面向多业务场景的AFDX端系统设计与实现

作者姓名 祝钊华

学校导师姓名、职称 潘伟涛 副教授

企业导师姓名、职称 王立民 高工

申请学位类别 工程硕士

学校代码 10701 学 号 20011210498
分 类 号 TN915 密 级 公开

西安电子科技大学

硕士学位论文

面向多业务场景的 AFDX 端系统设计与实现

作者姓名：祝钊华

领　　域：电子与通信工程

学位类别：工程硕士

学校导师姓名、职称：潘伟涛副教授

企业导师姓名、职称：王立民 高工

学　　院：通信工程学院

提交日期：2023 年 3 月

Design and Implementation of AFDX End System for Multiple Business Scenarios

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in Engineering

By

Zhu Zhaohua

Supervisor: Pan Weitao Title: Associate Professor

Supervisor: Wang Limin Title: Senior Engineer

March 2023

摘要

伴随着航空航天电子系统的发展，航空航天数据传输对网络的实时性、安全性、可靠性以及时延有了更高的要求，传统的数据传输方式已经无法适应日益复杂化的航空电子系统，迫切需要研发新技术取代原有技术，以支持更高速、更可靠的数据传输。航空电子全双工交互式以太网（Avionics Full Duplex Switched Ethernet，AFDX）作为新一代航空航天电子系统的传输网络技术，是一种特殊的确定性定制网络，具有高实时性、高安全性、高可靠性和低时延的特性，能够满足现代航电系统对网络传输的技术要求。虽然自研的第一代 AFDX 端系统在这些方面已经得到验证，但是由于用户对提升产品灵活性以及适用性的迫切需求，在第一代端系统的基础上又开展了新型 AFDX 端系统的研制工作，本文结合科研项目“AFDX 端系统知识产权（Intellectual Property，IP）核以及驱动软件研制外包”，重点研究了面向多业务场景的 AFDX 端系统的设计与实现。

本文首先概述 AFDX 技术的相关背景及研究现状；其次，重点介绍 AFDX 端系统的组成及相关关键技术；第三，基于端系统通信 IP 的设计需求，提出端系统 IP 核的实现架构，描述数据收发的处理流程，同时说明数据业务帧格式及表项内容含义；第四，对 AFDX 端系统 IP 核进行详细设计，阐述各个模块的功能与实现方法；第五，针对新型端系统 IP 核主要模块功能进行仿真验证，并搭建整体仿真环境，以测试多业务场景下数据的传输处理；最后，基于现场可编程门阵列（Field-Programmable Gate Array，FPGA）平台，完成新型端系统的板级测试验证，结果表明本文设计的新型 AFDX 端系统能够支持多业务场景下数据的传输处理，并且灵活性和适用性相较于第一代有显著提升，达到预期设计目的。

本文的创新点在于，第一，构建新型 AFDX 端系统 IP 核架构，支持通信端口及服务访问点数据的传输处理，将数据细分为七种类型，使端系统可面向多业务场景，提升 AFDX 端系统的适用性与灵活性；第二，增加故障注入功能，通过外部激励模拟航空航天场景下可能产生的故障，并直观地观察故障导致的后果；第三，支持十、百、千兆端口及多种功能的灵活配置，以匹配用户实际需求，同时，用户可以通过寄存器获知内部状态，监控 AFDX 端系统工作情况。除此之外，新型 AFDX 端系统还优化了第一代 AFDX 端系统沿用模块，以为日后 AFDX 端系统的升级换代做准备。

关 键 词： AFDX 端系统， 通信端口， 服务访问点， 故障注入， 配置

ABSTRACT

With the development of aerospace electronic systems, aerospace data transmission has higher requirements for real-time, security, reliability, and latency of the network. Traditional data transmission methods are no longer suitable for increasingly complex aerospace electronic systems, and there is an urgent need to develop new technologies to replace existing technologies to support faster and more reliable data transmission. Avionics Full Duplex Switched Ethernet (AFDX), as a transmission network technology for the new generation of aerospace electronic systems, is a special deterministic customized network with high real-time, high security, high reliability, and low latency characteristics, which can meet the technical requirements of modern avionics systems for network transmission. Although the self-developed first generation AFDX end system has been validated in these aspects, due to the urgent need of users to improve product flexibility and applicability, a new AFDX end system has been developed on the basis of the first generation end system. This thesis combines the scientific research project "Intellectual Property (IP) core and driver software development outsourcing of AFDX end system", The focus was on the design and implementation of an AFDX end system for multiple business scenarios.

This thesis first outlines the relevant background and research status of AFDX technology; Secondly, focus on introducing the composition and related key technologies of the AFDX end system; Thirdly, based on the design requirements of the end system communication IP, propose the implementation architecture of the end system IP core, describe the processing flow of data transmission and reception, and explain the format of data business frames and the meaning of table entries; Fourthly, conduct a detailed design of the AFDX end system IP core, explaining the functions and implementation methods of each module; Fifthly, simulate and verify the main module functions of the new end system IP core, and build an overall simulation environment to test data transmission and processing in multiple business scenarios; Finally, based on the Field Programmable Gate Array (FPGA) platform, board level testing and verification of the new end system were completed. The results showed that the new AFDX end system designed in this thesis can support data transmission and processing in multiple business scenarios, and has significantly improved flexibility and applicability compared to the first generation, achieving the expected design objectives.

The innovation of this thesis lies in the construction of a new AFDX end system IP core architecture, which supports the transmission and processing of comport port and service access point data. The data is divided into seven types, making the end system suitable for multiple business scenarios and improving the applicability and flexibility of the AFDX end system; Secondly, add fault injection function, simulate possible faults in aerospace scenarios through external excitation, and visually observe the consequences caused by faults; Thirdly, it supports flexible configuration of 10, 100, and gigabit ports and multiple functions to match the actual needs of users. At the same time, users can obtain internal status through registers and monitor the working status of the AFDX end system. In addition, the new AFDX end system has also optimized the modules used in the first generation AFDX end system to prepare for future upgrades and upgrades of the AFDX end system.

Keywords: AFDX End System, Comport, Service Access Point, Fault Injection, Configuration

插图索引

符号对照表

符号	符号名称
bit	比特位
Byte	字节
Gbps	吉比特每秒
Kbps	千比特每秒
Mbps	兆比特每秒
MHz	兆赫兹
ms	毫秒

缩略语对照表

缩略语	英文全称	中文对照
AFDX	Avionics Full Duplex Switched Ethemet	航空电子全双工交换式以太网
ARCNET	Auxiliary Resource Computer Network	广泛局域网
BAG	Bandwidth Allocation Gap	带宽分配间隙
BD	Buffer Description	缓存描述符
BE	Best Effort	尽力传输
CDR	Clock Data Recovery	时钟数据恢复
COM	Comport	通信端口
CRC	Cyclic Redundancy Check	循环冗余检查
DSP	Digital Singnal Processor	数字信号处理器
EMIF	External Memory Interface	外部存储器接口
FDDI	Fiber Distributed Data Interface	光纤分布式数据接口
FIFO	First Input First Output	先进先出队列
FPGA	Field-Programmable Gate Array	现场可编程门阵列
GMII	Gigabit Media Independent Interface	千兆介质无关端口
IEEE	Institute of Electrical and Electronics Engineers	电气和电子工程师协会
IP①	Intellectual Property	知识产权
IP②	Internet Protocol	网络协议
MAC	Media Access Control	媒体介入控制
PCS	Physical Coding Sublayer	物理编码子层
PHY	Physical Layer	物理层
Q	Queue	队列
RAM	Ramdom Access Memory	随机存取存储器
RC	Rate-Constrained	速率受限
RD	Runing Disparity	运行不一致性

① IP 示意为知识产权时，通常与“核”合用，即“IP 核”

② IP 示意为网络协议时，不与“核”合用

S	Sample	采样
SAP	Service Access Point	服务访问点
SerDes	SERializer-DESerializer	串行解串器
SGMII	Serial Gigabit Media Independent Interface	串行千兆位媒质独立
SN	Sequence Number	序列号
SNMP	Simple Network Management Protocol	简单网络管理协议
TFTP	Trivial File Transfer Protocol	简单文件传输协议
TSN	Time-Sensitive Network	时间敏感网络
TTE	Time Triggered Ethernet	时间触发以太网
UDP	User Datagram Protocol	用户数据报协议
VL	Virtual Link	虚拟链路

目　录

第一章 绪论

1.1 研究背景

自 1973 年以太网[1]发明以来，数十年间，以太网技术得到迅速发展。从诞生到以太网技术的标准化、规范化，1983 年电气和电子工程师协会（Institute of Electrical and Electronics Engineers，IEEE）802.3 标准[2]顺利通过并发布，再到以其高灵活性、结构简单、易于互联及维护等特点[3]，以太网技术逐渐取代令牌环网、光纤分布式数据接口（Fiber Distributed Data Interface，FDDI）及基于令牌总线的广泛局域网（Auxiliary Resource Computer Network，ARCNET）等传统局域网技术[4]，已然成为当今应用最为广泛的局域网技术。与此同时，伴随着用户需求的不断增长，以太网技术势必走上继续提升带宽、速率及可靠性等技术指标的道路，故而多个 IEEE 衍生标准也相继出现，为更复杂及多样化的网络设备提供解决方案。以太网技术的成功应用，也使其成为各个领域的重点研究对象，其中就包括航天航空领域[5]。

1977 年，ARINC429 总线协议被提出，协议定义了航空电子设备及相关系统间的数字信息传输要求[6]，因为具有结构简单、传输可靠、安全性较好等特点，所以被广泛应用于主流民用飞机上，例如空客 A310、A330，波音 B737、B757、B767，国产大型客机 C919 等等[7]，但问题在于 ARINC429 总线协议是单向传输的，并且传输速率最高只能支持 100Kbps，不适用于高速双向传输的应用场景。20 世纪 90 年代，AIRBUS 公司综合考虑成本、适用性等各项因素后，决定升级现有航空总线协议，而此时全双工交换式以太网技术已日渐成熟且潜力巨大，同时也能匹配 ARINC429 总线协议及航空电子系统，最终被引入至航天航天领域并更改，以满足新一代飞机的需求。机载总线协议经历了从 ARINC429 到 ARINC629 再到 ARINC664[8]的发展历程，结构由网状变为星型，传输速率从 100 Kbps 提升至 100 Mbps，性能指标有了显著的提升[9]。

从单一传输到双向传输，从 Kbps 的传输速率到 Mbps 甚至是 Gbps 的传输速率，从网状结构到星型结构，这与航空电子系统的高速发展脱不开干系。AFDX，亦即航空电子全双工交换式以太网[10]，完全符合 ARINC664 Part7 协议，凭借高可靠性、易拓展、低时延、高安全性等优点[11]，AFDX 技术在新一代大飞机的机载网络系统应用上占据着一席之地，为持续发展的航空电子系统添砖加瓦。AFDX 网络包括 AFDX 端系统及 AFDX 交换机，其中 AFDX 端系统承载着收集及处理数据的作用，包含多个关键技术以匹配航空电子系统的特定要求，是整个系统中极为重要的一环，本文即着重研究 AFDX 端系统的构建与实现。

1.2 国内外研究现状

ARINC 664 协议之所以被提出，缘由是 AIRBUS 公司想发展新一代飞机，A380 即第一个采用 AFDX 技术的飞机型号。随着民用航空通用标准组织对该项技术协议的认可，国外针对 AFDX 相关技术的研究更是步履不停，AFDX 端系统涵盖了多项技术，例如调度算法[12]、冗余管理[13]、流量控制[14]等等，研究点纷杂。同时，人们也仍在继续寻求更优的解决方案，用于提升时延、抖动[15]、速率等技术指标。现如今，AFDX 技术在国外的应用十分成熟，能够完善地进行应用、测试以及验证，例如 AIM 公司已经可以生产性能十分优秀的 AFDX 端系统、网络测试平台等设备，以测试并分析系统的各项性能指标[16]。甚至有公司在 AFDX 技术的基础上继续发展，研究时间触发以太网（Time-Triggered Ethernet，TTE[17]）、时间敏感网络（Time-Sensitive Network，TSN[18]）等技术，从而更高速、更实时、更灵活地传输数据。

相比于国外的研究，受限于技术落后及政策倾斜，国内对 AFDX 技术的研究显得滞后，但随着近几年航空航天领域被不断重视，结合国家发展国产大飞机的背景，AFDX 技术的应用潜力也逐渐被相关从业者发现，国内多个研究所和各个高校团队都在积极探讨 AFDX 技术在我国航空电子系统应用的可行性[19]，并已有团队成功搭建出 AFDX 网络地面仿真平台，验证系统的各项技术要求及性能指标，与传统机载总线协议相比，结果显示 AFDX 技术在性能表现上更为突出。但不可否认的是，国内 AFDX 系统相应测试设备仍有所缺失，研究人员无法进行更深入的测试验证，不过相信在不久的将来，随着研究的继续深入，技术的提升，测试、验证完整性的提高，AFDX 网络终将能够被更为广泛地应用于国产大飞机上，以满足用户更高的需求。另外，针对航空航天领域，在 AFDX 技术的基础上，国内也与时俱进地研究 TTE、TSN 等技术。

综上所述，尽管国内外针对 AFDX 技术的研究进程有所差异，但我国正在逐步提升实力，缩小与技术发达国家之间的差距，稳扎稳打，奋力前行，以实现 AFDX 技术在民用飞机上的广泛运用。本论文即研究 AFDX 端系统，以期在时代发展的洪流中贡献一份力量。

1.3 AFDX 网络介绍

AFDX 是一种特殊的确定性定制网络[20]，具有高实时性、高安全性、高可靠性和低时延的特性[21]。在 IEEE 802.3 的基础上，为符合航空电子系统的特殊要求，对原协议做相应更改提出 ARINC 664 协议，ARINC 664 协议分别对端系统以及交换机规范提出了要求[22]。针对端系统，规范中从 MAC 层、IP 层及 IP 层以上、网络层三方面，对虚拟链路、端系统性能、冗余概念等技术及性能[23]都做出了详尽的解释与说明。

