轮询仲裁、基于 QoS 机制的调度处理、基于 FDIR 机制的错误重发恢复以及数据并行处理等关键技术；最后给出了数据传输和各个关键技术的仿真测试结果，验证了节点的功能，为后续 SpaceFibre 总线网络进一步实现和在轨应用奠定了技术基础。

参考文献(References)

[1] ECSS. ECSS-E-ST-50 – 11C—SpaceFibre-very high-speed serial link[S]. Noordwijk, the Netherlands: ESA-ESTEC Requirements & Standards Division, 2019.

[2] PARKES S, MCCLEMENTS C, MCLAREN D, et al. SpaceFibre implementation, test and validation [C]// Proceedings of International SpaceWire Conference (SpaceWire), 2014.

[3] YU B, GONZALEZ-VILLAFRANCA A, FERRER A, et al. Integrating STAR-Dundee SpaceFibre codec with TI TLK2711[C]// Proceedings of International SpaceWire Conference (SpaceWire), 2014.

[4] PARKES S, MCCLEMENTS C, MCLAREN D, et al. SpaceFibre implementation, test and validation [C]// Proceedings of International SpaceWire Conference (SpaceWire), 2014.

[5] PARKES S, FLORIT A F, VILLAFRANCA A G, et al. SpaceFibre network and routing switch[C]//Proceedings of IEEE Aerospace Conference, 2017: 1 – 7.

[6] PARKES S, FERRER A, GONZALEZ A, et al. A radiation tolerant SpaceFibre interface device [C]// Proceedings of the 5th International SpaceWire Conference, 2013: 123 – 128.

[7] JUNGEWELTER D, COZZI D, KLEIBRINK D, et al. AXI-based SpaceFibre IP core implementation[C]//Proceedings of International SpaceWire Conference (SpaceWire), 2014.

[8] SIEGLE F, HABINC S, BOTH J. SpaceFibre port IP core(GRSPFI)[C]// Proceedings of the 7th International SpaceWire Conference, 2016: 218 – 222.

[9] HIROKI H, MITSUNOBU K, KYOKO M, et al. Programmable SpaceFibre interface with NanoBridge field programmable gate array [C]//Proceedings of the 8th International SpaceWire Conference, 2018: 92 – 96.

[10] 徐曙清，王震，董瑶海，等. SpaceWire 与 SpaceFibre 高速总线发展与研究[J]. 上海航天，2014，31(1)：29 – 36.
XU Shuqing, WANG Zhen, DONG Yaohai, et al. Research and development of SpaceWire and SpaceFibre high speed bus[J]. Aerospace Shanghai, 2014, 31(1): 29 – 36. (in Chinese)

[11] 伊小素，王家兴，姜梦茹，等. SpaceFibre 星载网络服务质量实现研究[J]. 宇航学报，2019，40(2)：207 – 214.
YI Xiaosu, WANG Jiaxing, JIANG Mengru, et al. Research on realization of quality of service for SpaceFibre onboard networks[J]. Journal of Astronautics, 2019, 40(2): 207 – 214. (in Chinese)

[12] 张春熹，刘辉，伊小素，等. 基于 SpaceFibre 光网络的即插即用技术及性能分析[J]. 半导体光电，2018，39(6)：868 – 873.
ZHANG Chunxi, LIU Hui, YI Xiaosu, et al. Plug-and-play network and performance analysis based on SpaceFibre[J]. Semiconductor Optoelectronics, 2018, 39(6): 868 – 873. (in Chinese)

[13] 俞迅. 32 位 CRC 校验码的并行算法及硬件实现[J]. 信息技术，2007，31(4)：71 – 74.
YU Xun. The 32-bit cyclic redundancy check parallel algorithm and hardware implementation [J]. Information Technology, 2007, 31(4): 71 – 74. (in Chinese)

[14] 袁征，冶晓隆，郭超. 基于 FPGA 的 10G 以太网并行 CRC 设计[J]. 计算机工程与设计，2014，35(5)：1510 – 1515.
YUAN Zheng, YE Xiaolong, GUO Chao. Implementation of parallel CRC for 10 gigabit Ethernet based on FPGA[J]. Computer Engineering and Design, 2014, 35(5): 1510 – 1515. (in Chinese)

[15] 左飞飞，杜英森，刘剑霏. 基于递推法的 CRC – 32 校验码并行改进算法[J]. 探测与控制学报，2019，41(1)：97 – 101.
ZUO Feifei, DU Yingsen, LIU Jianfei. Improved parallel algorithm for CRC – 32 check code based on recursive method[J]. Journal of Detection & Control, 2019, 41(1): 97 – 101. (in Chinese)