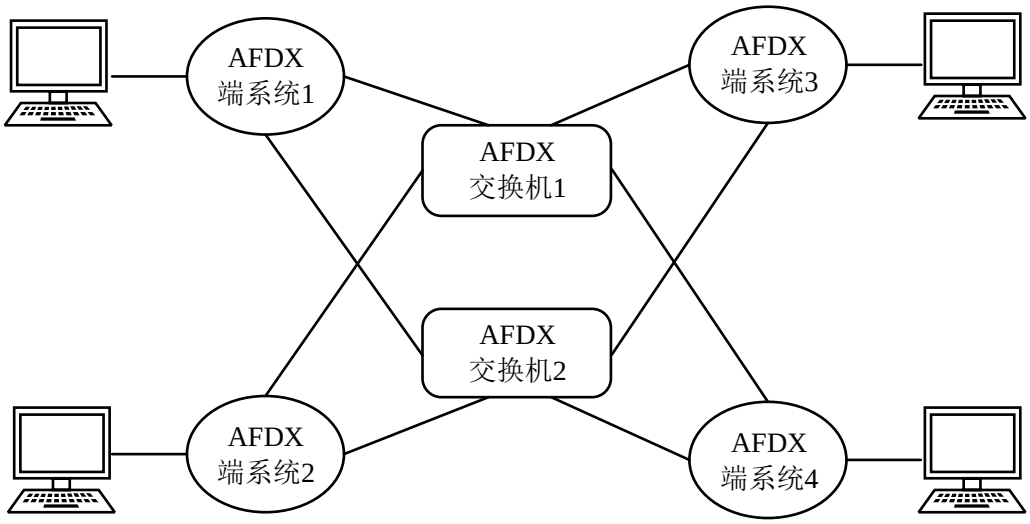

图1.1 AFDX 网络拓扑连接图

典型的 AFDX 网络拓扑结构如图 1.1 所示，完整的 AFDX 网络由 AFDX 端系统、AFDX 交换机以及链路组成。其中 AFDX 端系统通常向上连接终端设备，如主机、收音器、摄像头等。为实现冗余功能，一个端系统至少有两个端口与不同交换机进行数据交互。不同设备之间的通信链路称为虚拟链路（Virtual Link ，VL[24]），一条虚拟链路只能由一个源端节点发起，可传输至一个或多个目的端节点。

AFDX 网络中有两种业务类型数据：速率受限（Rate-Constrained，RC）业务和尽力传输（Best Effort，BE）业务[25]。

速率受限业务，顾名思义，即在传输时业务受带宽影响，为保障不同虚拟链路号数据流的业务带宽[26]，对隶属同个虚拟链路号下的 RC 数据帧，在发送完一帧之后至少要等待一定时间间隙才能发送下一帧，这个时间间隙被称为带宽分配间隙（Bandwidth Allocation Gap，BAG[27]），BAG 的数值由用户配置。

尽力传输业务，通常被理解为标准以太网业务，优先级次于速率受限业务，在 RC 数据帧的传输间隙之间插空传输，BE 业务没有 BAG 的限制，也不存在多个虚拟链路号，只要有空闲间隙即可持续传输。

1.4 主要工作与内容安排

本文依托于实验室科研项目“AFDX 端系统 IP 核及驱动软件研制外包”，围绕 AFDX 技术的出现原因，对研究背景进行调研，通过国内外研究现状的对比表明本文的研究意义，进而根据科研项目实际需求和 ARINC 664 协议提出新型 AFDX 端系统架构，并详细地介绍帧格式、表项含义和各模块的具体实现方式，在其中融入本文的创新点。之后通过代码书写和工程创建，结合创新点及主要功能点，对新型 AFDX 端

系统进行仿真和上板验证，结果表明设计达到预期设计目的。本文各章节安排如下所示：

第一章：绪论。本章首先介绍以太网的发展历史，继而与航空电子系统总线的发展历程相关联，引出 AFDX 技术，以时间为维度，对本课题的研究背景做了简明扼要的说明，然后针对国内外研究现状分别进行阐述，以表明课题研究的重要意义，接着简单介绍 AFDX 网络的组成，最后给出本文的主要工作以及内容安排。

第二章：AFDX 端系统组成及相关关键技术。本章首先介绍 AFDX 端系统的组成，接着概述与端系统相关的关键技术，即串行千兆位媒质独立（Serial Gigabit Media Independent Interface，SGMII）相关技术以及媒体介入控制（Media Access Control，MAC）层、 网络协议（Internet Protocol，IP）层和用户数据报协议（User Datagram Protocol，UDP）层规范。

第三章：面向多业务场景的 AFDX 端系统 IP 核架构设计。本章首先根据项目的设计需求，提出端系统 IP 核的总体架构，并通过对比新旧两代端系统以展现设计的改进点，最后给出端系统所规定的帧格式和所用表项的内容含义，以方便读者对后续行文的理解。

第四章：面向多业务场景的 AFDX 端系统 IP 核的设计与实现。本章分五个部分介绍端系统 IP 核的构成，分别为发送侧模块划分、接收侧模块划分、故障注入模块、配置及监控模块以及接口模块，详细介绍各组成部分的功能及具体实现，并在其中融入本文的创新点。

第五章：仿真验证与板级验证。本章首先介绍为面向多业务场景的 AFDX 端系统 IP 核搭建的仿真环境，并结合创新点，对 IP 核的主要功能点进行仿真验证。接着介绍测试环境的搭建，采用上板测试的方式验证实际环境下的数据流是否能被正常处理，最后总结测试中遇到的问题并给予相应解决办法。

第六章：总结与展望。本章对参与完成的科研项目“AFDX 端系统 IP 核及驱动软件研制外包”进行工作总结，并结合过程中发现的不足，对未来的研究进行展望，以盼日后设计能够被继续改进。

第二章 AFDX 端系统组成及相关关键技术

本章首先从数据处理流程和系统业务两方面简单介绍 AFDX 端系统。鉴于板卡限制，实现网口通信是项目得以顺利进行的前提，故而本章接着概述 SGMII 接口中串行解串器（SERializer-DESerializer，SerDes[28]）、物理编码子层（Physical Coding Sublayer，PCS[29]）相关技术，再从工程层面给出应用经验。同时，本章还将阐述 MAC 层、IP 层及 UDP 层协议规范。

2.1 AFDX 端系统简介

如图 2.1 所示，为本文所依托科研项目中航空电子系统音频控制单元的整体架构，其中虚线框内是 AFDX 端系统，负责数据的封装处理等操作。

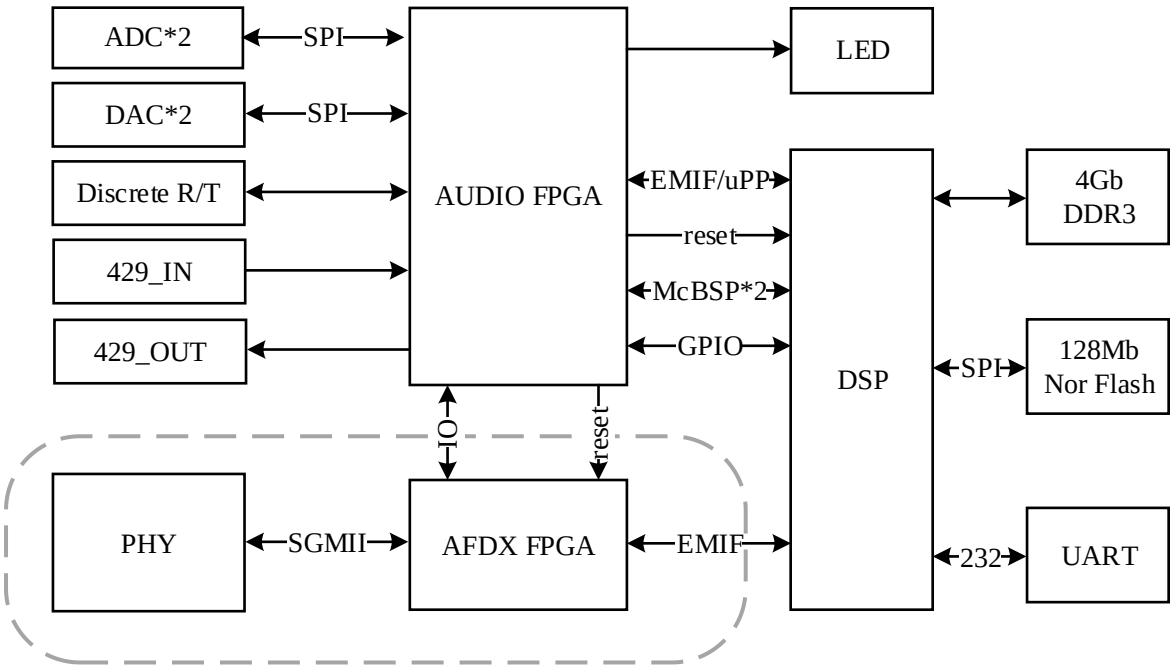

图2.1 音频控制单元整体架构

AFDX 端系统通过外部存储器接口（External Memory Interface，EMIF）与数字信号处理器（Digital Singnal Processor，DSP）相连，采用 SGMII 接口和物理层（Physical Layer，PHY）芯片相连。音频控制单元接收处理过程是发送处理过程的逆操作，在此仅简单介绍发送处理过程：

（1）模数转换器采样音频信号，经由音频处理芯片传输至 DSP；

（2）DSP 处理音频信号，并把数据流切分为数据帧，之后通过 EMIF 接口把数据下发至 AFDX 端系统；

（3）端系统负责数据帧的封装、调度等工作，最后采用 SGMII 接口把完整数据帧递交给 PHY 芯片。

AFDX 端系统承载着两类系统业务，第一类是采样语音业务，简单理解，每隔相同的时间间隙，采样器都会收集定量语音信息传输至音频控制单元内部处理，这类业务可作为通信端口（Comport，COM[30]）下的采样（Sample ，S）数据供端系统处理。

另一类是语音控制业务，用户可以通过操作按钮人为地改动语音的音量、音高等信息，音频控制单元内部生成控制信息后由 DSP 切分为数据帧进行传输。不同于采样语音业务，由于语音控制业务的产生不具备采样性质，且在生成时就已经携带帧头，故而可作为服务访问点（Service Access Point，SAP[31]）数据供端系统处理。

针对端口类型和 COM 端口数据类型，有必要作进一步说明。ARINC 664 part 7 就航空电子服务有特别说明，端系统可以利用两种类型的端口传输数据帧：COM 端口和 SAP 端口。SAP 端口用于进行简单文件传输协议（Trivial File Transfer Protocol ，TFTP[32]）的传输以及与兼容网络通信。

COM 端口提供两类服务：采样服务和队列（Queue ，Q）服务[33][34]。其中采样数据是实时更新的数据，即如果旧的数据在新的数据来临前没有被发送，则会被覆盖，这种模式适用于需要被不断更新且有采样性质的数据，比如传感器的数据等。队列数据是指端系统内部存在一个缓存区，能存储几个完整的帧数据，新消息来临时旧数据并不会马上被覆盖，适用于每一帧都同等重要的数据。

本文所依托科研项目仅需端系统处理采样语音业务和语音控制业务，为扩大端系统的应用前景，在设计 AFDX 端系统 IP 核时，会实现 COM 端口 S 数据、COM 端口 Q 数据和 SAP 数据处理功能，同时又进一步把数据细分为七种类型（详见 3.4.1 小节），使 AFDX 端系统可面向多业务场景。

2.2 AFDX 端系统相关技术概述

板级验证采用的板卡 PHY 芯片仅支持 SGMII 格式的用户侧接口，综合考量，最后采用 SerDes IP 核与 PCS IP 核实现数据的编解码与并串转换，以完成数据的高速可靠传输[35][36]，本节将介绍 SGMII 包含的部分相关技术，包括 8B/10B 编码[37]、时钟恢复、时钟修正、串行和解串、预加重和线路均衡[38]，之后总结 SGMII 接口的应用经验。端系统的主要功能是封装、处理数据流等，所以组装数据帧时必须考虑 MAC 层、IP 层和 UDP 层的协议规范，本节也讲解了数据帧处理时需要遵循的协议规范。

2.2.1 SGMII 部分相关技术

本文描述的项目在 Libero 及 Vivado 平台均已实现，两者皆能采用代码设计和图

形化设计，板卡 PHY 芯片用户侧逻辑与网口之间的接口为 SGMII，实现网口通信是项目得以顺利进行的前提条件，所以在给出新型 AFDX 整体架构之前，有必要研究 SGMII 接口包含的技术点。如图 2.2 所示，为 Libero 平台下的 SGMII IP 核构成，为实现冗余功能，收发器启用两个通道，分别与 PCS IP 核相连。在此先简单比较 Libero 及 Vivado 平台间的区别。

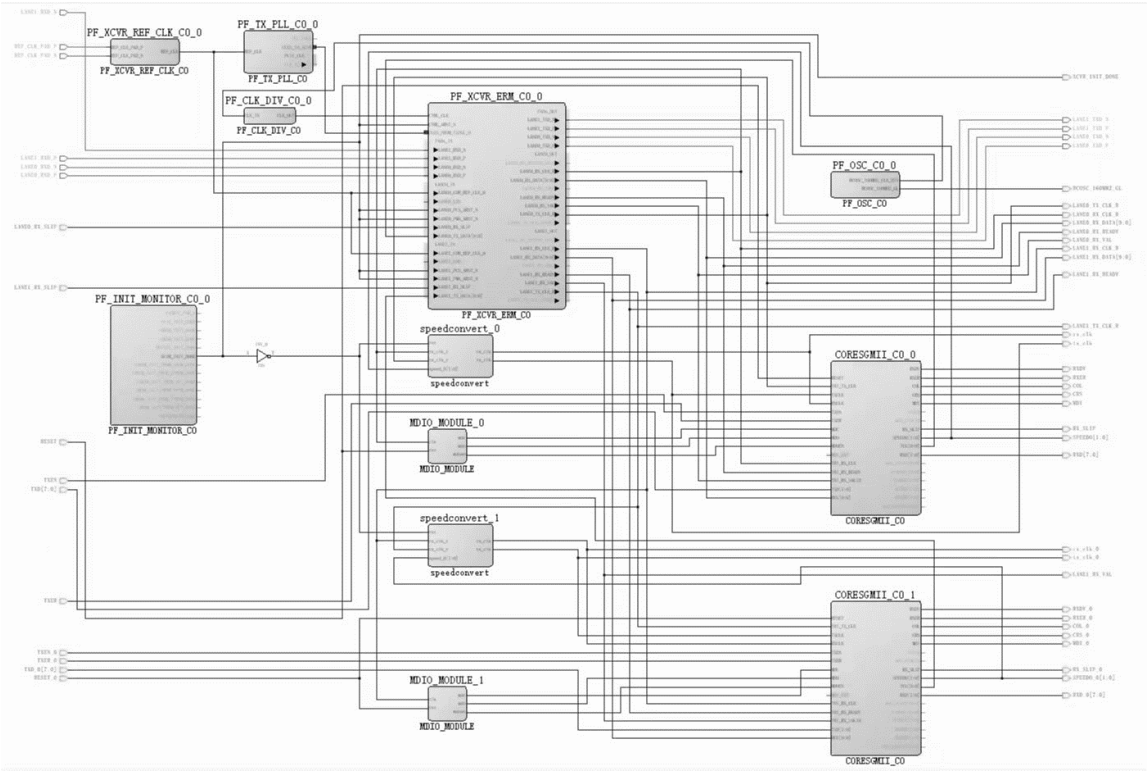

图2.2 SGMII IP 核构成

区别一：Libero 平台的实现板卡为 Microsemi，而 Vivado 平台则采用 Xilinx 板卡，两者 IP 库不同，用 Vivado 构建工程时，使用的 IP 核已包括 SerDes 及 PCS 功能，只需输入 GMII 格式数据，即可输出串行数据流至网口。而 Libero 平台无类似 IP 核，设计时只能采用收发器和 PCS 相连共同实现 SGMII 接口，要求使用者熟悉 SGMII 相关技术。

区别二：尽管 Microsemi 板卡固化时间比 Xilinx 板卡短，但 Libero 平台仿真、上板和调试需调用多款软件，给用户使用带来不便，而使用 Vivado 平台完成这些工作则显得方便许多，尤其是 SGMII IP 核的仿真，Vivado 平台能较快观察到波形图，而 Libero 平台则需等待漫长的初始化过程。

区别三：由于 Libero 使用多个 IP 核共同构建 SGMII，故而有利于使用者更深刻地从工程层面认识 SGMII IP 核，且 Libero 收发器在配置时有默认参数，不需用户手动配置内部功能，不过如果用户对 IP 核外部接口认识不深，易发生差错，此时通过

上板抓信号的方式纠正会很麻烦，因为一次只能选取一个时钟域的信号。

综上所述，尽管 SerDes IP 核及 PCS IP 核为黑盒代码，不属于用户可编辑硬件逻辑，但若要正确使用，对相关技术的研究与了解必不可少。另外，SGMII 接口是 AFDX 端系统 IP 核的一部分，清晰地认知该接口便于设计者解决仿真上板可能遇到的问题，也便于使用者对端系统有整体的掌握。

（1）8B/10B 编码

电容阻抗会随频率的提高而降低，数据流中如果存在长连“0”或“1”时，这段数据可以被认为是直流信号，经过电容后信号强度降低，接收端也无法从这种信号里提取足够的定时信息，为信号的正确识别引入不确定性。8B/10B 编码将 8 bit 数据编为 10 bit，组合情况从 256 种提升至 1024 种，只需从中选取“0”、“1”数目相近的编码取代不利数据，即可保持较好的直流平衡，提高数据传输的可靠性。

8B/10B 编码中分为两种字符：数据字符（D.X.Y，X、Y 分别是 8 bit 数据低 5 bit 和高 3 bit 的十进制表示）和控制字符（K.X.Y），控制字符用于指示数据帧的起始、结束等状态和不同设备之间的同步对齐。将 8 bit 数据拆分成两部分，前 3 bit HGF 和后 5 bit EDCBA，分别编为 4 bit fghj 和 6 bit abcdei。

8B/10B 编码还应考虑已编码数据“0”、“1”个数的差值，即不一致性，它仅有三种情况：“+2”、“0”和“-2”，“-2”表示“0”的个数比“1”少两个。累计的已编码数据不一致性称为运行不一致性（Runing Disparity，RD），只有“+1”和“-1” 两种情况，分别表示“1”比“0”多和“0”比“1”多。

（2）时钟恢复

采用 SGMII 接口传输数据时，高速串行数据流中已携带时钟信息，SerDes 中采用时钟数据恢复（Clock Data Recovery，CDR）技术进行时钟恢复[39]。

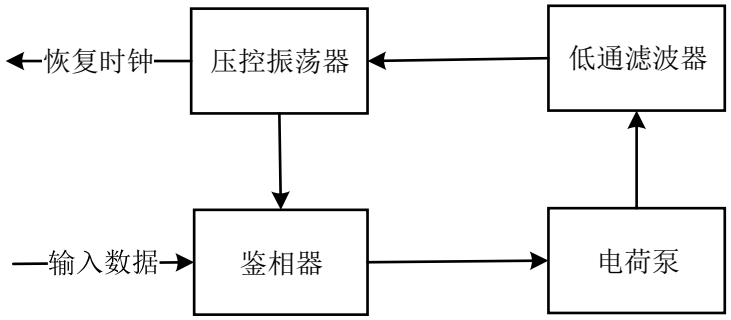

图2.3 时钟恢复原理框图

如图 2.3 所示，时钟恢复的工作原理是鉴相器比较高速串行输入数据和压控振荡器输出，产生的相位误差经过电荷泵后转化为电流，再通过低通滤波器转化为控制电压，驱动压控振荡器向目标频率逼近，最终达到锁定状态[40]。

（3）时钟修正

Serdes IP 核对内部工作时钟有着严格的要求，一般而言经 CDR 恢复的时钟不能直接作为传输时钟，虽然参考时钟和恢复时钟之间的频差很小，但由于高速串行流恢复的时钟至少是 ns 级别的，每秒可能增加或者缺失成千上万个符号。

时钟修正功能对控制字符的利用可以解决频差问题，具体到实现上可以检测接收端先进先出队列（First Input First Output，FIFO）的状态：倘若 FIFO 将满，则下一个控制字符来临时不写入 FIFO；倘若 FIFO 将空，则下一个控制字符来临时被两次写入至 FIFO，如图 2.4 所示：

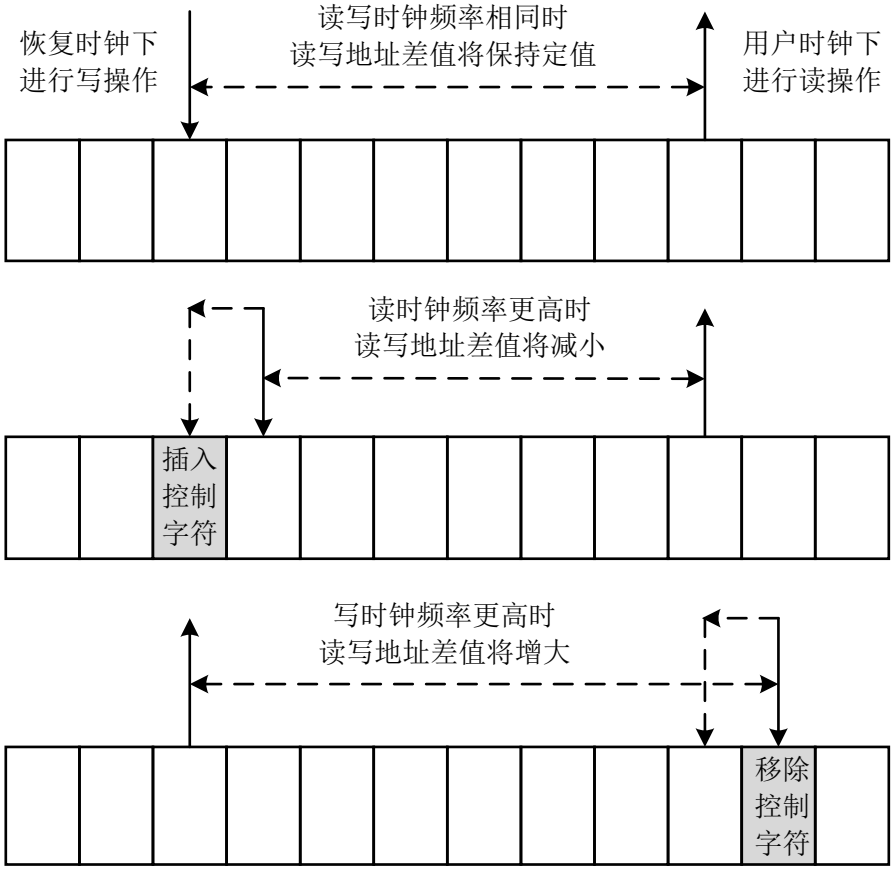

图2.4 时钟修正示意图

（4）串行和解串

SerDes IP 核内部在实现并串转换时，2n bit 的并行数据通常会被串行器转换成 n bit，以降低均衡器的工作频率，转换完成的数据经均衡器滤波后再进行下一步串行化。串行器和解串器采用双倍速率同步动态随机存储器，用面积换取速度从而减少高频率电路以降低电路噪声。串并转换同理，不再赘述。

（5）预加重

如果串行数据流连续的相同比特后跟着若干反相比特，会造成符号间干扰。连续的相同比特将电路中的电容充电完全，而后的反相比特由于持续时间较短，无法反相补偿，电压值极有可能会被忽略。针对这个问题，SerDes IP 核内部会在转变开始时加入过量驱动，并且在任意的连续相同数值时间内减少驱动量，降低差错率[41]，这种技术被称为预加重。

（6）线路均衡

均衡器一般位于 SerDes 的前端，其主要作用是补偿因频率差异导致的阻抗/衰减差异。均衡器大致可以分为两类：无源均衡器和有源均衡器。前者能够被看作是滤波器，其频率响应能够补偿线路传输时的衰减，后者则能够被看作是和频率息息相关的放大器或者衰减器，具体又细分为两种，第一种被称为固定形式有源均衡器，顾名思义，它的频率响应和输入数据流无关，可由用户配置，适用于恒定通道，第二种被称为自适应均衡器，它的频率响应和输入数据流有关，适用于可变通道。

进行 SGMII 的调试时，PCS IP 核由于只涉及 8B/10B 编码，在 10 bit 数据收发侧进行回环测试，通过比对 8 bit 收发数据的一致性即可。而将 PCS IP 核与 SerDes IP 核相连，对收发的串行数据进行回环测试时，由于涉及 SerDes IP 核内使用的多项技术，在接口调试时出现了不少问题，最后总结了以下几点：

第一点：恢复时钟和复位的时间点。由于 PCS IP 核在空闲时刻会和 SerDes IP 核进行链路训练，以确保数据对齐，所以不能在 SerDes IP 核恢复时钟前就对 PCS IP 核进行复位，否则会导致两者沟通失败，数据始终无法对齐。

第二点：PCS IP 核与 SerDes IP 核的时钟连接关系。PCS IP 核有两个输入时钟，并不都与 SerDes IP 核输出的时钟相连，而是一个时钟由 SerDes IP 核给出，另一个时钟经参考时钟变化而得。

第三点：PHY 芯片速率配置。PHY 芯片使用方法在手册中有非常详细的介绍，其中和 SGMII 相关的技术点在于 PHY 的速率自适应。需要注意 SGMII 是默认工作在千兆模式下的，并且会和 PHY 芯片沟通两者之间的状态，如果两者有差异，沟通失败会导致数据不互通。所以，互通之前需要配置 PHY 芯片的工作模式。具体配置方法将在第三章详细设计中给出。

2.2.2 MAC 层、 IP 层及 UDP 层协议规范

七层模型[42]中与端系统相关的有物理层、数据链路层、网络层和传输层，其他如应用层无需端系统关心，由于端系统负责数据的封装，故而数据处理时需要符合 MAC 层、IP 层和 UDP 层协议。对于相同的虚拟链路，不允许同时存在多个 UDP 端口号。一般在端口划分时，AFDX 通信端口数值范围从 1024 到 65535 用于队列或采样，

AFDX 服务访问点端口数值范围从 0 到 1023 用于标准通信，范围从 1024 到 65535 用于双向通信。下面分别介绍 MAC 层、IP 层：

（1）MAC 层协议规范

MAC 层规定了数据帧中的目的 MAC 地址、源 MAC 地址及协议类型，端系统采用 IPv4 协议，ARINC 664 part 7 中对目的 MAC 地址和源 MAC 地址有所解释。如图 2.5 所示，目的 MAC 地址的低 16 位是虚拟链路号，高 32 位是固定域。AFDX 系统中，端系统的固定域理应相同，从最高位开始第 7 bit 和第 8 bit 为“1”，这两比特的含义分别是本地管理地址和组地址，其他比特取值没有规定，可由使用者任意规定。

目的MAC地址 48 bit	
固定域 32 bit	虚拟链路标识符 16 bit
xxxx xx11 xxxx xxxx xxxx xxxx xxxx xxxx	

图2.5 目的 MAC 地址格式

如图 2.6 所示，源 MAC 地址由四部分组成：头部和尾部各有 24 bit 和 5 bit 的固定域， 16 bit 的用户定义标识符和 3 bit 的接口标识符。其中固定域从最高位开始第 7 bit 和第 8 bit 的含义分别是局部管理地址和个体地址，通常头部固定域的数值为“0000 0010 0000 0000 0000 0000”。

源MAC地址 48 bit			
固定域 24 bit	用户定义标识符 16 bit	接口标识符 3 bit	固定域 5 bit
0000 0010 0000 0000 0000 0000	nnnn nnnn nnnn nnnn	mmm	0 0000

图2.6 源 MAC 地址格式

用户定义标识符可以用来给每台在网络上 IP 可寻址的终端设备分配一个唯一且有意义的 IP 地址。接口标识符与冗余概念相关，其不同数值表示 MAC 与 AFDX 端系统冗余网络的连接关系，ARINC 664 part 7 中说明“001”表示 MAC 与网络 A 相连，“010”表示 MAC 与网络 B 相连，其余数值均未使用。

（2）IP 层协议规范

源 IP 地址是单播地址，用来指示 AFDX 端系统发送侧的某个分区，目的 IP 地址则是源 AFDX 端系统给某个或多个目的 AFDX 端系统传输 IP 数据包的标识，如图 2.7 所示：

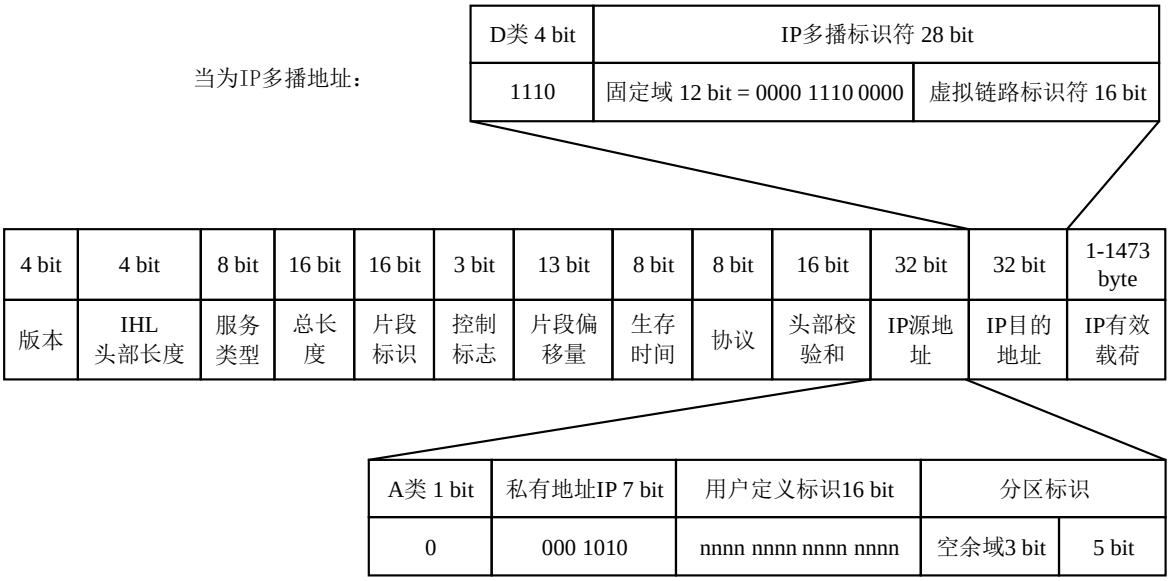

图2.7 目的 IP 地址格式

源 IP 地址前 8 bit 为“0000 1010”，为私有网络单播地址，另外还有 16 bit 的用户定义标识和 8 bit 的分区标识，前者与 MAC 层协议规范相同，也应该用来给每台在网络上 IP 可寻址的终端设备分配一个唯一且有意义的 IP 地址，后者分为两部分，后 5 bit 无特殊含义，前 3 bit 用来标识分区。

目的 IP 地址如果是单播地址，则用以辨识目的设备，否则需和图 2.7 格式兼容，前 16 bit 通常为“1110 0000 1110 0000”。

2.3 本章小结

本章简单介绍了 AFDX 端系统在音频控制单元中的作用和适用的业务场景，给出了 Libero 软件与 Vivado 软件构建 SGMII 接口的区别，并概述了 SGMII 接口中用到的诸如时钟恢复、时钟修正等技术，之后总结出三点应用经验，最后从数据帧格式层面阐述了端系统所要遵循的 MAC 层、IP 层及 UDP 层协议规范。

第三章 面向多业务场景的 AFDX 端系统 IP 核架构设计

本章以科研项目及 ARINC 664 协议中涵盖的技术要求为基础，给出面向多业务场景的 AFDX 端系统 IP 核的整体架构，通过与上一代端系统之间的比较，体现其改进和创新之处，并简略介绍端系统的模块功能和处理流程，最后说明新型端系统中数据帧的格式和所用表项内容的含义。

3.1 AFDX 端系统 IP 核设计需求

结合科研项目技术要求与协议规范，在第一代 AFDX 的基础上，进一步研究 ARINC 664 协议，为了提升端系统的适用性，新型端系统在符合协议的基础上，结合用户需求和真实使用场景下的异常情况，增添了几项协议中并未提及的新功能，以方便用户测试端系统功能。新型 AFDX 端系统设计需求总结基本如下：

（1）支持过滤：针对网络中传输的业务数据，可过滤源 MAC 地址、源端口号及虚拟链路号不符合要求的数据帧。

（2）支持调度及流量控制：完成不同业务流量之间的调度工作，并对不同虚拟链路号的 RC 业务数据帧进行流量控制。

（3）支持序列号（Sequence Number，SN）完整性：针对同一虚拟链路号下的数据，发送侧能够保持发送序列号的次序，接收侧能够根据数据帧的次序正常接收，如若前后数据帧次序混乱，接收端有能力抛弃异常数据帧。

（4）支持不同应用层通信接口：由于上层应用的多样性，端系统需要支持不同类型的端口传输数据。

（5）支持冗余管理：为解决某个交换机发生故障而致使系统瘫痪的问题，端系统发送侧至少含有两个发送端口，能够对数据帧进行冗余备份，同时，接收侧在收到两份相同的数据时，能丢弃多余的数据。

（6）支持板卡硬件加速[43]：端系统板卡可为数据流添加 UDP、IP 头进行加速处理，以缓解上位机的压力，同时还能降低数据在传输过程中的时延。

（7）支持监控及配置：可配置端系统内运行参数以及表项内容，可监控端系统内收发帧数、速率等重要信息。

（8）支持故障注入：在端系统中，能注入部分错误，模拟真实场景可能发生的故障，以观察实际现象。

（9）支持多业务场景：端系统需支持 COM 业务、SAP 业务、BE 业务及细分的七种类型业务数据的操作处理。

3.2 AFDX 端系统 IP 核实现架构

作为本文第一个创新点，在第一代 AFDX 端系统的基础上，根据 ARINC 664 协议和技术需求，设计如图 3.1 所示的新型端系统 IP 核实现架构。新型端系统大致分为发送侧与接收侧，其中发送侧涵盖数据分类模块、COM 数据组帧模块、SAP 数据帧头替换模块、业务调度模块等等。接收侧涵盖业务类型匹配模块、COM 数据拆帧及 SAP 数据帧头还原模块等。

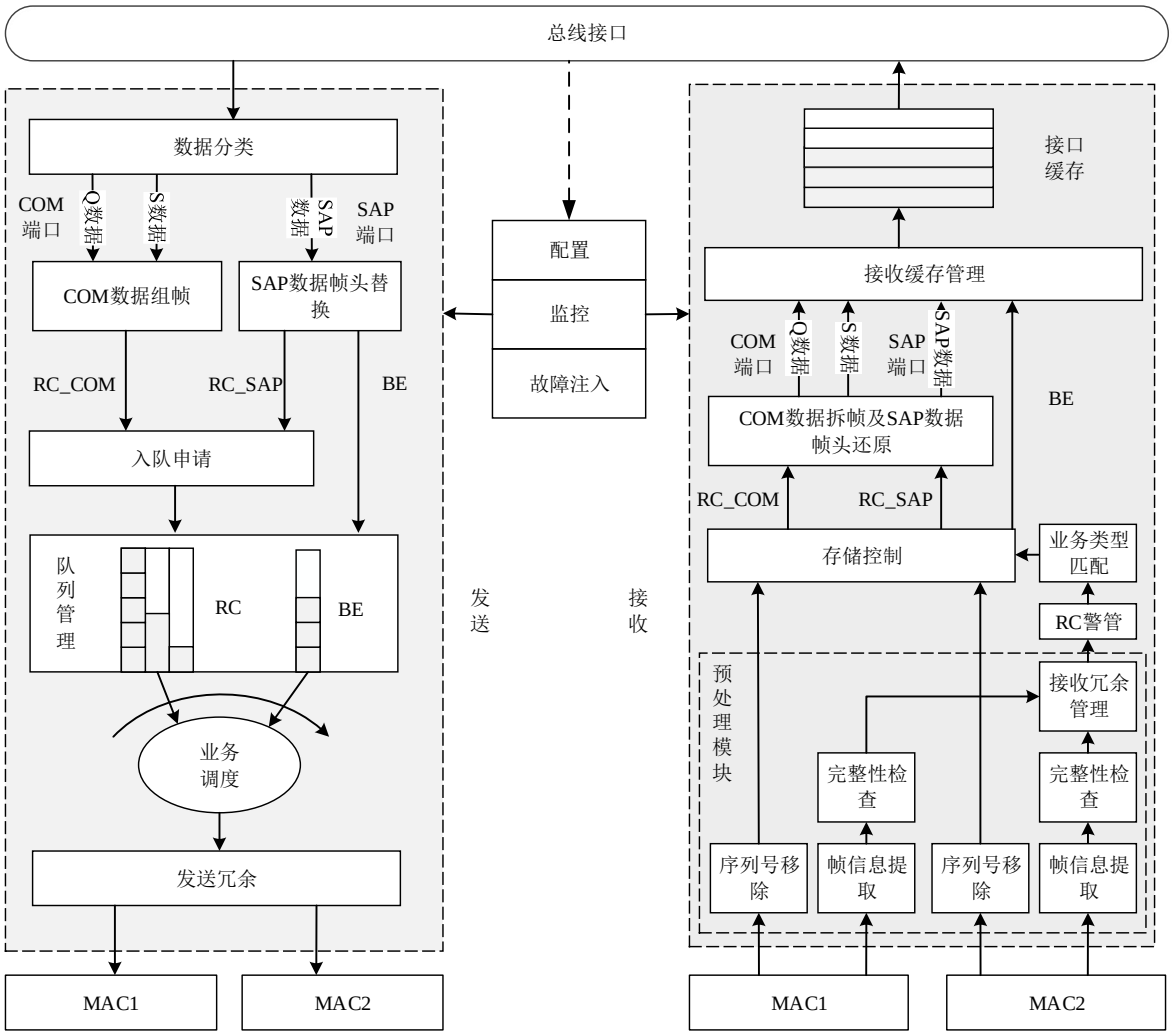

图3.1 AFDX 端系统 IP 核架构

3.2.1 新型端系统改进点

（1）扩增业务类型，端系统可面向多业务场景

新型端系统与第一代端系统的第一个差异在于上层下发的数据，第一代端系统仅支持一种业务的数据处理，即不管是 SAP 数据还是 COM 数据，统一在数据帧前添加帧头，这种方式限制了端系统可向上连接的设备种类，另外也不够符合协议规范。

新型 AFDX 端系统在发送侧设立 COM 数据组帧模块和 SAP 数据帧头替换模块，

相应的在接收侧设立 COM 数据拆帧及 SAP 数据帧头还原模块，不同类型数据有不同的处理流程，新型端系统既可以处理裸数据，也可以处理有帧头的数据。同时接收端新设立业务类型匹配模块，第一代端系统只需要删除帧头，而新版端系统将根据五元组，唯一确认一个数据帧的业务，因为在现有设计下虚拟链路号并不能唯一确认数据帧的业务类型。

相比于第一代端系统，新型端系统可面向多业务场景，应用范围更广，且因为协议一致性，也能支持与其他厂商的类似产品通信。

（2）优化队管和调度设计

第一代 AFDX 端系统采用静态存储的方式存储数据帧，针对多个虚拟链路下的数据帧，要规划一大块缓存区，占用板卡很多存储资源，不利于时序收敛。另外，如果端系统上接的设备发送多种业务，业务量不超过 256 条，且部分虚拟链路下的数据帧流量少，不需要过多存储空间，原队列管理模块设计方式将造成板卡存储资源的浪费。

第一代 AFDX 端系统的调度方式为公平轮询，只需从队列管理模块获取各个队列的状态列表再调度即可，虽然也能完成数据业务的调度工作，但 ARINC 664 协议针对端系统的调度功能有更多要求。

新型 AFDX 端系统设计全新的队列管理和调度，队列管理采用共享缓存和静态缓存结合的方式，区分对待采样数据和其他类型数据，并且可以配置每个队列数据帧的存储区大小，虽然设计复杂性提升，但更节省板卡的存储资源。调度方面则增加了一些新的功能，比如子 VL 调度[44]，更符合协议要求，同时也提升了端系统的性能和适用性。

（3）实现故障注入、监控及配置功能

第一代 AFDX 端系统不具备故障模拟能力，但现实情况是航空航天领域通信设备有时候会受到环境的影响，从而可能会出现多种故障，因此在新型端系统设计时添加故障模拟功能。

故障注入模块耦合端系统内部多个功能模块，通过上层应用指令输入，驱动硬件板卡产生故障激励，影响板卡正常工作逻辑，继而发生故障，真实地模拟 AFDX 端系统的应用场景，并且直观地观察故障导致的现象，便于研究人员在地面仿真验证以衡量出错成本。

新型端系统还增加监控及配置功能，硬件板卡内部逻辑定时寄存端系统的工作参数，再通过寄存器周期性上传给终端，用户可通过图形界面直接观测当前端系统的运行状态，以确认端系统是否处于正常工作状态。另外，用户还可以通过上层应用配置端系统部分模块所需的表项内容、运行参数等，例如网口速率，用户可以将端系统网口速率配置为千兆、百兆或者十兆工作模式，使端系统能够兼容多种网口速率的交换

机。

3.2.2 端系统模块功能简介

发送侧模块功能简介：

（1）数据分类模块：对上层接口下发的数据流分类，之后递交给 COM 数据组帧模块和 SAP 数据帧头替换模块进行处理。

（2）COM 数据组帧模块：COM 数据帧只有裸数据，此模块负责裸数据的帧头添加，将其包装成完整数据帧。使端系统可面向多业务场景的关键模块。

（3）SAP 数据帧头替换模块：SAP 数据帧本身带有帧头，但为了设备标识的一致性，此模块负责原有帧头的替换。使端系统可面向多业务场景的关键模块。

（4）入队申请模块：与队列管理模块进行交互，以确定来自于 COM 数据组帧模块和 SAP 数据帧头替换模块的数据帧是否可以存入缓存区。

（5）队列管理模块：缓存 COM 数据、SAP 数据和 BE 数据，等待业务调度模块的指令。

（6）业务调度模块：根据 RC 数据帧 BAG 的要求确定 RC 是否能被调度，如果可以则向队列管理模块发送调度指令，RC 数据帧发送间隙调度 BE 数据帧。

（7）发送冗余模块：给数据帧添加序列号，将需要发送的数据冗余备份，从至少两个端口发送，以避免一个交换机瘫痪的故障。

接收侧模块功能简介：

（1）预处理模块：包含序列号移除、帧信息提取、完整性检查和接收冗余管理。一路数据经过帧尾序列号移除后递交给存储控制模块，另一路数据被用于数据信息的提取以供后续模块的进一步操作，完整性检查检测数据帧是否连续，如为异常帧则向存储模块传递丢弃信号，接收冗余管理对来自两路 MAC 的数据帧进行比对，如发现数据帧是冗余帧，向存储模块传递丢弃信号。

（2）RC 警管模块：检查相同虚拟链路号下的 RC 数据帧是否满足 BAG 要求，如发现不满足则向存储模块传递丢弃信号。

（3）业务类型匹配模块：根据帧信息提取模块提取的信息，查询 RC 数据帧的类型，并把信息传递给存储控制模块。使端系统可面向多业务场景的关键模块。

（4）存储控制模块：存储已去除序列号的数据帧，并根据丢弃信号丢弃相应数据帧，如业务类型模块传递帧信息至此，则将数据传递给后续模块进行处理。

（5）COM 数据拆帧及 SAP 数据帧头还原模块：对存储控制模块传来的 COM 数据，需要去除帧头，将裸数据提交给下一模块；对存储控制模块传来的 SAP 数据，需要还原帧头并提交给下一模块。使端系统可面向多业务场景的关键模块。

除去以上模块，新型 AFDX 端系统架构中还包括配置模块、监控模块、故障注入

模块和 MAC。配置模块负责配置端系统运行参数、表项等，监控模块负责监控端系统的运行状态，比如收发帧数、速率等，故障注入模块负责产生故障激励，模拟真实应用场景下因环境等因素造成的差错，以便于观察故障现象，MAC 则与网口十、百、千兆速率的切换紧密相关。

3.2.3 端系统数据处理流程

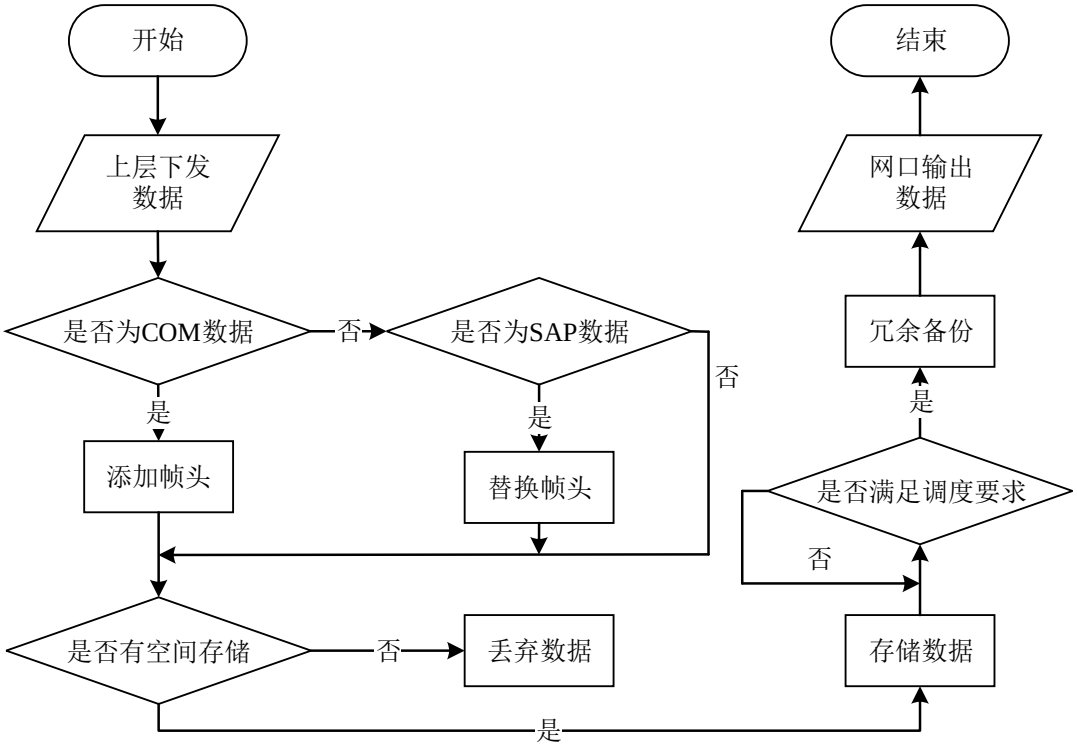

图3.2 发送侧数据处理流程

发送侧数据处理流程如图 3.2 所示：

（1）总线下发数据，数据分类模块根据数据帧头内容判别数据属于 COM 帧、SAP 帧还是普通 BE 帧。

（2）如果是 COM 数据帧，则在 COM 数据组帧模块为裸数据添加帧头；如果是 SAP 数据帧，则在 SAP 数据帧头替换模块替换 SAP 数据帧原有帧头；如果是 BE 帧，则不进行任何操作，将数据传递给入队申请模块。

（3）入队申请模块与队列管理模块进行信息交互，以确认队列中是否还有剩余空间存储完整数据帧，如果有则数据入队，否则丢弃数据。

（4）业务调度模块根据规则调度数据帧，检查 RC 数据帧是否符合 BAG，RC 数据帧之间是否能发送 BE 数据帧，如果满足要求，传递调度指令给队列管理模块，数据出队。

（5）在发送冗余模块对数据进行冗余备份，经 MAC 传输至网口。

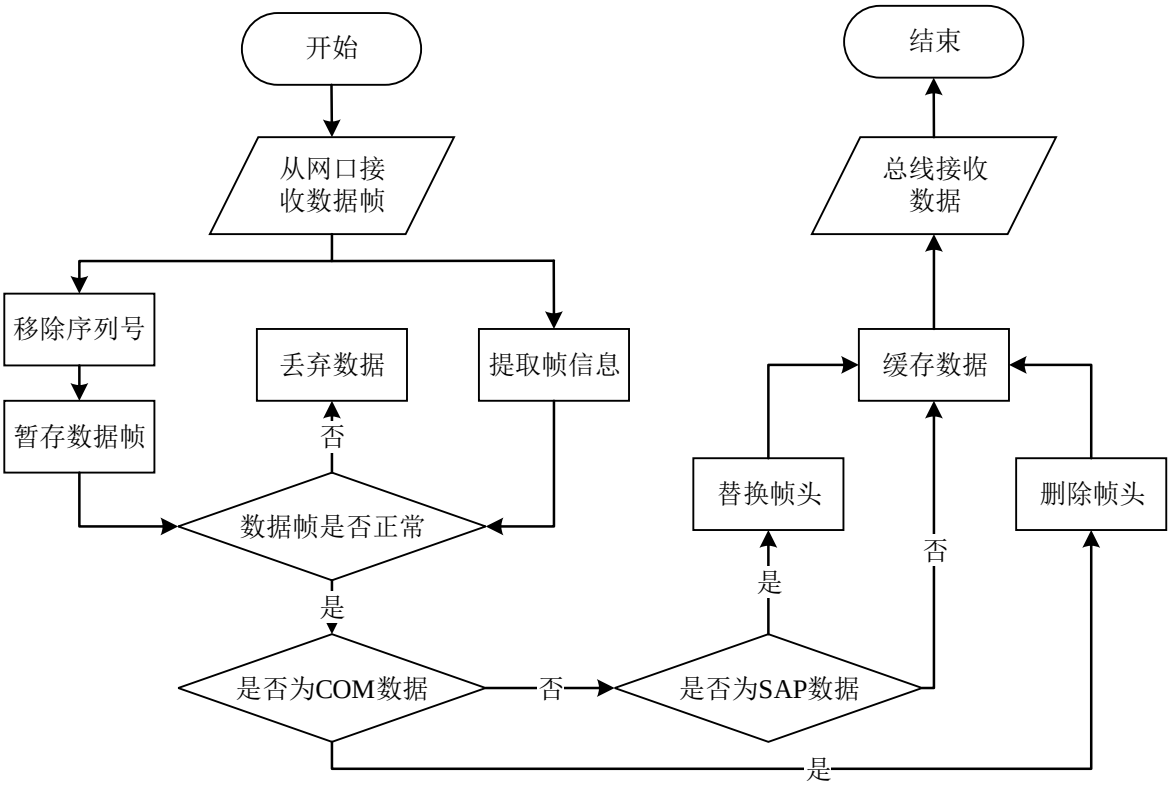

图3.3 接收侧数据处理流程

接收侧数据处理流程如图 3.3 所示：

（1）接收侧从网口接收数据，一份数据进入序列号移除模块，去除序列号的数据之后被递交给存储控制模块，另一份数据进入帧信息提取模块。

（2）帧信息被提取后，会依次进入完整性检查模块、接收冗余管理模块、RC 警管模块和业务类型匹配模块，判断数据帧是否正常，不正常的会被丢弃；

（3）正常的数据帧会根据数据类型决定后续操作。如果是 COM 数据帧，则拆除数据帧头变成裸数据；如果是 SAP 数据，则还原 SAP 数据帧原有帧头；如果是 BE 数据帧，则不进行任何操作，将数据传递给接收缓存管理模块；

（4）数据在缓存中存储，待总线读取，流程结束。

3.3 端系统帧格式定义

端系统帧格式定义如图 3.4 所示，上层应用下发的数据除载荷外，还含有关键帧头信息，关键帧头信息会随着数据流分级地向后续模块传输，以便端系统对数据进行正确处理和操作，且这些信息只在端系统内部使用，进入 MAC 的数据帧已不再携带这些信息。

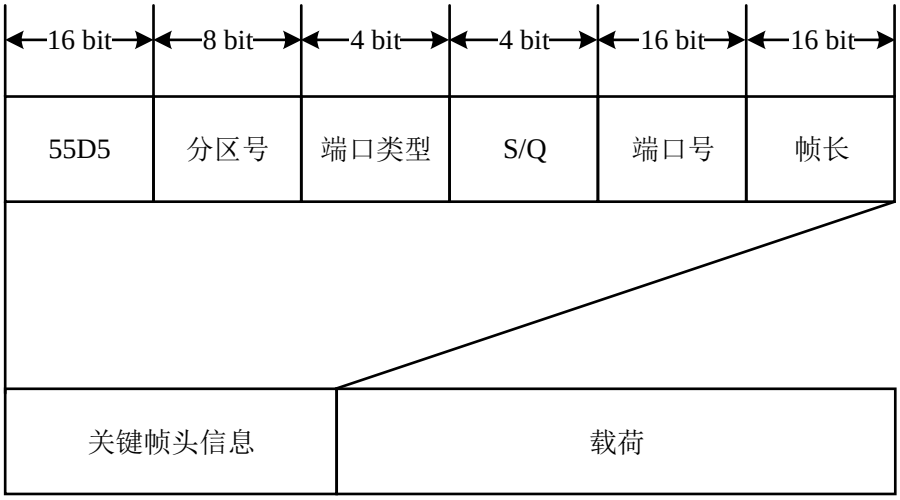

图3.4 端系统帧格式

关键帧头信息各字段含义如下：

（1）55D5：帧头定界符，端系统凭借帧头定界符确定一个完整数据帧的起始点。

（2）分区号：分区[45]的设置与隔离不同应用息息相关，端系统内部会根据分区号将数据存入不同的缓存区，可以有效隔离不同应用数据，避免因为某一个或几个应用出现故障后系统陷入瘫痪。

（3）端口类型：端系统通过识别该字段辨别数据帧属于 COM 端口数据还是 SAP 端口数据。

（4）S/Q：COM 端口数据类型，端系统通过识别该字段辨别数据帧属于 S 数据还是 Q 数据。

（5）端口号：端口号与虚拟链路号存在映射关系，通过端口号也能识别数据帧是 RC 数据帧还是 BE 数据帧，且不同应用下发的数据端口号不重叠，相当于数据帧的身份标识。

（6）数据帧帧长：指示载荷的字节长度，用于后续模块的处理。

3.4 端系统帧表项内容定义

AFDX 系统是一种定制网络，各类数据业务由用户根据实际情况事先规划，如需更改业务规划，可以通过配置表项的方式实时更新，其中端系统包括以下表格：硬件协处理器加速表、S 数据起始地址存储表、业务调度表、BAG 表、子 VL 映射表、RC 索引表、RC 警管表、业务类型匹配表、VL 索引表、VL 业务查找表。在运行端系统，应用下发数据帧之前，用户必须根据实际需求生成表项文件并写入端系统内部，下面将针对各表项内容含义作出说明。

3.4.1 硬件协处理器加速表

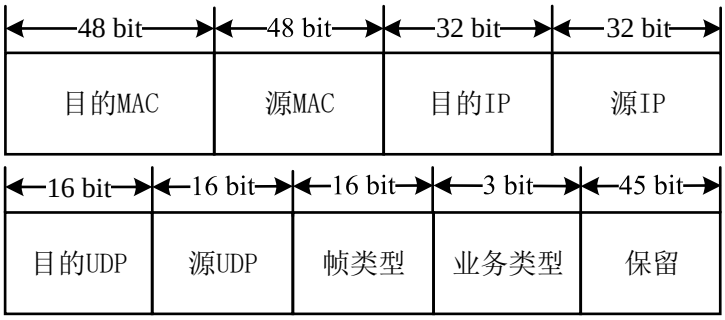

图3.5 硬件协处理器加速表

硬件协处理器加速表一共 256 bit，各字段含义及长度如图 3.5 所示，此表用于 COM 数据组帧模块及 SAP 数据帧头替换模块，以端口号为地址查询表项，其中业务类型涵盖七种，标号从“001”到“111”，分别称为 MAC_COM、IP_COM、UDP_COM、MAC_SAP、IP_SAP、UDP_SAP 和 MAC_RAW。

MAC_COM：应用下发数据只有裸数据，此类型数据帧在组帧时需要在帧头添加 MAC 字段。

IP_COM：应用下发数据只有裸数据，此类型数据帧在组帧时需要在帧头添加 MAC 字段和 IP 字段。

UDP_COM：应用下发数据只有裸数据，此类型数据帧在组帧时需要在帧头添加 MAC 字段、IP 字段和 UDP 字段。

MAC_SAP：应用下发数据已携带 MAC 头，此类型数据帧需要在模块内替换源 MAC 字段。

IP_SAP：应用下发数据已携带 IP 头，此类型数据帧需要在模块内替换源 IP 字段。

UDP_SAP：应用下发数据已携带 UDP 头，此类型数据帧需要在模块内替换源 UDP 字段。

MAC_RAW：应用下发数据帧头完整，完美匹配端系统，不需要进行任何操作。

针对帧类型字段，可以把数据帧统一设置成“0800”，结合源 MAC 字段，区分 RC 数据帧和 BE 数据帧。另外，也可以将 RC 数据帧的帧类型定义为例如“0888”，BE 定义为“0800”，这取决于用户实际需求。需要说明的是，端口号与帧类型一一对应，不存在某个端口号既是 RC 数据帧又是 BE 数据帧的情况。

3.4.2 S 数据起始地址存储表

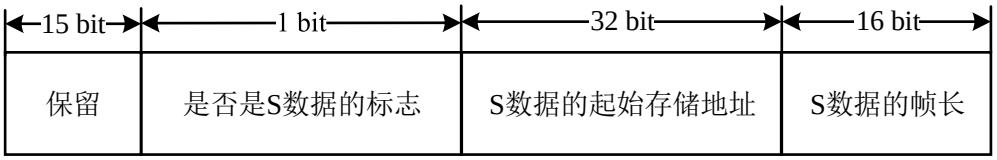

图3.6 S 数据起始地址存储表

S 数据起始地址存储表一共 64 bit，各字段含义及长度如图 3.6 所示，此表用于队列管理模块，以端口号为地址查询表项。如果 S 数据的标志为“1”，则将数据搬移至存储区，不同端口号的存储地址不同，且存储区只容许一个完整帧的存在，故而只需表项给出 S 数据的起始存储地址即可，S 数据的帧长规定了存储数据的最大帧长，如若数据帧不符合要求则会被丢弃。

3.4.3 BAG 表

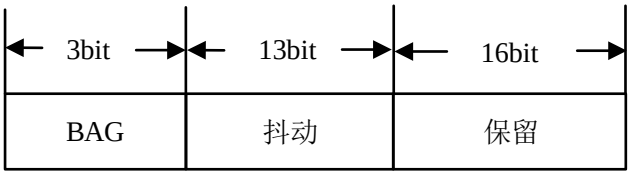

图3.7 BAG 表

BAG 表一共 32 bit，各字段含义及长度如图 3.7 所示，此表用于业务调度模块，以虚拟链路号为地址查询表项。BAG 即带宽分配间隙，端系统的 BAG 范围从“000”到“111”，表示 1ms 到 128ms，且数值只能是 2 的幂次方。BAG 相当于令牌桶。

抖动值是在 BAG 的基础上扩增了令牌桶的容量，范围为 40us~500us。调度器维系着一个计数器，计数器的最大值为 BAG 与抖动值的和，所以在启用抖动值时，数据并不总是以 BAG 的间隙传输的，两个 RC 数据帧帧间间隔最小为 BAG 与抖动值的差。比如计数器达到最大值，此时有一个数据帧被调度，计数器的值减去 BAG 变为抖动值，又过了一定时间，计数器达到 BAG，此时又有一个数据帧被成功调度，这两个帧的间隔即为最小时间间隔。

3.4.4 业务调度表

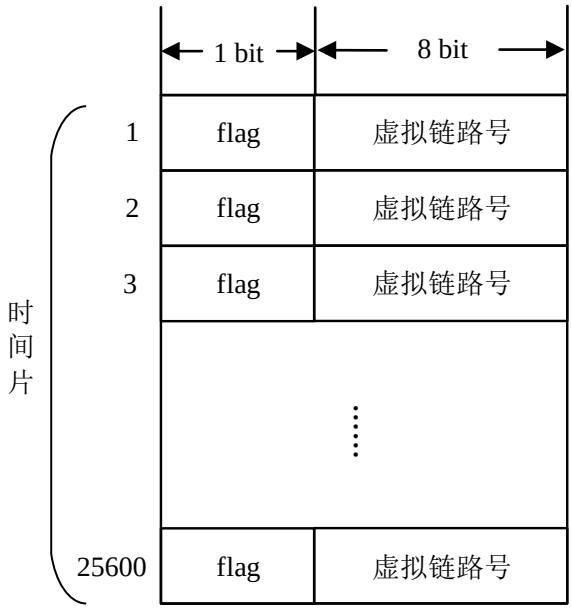

图3.8 业务调度表

业务调度表一共 9 bit，各字段含义及长度如图 3.8 所示，此表用于业务调度模块。为方便理解，已列出全表表项，查表时地址逐一递增，总的地址数等于调度片的数目，每一个调度片[46]都可以规划一个业务，可以把这种调度方式理解为基于时隙的调度。

“flag”表示是否有 RC 数据帧规划在此调度片，如为“1”，则虚拟链路号（VL_ID）有效；如为“0”，则表示此时间片没有 RC 数据帧规划，业务调度模块可以调度 BE 数据帧。

3.4.5 子 VL 映射表

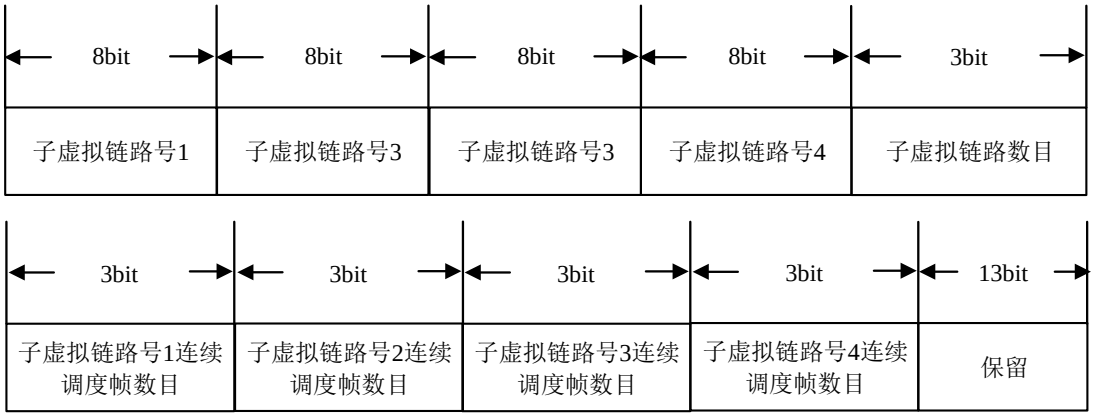

图3.9 子 VL 映射表

子 VL 映射表一共 60 bit，各字段含义及长度如图 3.9 所示，此表用于业务调度模块。ARINC 664 part 7 中就子虚拟链路有具体解释，子虚拟链路的存在并不是必要

的，此机制的应用是为了提高链路的带宽利用率，已知 RC 数据帧有 BAG 的概念，最高为128ms。当一个数据帧的发送间隙为256ms时，由于只能将间隙配置成128ms，会降低带宽利用率，当将一个虚拟链路号分配给两个端口号的数据帧使用时，显然能提高带宽的利用率，这两个端口号也被称为子虚拟链路号。

协议表明，同一虚拟链路号下最多有四个子虚拟链路号，且子虚拟链路之间采用公平轮询的方式进行调度。另外表项还添加了连续调度的帧数目，以应对突发数据流。

3.4.6 RC 索引表

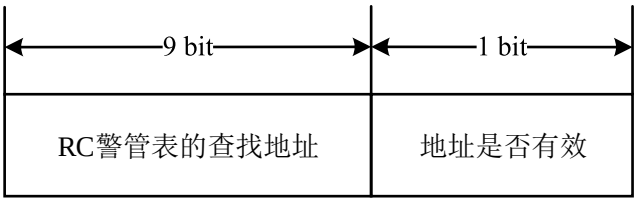

图3.10 RC 索引表

RC 索引表一共 9 bit，各字段含义及长度如图 3.10 所示，此表用于 RC 警管模块。引入 RC 索引表，将大数值的虚拟链路号映射为小数值的虚拟链路号，节省端系统板卡的存储资源，以虚拟链路号为地址查询表项，若发现地址有效，则以表项中的地址作为 RC 警管表的查找地址。

3.4.7 RC 警管表

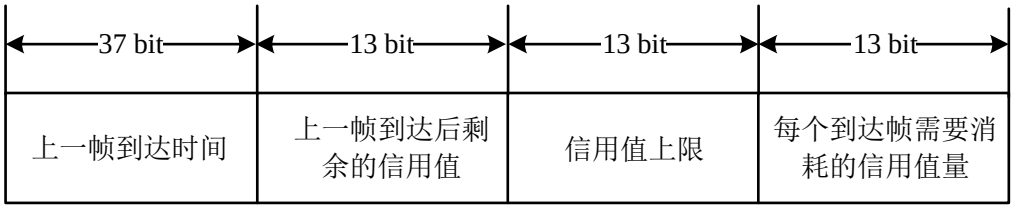

图3.11 RC 警管表

RC 警管表一共 76 bit，各字段含义及长度如图 3.11 所示，此表用于 RC 警管模块。用“Time”表示上一帧到达时间，“ACrew”表示上一帧到达后剩余的信用值，“ACmax”表示信用值上限，“Smax”表示每个到达帧需要消耗的信用值量，“Jitter”表示抖动值。

ARINC 664 part 7 针对 RC 警管有所说明，抖动值的范围从 40us 到 500us，BAG 的取值从 1ms 到 128ms，且是 2 的幂次方。ACmax = Smax(1 + Jitter/BAG)，由此公式可知如何配置各虚拟链路号 RC 数据帧的 ACmax，并且由公式可知令牌的增速为 Smax/BAG，可以设每 20us 令牌数加一以简化设计实现，此时， Smax=BAG/20 ，ACmax = BAG (1+Jitter/BAG) /20。

ACmax 的取值范围从 52us 到 6425us，Smax 的取值范围从 50us 到 6400 us，同一情况下 Accountmax > Smax，两者皆只需 13 位二进制数表示。AFDX 端系统用户侧工作频率为 125MHz，需用计数器进行 2500 分频产生周期为 20us 的时钟，用于计数器的累加。

3.4.8 业务类型匹配表

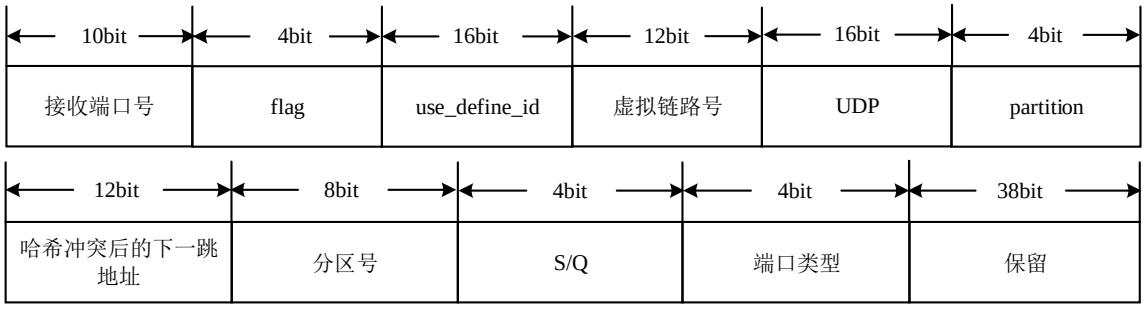

图3.12 业务类型匹配表

业务类型匹配表一共 128 bit，各字段含义及长度如图 3.12 所示，此表用于业务类型匹配模块，且只用于 COM 数据帧。其中 Flag 暂时未使用，use_define_id 和 partition 为 IP 字段提取内容。不同数据帧在发送侧会按照预先规划添加帧头，数据帧由 IP 字段、VL 号、UDP 字段唯一确定，在进行业务匹配时，将数据帧的这些信息提取后和表项内容比对，比对正确的才是 AFDX 端系统要收取的数据帧。

3.4.9 VL 索引表

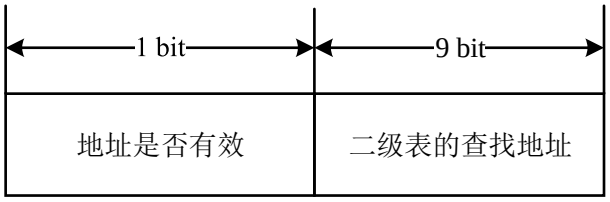

图3.13 VL 索引表

VL 索引表一共 10 bit，各字段含义及长度如图 3.13 所示，此表用于业务类型匹配模块。由于一个端系统最多能收到 512 条业务，但实际上网络中存在的虚拟链路远不止 512 条，为节省端系统板卡的存储资源，引入 VL 索引表，将数值大的虚拟链路号映射成数值小的虚拟链路号，以虚拟链路号为地址查询表项，若发现地址有效，即此 VL 被规划为 SAP 数据，则接收的数据帧的虚拟链路号等于二级表的查找地址，而二级表又被称之为 VL 业务查找表。

3.4.10 VL 业务查找表

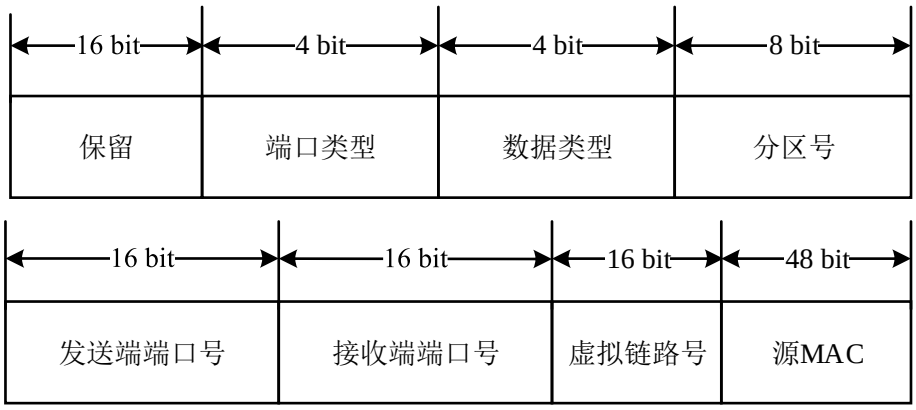

图3.14 VL 业务查找表

VL 业务查找表一共 128 bit，各字段含义及长度如图 3.14 所示，此表用于业务类型匹配模块。此表格与业务类型匹配表相似，故不做多余说明。

3.5 本章小结

本章注重整体介绍，便于读者对后续行文的理解。本章首先结合设计需求阐述了 AFDX 端系统 IP 核的总体架构，通过对比新旧两代端系统架构，总结了三方面的改进点，并且基于架构介绍了端系统模块功能以及数据处理流程，之后给出了端系统内部帧格式与表项内容的定义，详细讲解了各比特位的含义。

第四章　面向多业务场景的 AFDX 端系统 IP 核的设计与实现

第三章已给出面向多业务场景的 AFDX 端系统 IP 核总体设计方案的具体说明，本章将分为五个部分详细介绍各模块的功能并给出具体设计方法。面向多业务场景的 AFDX 端系统 IP 核的发送侧和接收侧这两个部分包含多个关键功能，除此之外，由于故障注入模块、配置及监控模块与接口模块跟发送侧和接收侧都有关联，故而把它们作为专门的章节。内容中与第一代 AFDX 端系统相差不大的模块仅做简单介绍。

4.1 发送侧模块划分

4.1.1　数据分类模块

（1）模块功能概述

接收上层应用下发的数据，提取帧头信息，根据端口类型字段对数据进行分类，将合法的数据帧发送至 COM 数据组帧模块和 SAP 数据帧头替换模块。

（2）模块接口框图

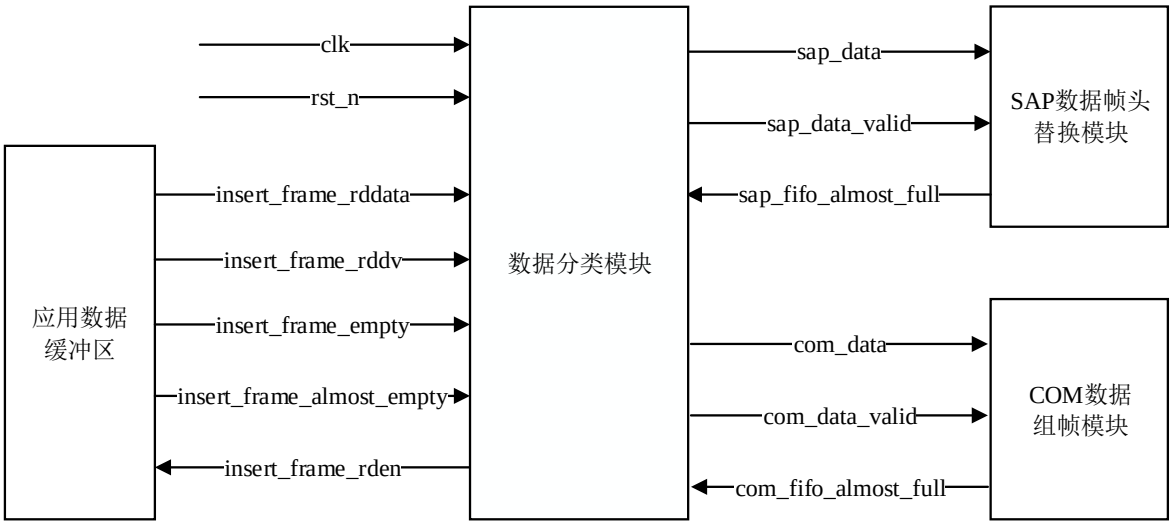

图4.1 数据分类模块接口

（3）模块详细设计

如果上层应用已下发数据帧，下连模块可以接收数据帧，即应用数据缓冲区非空，且 SAP 数据帧头替换模块或 COM 数据组帧模块非将满，则开始读取缓存区的数据，定位数据帧帧头后提取并寄存前 8 字节帧头信息，先判别数据帧帧长是否符合要求，如果符合要求则利用端口类型字段判别数据帧属于 COM 数据还是 SAP 数据，而后将两类数据分别送至 COM 数据组帧模块和 SAP 数据帧头替换模块。

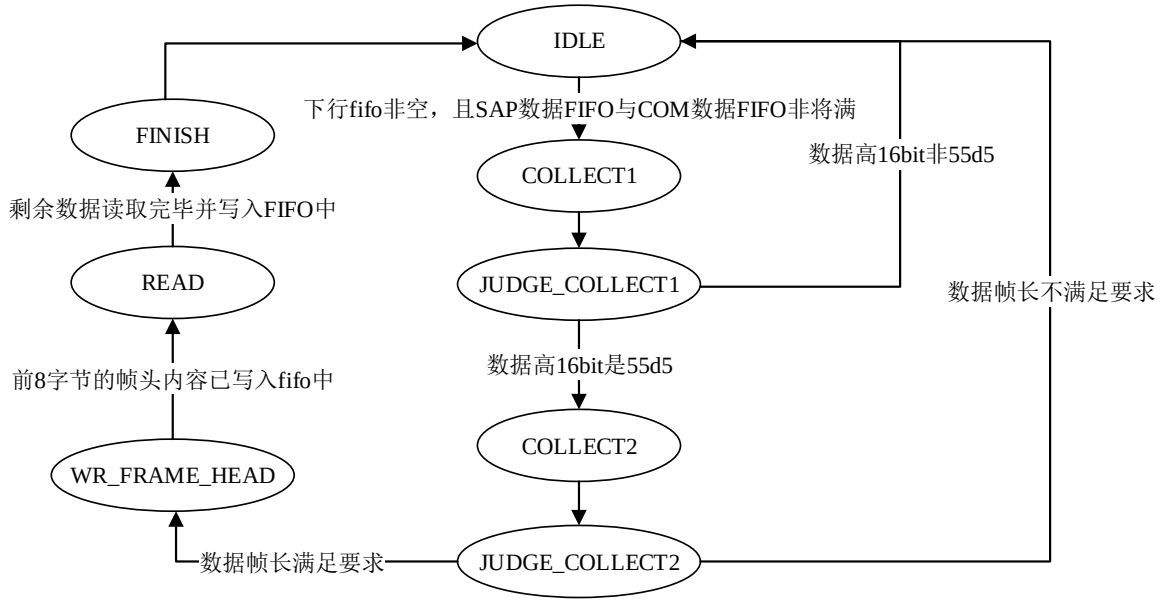

图4.2 数据分类状态机

IDLE：初态。检测到与上层应用相连的接口缓存区有数据且存储 COM 数据和 SAP 数据的 FIFO 未将满时，可跳入下一状态。

COLLECT1：提取数据帧帧头前 4 字节数据。

JUDGE_COLLECT1：判断提取的前 4 字节数据是否为“0x55d5”，如果相等则表明成功定位帧头，进入下一状态，否则返回初态。

COLLECT2：提取数据帧帧头第 5 至第 8 字节数据。

JUDGE_COLLECT2：寄存提取的帧头信息，判断数据帧帧长是否合法，如果合法则进入下一状态，否则返回初态。

WR_FRAME_HEAD：存储提取的前 8 字节帧头信息。

READ：从接口缓存区读取数据，直至读取的数据长度与帧长相等，同时将读取的数据存储。

FINISH：存储完一个完整数据帧后，回到初态，此状态下，可以根据端口类型字段确认数据帧属于 COM 数据还是 SAP 数据并送入后续模块进行处理。

4.1.2 COM 数据组帧模块

（1）模块功能概述

在 FPGA 板卡内部组帧可减少操作系统工作负担，提高数据传输速率，COM 数据组帧模块主要通过端口号识别数据帧属于 MAC_COM、IP_COM 还是 UDP_COM，接着再结合表项内容按照需求为数据帧增加帧头，以组成完整的数据帧，之后把数据递交给入队申请模块进行下一步处理。本模块是使 AFDX 端系统可面向多业务场景的关键模块之一。

（2）模块接口框图

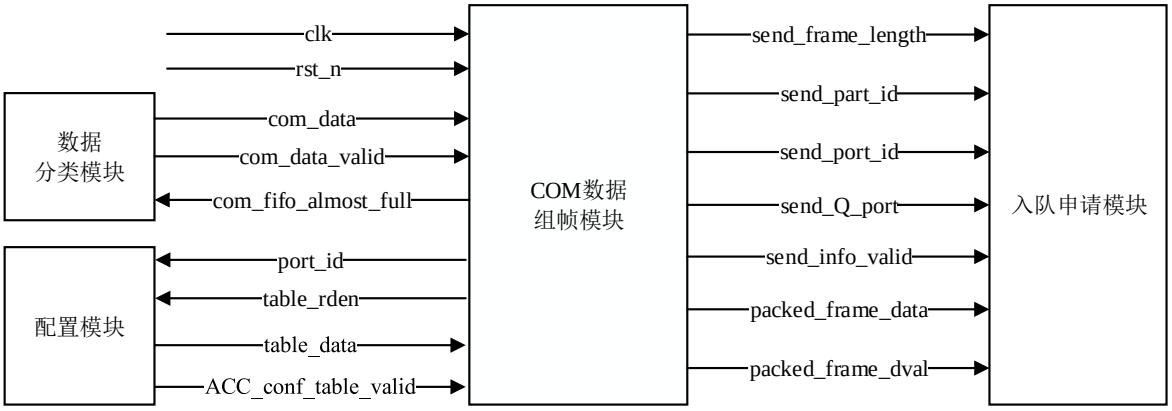

图4.3 COM 数据组帧模块接口

（3）模块详细设计

从数据分类模块接收 COM 数据后，先存储完整数据帧，并将帧头信息中的端口号作为查表地址，查询硬件协处理器加速表，获知数据帧所属的类型。

如果所属类型为 UDP_COM，则从表项中获取 MAC 地址、IP 地址、UDP 端口号等信息，用于生成数据的 IP 头和 UDP 头，待检验和计算完成之后，开始组帧直至完成，并协同帧信息一同递交给入队申请模块进行后续处理。

如果数据帧类属 MAC_COM，则要从表中获取 MAC 地址用于组帧，无需计算 IP 校验以及 UDP 校验和，随后把数据递交给入队申请模块。如果数据帧类属 IP_COM，则要从表中获取 MAC 地址和 IP 地址用于组帧，无需计算 UDP 校验和。

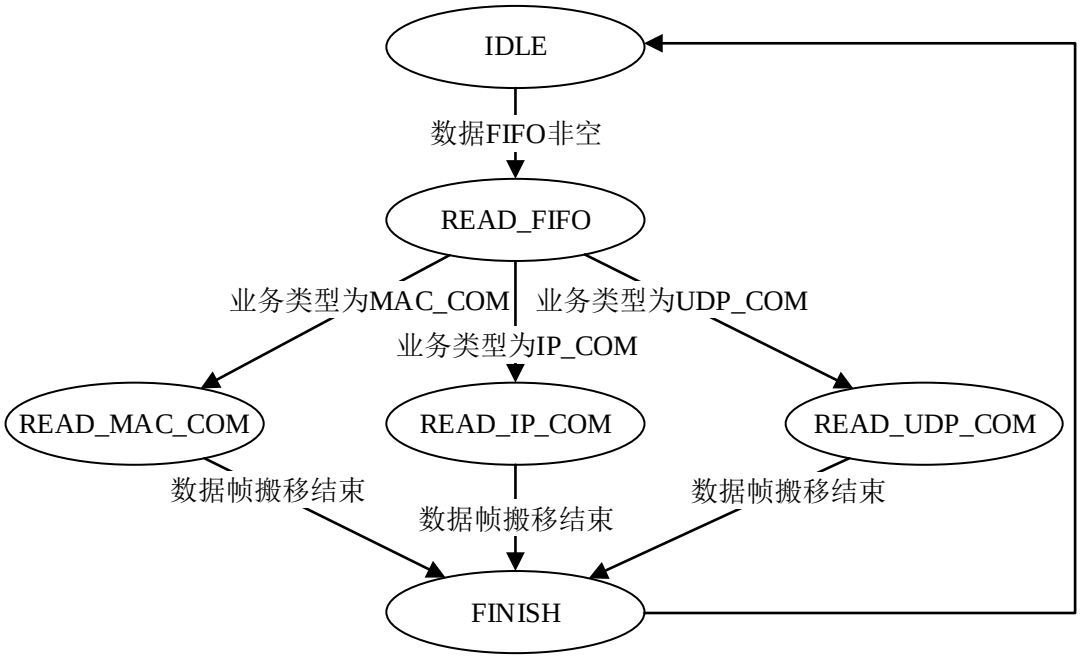

图4.4 COM 数据组帧状态机

IDLE：初态，当数据 FIFO 非空时，可跳入下一状态。

READ_FIFO：提取数据 FIFO 中的帧头信息，通过端口号查询硬件协处理器加速表，寄存表项内容中的信息，并依据表项内容获知的 COM 数据的业务类型决定跳入 READ_MAC_COM、READ_IP_COM 还是 READ_UDP_COM 状态。

READ_MAC_COM：通过从硬件协处理器加速表中获取的帧类型和 MAC 地址，完成 MAC 头的组装，同时从数据 FIFO 中读取剩余数据载荷，可跳入下一状态。

READ_IP_COM：通过从硬件协处理器加速表中获取的 IP 地址，协同从硬件协处理器加速表中获取的帧类型和 MAC 地址，完成 MAC 头和 IP 头的组装，同时从数据 FIFO 中读取剩余数据载荷，可跳入下一状态。

READ_UDP_COM：通过从硬件协处理器加速表中获取的 IP 地址和 UDP 端口号，协同从硬件协处理器加速表中获取的帧类型和 MAC 地址，完成 MAC 头、IP 头与 UDP 头的组装，同时从数据 FIFO 中读取剩余数据载荷，可跳入下一状态。

FINISH：结束数据帧的搬移工作，回到初态。

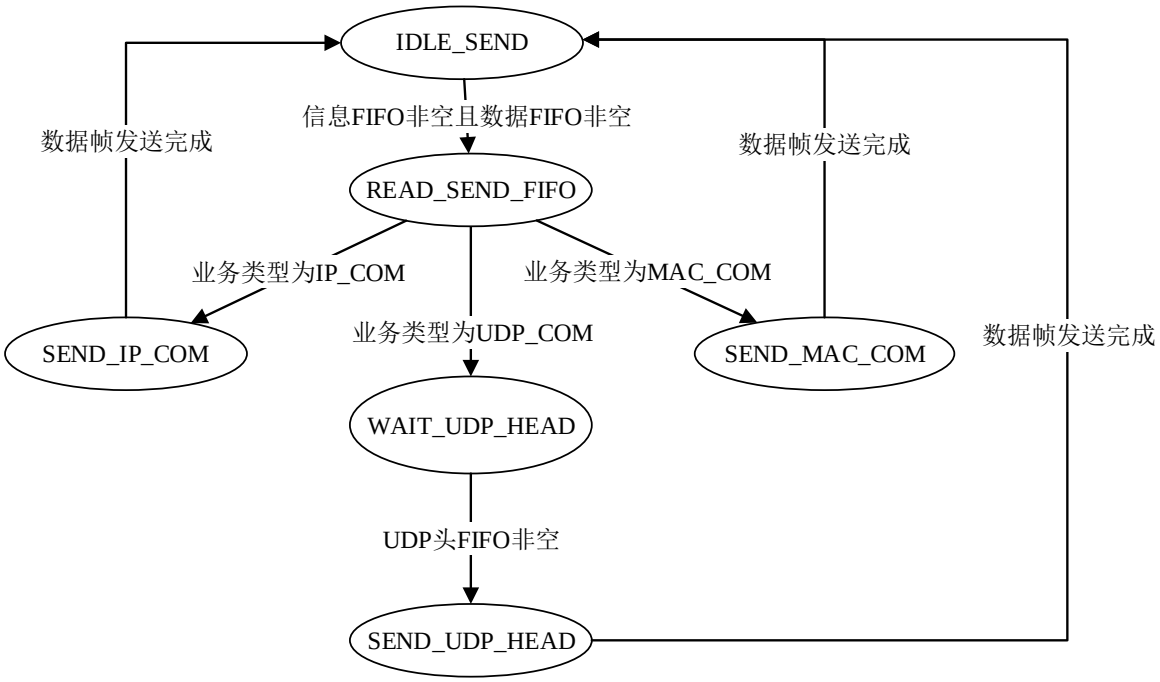

图4.5 COM 数据发帧状态机

IDLE_SEND：初态，COM 数据组帧状态机结束，信息 FIFO 和数据 FIFO 非空，此时可跳入下一状态。

READ_SEND_INFO：拉高信息 FIFO 读使能并将帧信息寄存，通过获知的 COM 数据的业务类型决定状态机下一步该跳入 SEND_MAC_COM 状态、SEND_IP_COM 状态还是 SEND_UDP_COM 状态。

SEND_MAC_COM：发送已组装完成的 MAC 头，并依据帧长从数据 FIFO 中读

取剩余数据载荷，回到初态。

SEND_IP_COM：发送已组装完成的 MAC 头和 IP 头，并依据帧长从数据 FIFO 中读取剩余数据载荷，回到初态。

WAIT_UDP_HEAD：如果 UDP 头未生成，则在此状态等候，直至 UDP 头生成，可跳入下一状态。

SEND_UDP_COM：发送已组装完成的 MAC 头、IP 头和 UDP 头，并依据帧长从数据 FIFO 中读取剩余数据载荷，回到初态。

4.1.3 SAP 数据帧头替换模块

（1）模块功能概述

与 COM 数据组帧模块类似，SAP 数据帧头替换模块的意义也是减少操作系统工作负担，提高数据传输速率，通过端口号识别数据帧属于 MAC_SAP、IP_SAP 还是 UDP_SAP，根据表项内容为数据帧替换相应的帧头，再把数据递交给入队申请模块进行下一步处理。

如果数据类型为 MAC_RAW，比如 TFTP 数据帧，则不需要任何处理，模块内部直接与队列管理模块交互，确认是否有足够缓存供数据存储。本模块是使 AFDX 端系统可面向多业务场景的关键模块之一。

（2）模块接口框图

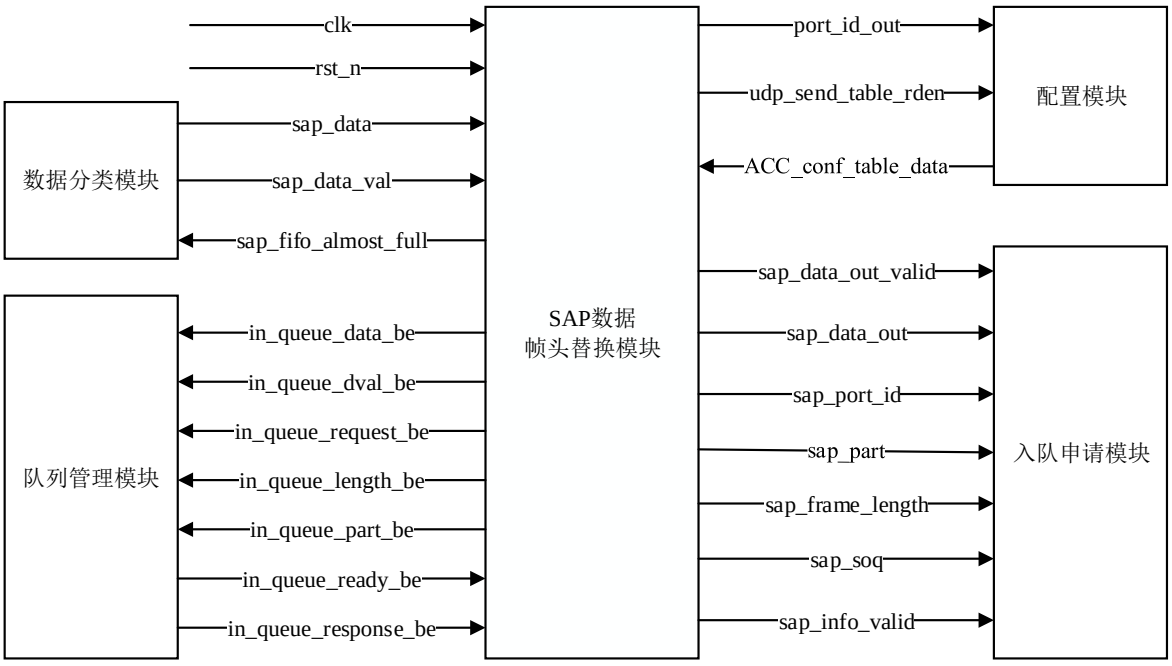

图4.6 SAP 数据帧头替换模块接口

（3）模块详细设计

从数据分类模块接收非 COM 数据后，先存储完整数据帧，并将帧头信息中的端口号作为查表地址，查询硬件协处理器加速表，获知数据帧所属的类型。

如果所属类型为 MAC_SAP，则从表项中获取新的源 MAC，替换原有 SAP 数据的源 MAC，替换完成后协同帧信息一同递交给入队申请模块进行后续处理。

如果数据帧类属 IP_SAP，则用表项中的源 IP，替换原有 SAP 数据的源 IP，替换完成后递交给入队申请模块。

如果数据帧类属 UDP_SAP，则用表项中的源 UDP，替换原有 SAP 数据的源 UDP，完成替换工作后递交给下一模块。

另外当数据属于 MAC_RAW（一般是 BE 数据帧）类型时，SAP 数据帧头替换模块不需要对数据进行任何操作，而是向队列管理模块发送入队请求。

如果队列中有存储空间，SAP 数据帧头替换模块会得到允许入队的反馈，此时数据帧和帧信息会一同递交给队列管理模块以进行后续操作。相反，队列中没有剩余的存储空间供 MAC_RAW 数据存储时，表明上层应用流量异常，板卡已无法处理，这时数据帧将被丢弃。

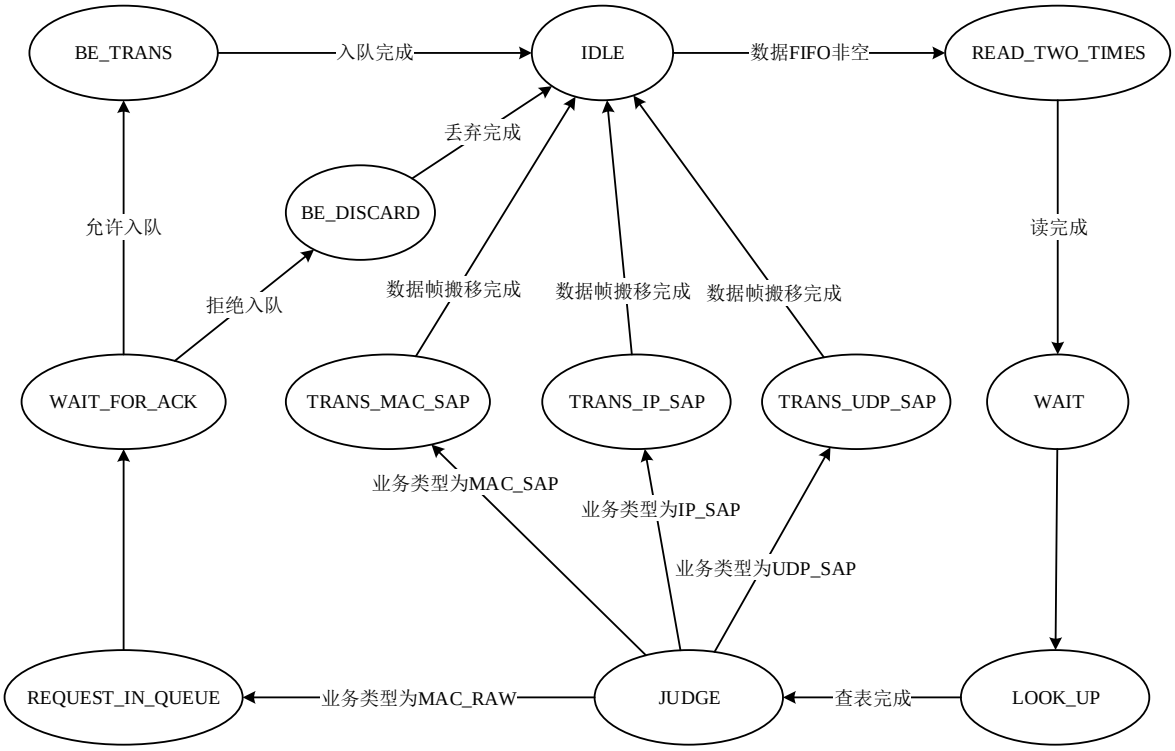

图4.7 SAP 数据帧头替换状态机

IDLE：初态，如果 FIFO 中有 SAP 数据，可跳入下一状态。

READ_TWO_TIMES：等待 SAP 数据的帧头信息被提取完成之后，可跳入下一状态。

WAIT：将提取的数据帧的帧头信息中的帧长信息寄存完成之后，可跳入下一状态。

LOOK_UP：将数据帧的端口号作为查表地址查询硬件协处理器加速表，待获取相应表项内容后可跳入下一状态。

JUDGE：依据表项内容获知的 SAP 数据的业务类型决定跳入 TRANS_MAC_SAP 状态、TRANS_IP_SAP 状态、TRANS_UDP_SAP 状态还是 REQUEST_IN_QUEUE 状态。

TRANS_MAC_SAP：当硬件协处理器加速表显示当前数据帧类型是 MAC_SAP，用表项内容中的源 MAC 地址替换掉目标 SAP 数据的旧 MAC 地址，待数据帧结束搬移后，可跳入下一状态。

TRANS_IP_SAP：当硬件协处理器加速表显示当前数据帧类型是 IP_SAP，用表项内容中的源 IP 地址替换掉目标 SAP 数据的旧 IP 地址，待数据帧结束搬移后，可跳入下一状态。

TRANS_UDP_SAP：当硬件协处理器加速表显示当前数据帧类型是 UDP_SAP，用表项内容中的源 UDP 地址替换掉目标 SAP 数据的旧 UDP 地址，待数据帧结束搬移后，可跳入下一状态。

REQUEST_IN_QUEUE：当硬件协处理器加速表显示当前数据帧类型是 MAC_RAW，数据不需要任何处理，此时向队列管理模块发送入队请求，可跳入下一状态。

WAIT_FOR_ACK：等待队列管理模块给出的反馈，并依据反馈信号决定跳入 BE_TRANS 还是 BE_DISCARD 状态。

BE_TRANS：如果队列管理有剩余空间供完整 MAC_RAW 帧存储，将完整数据帧传送给队列管理模块，返回初态。

BE_DISCARD：如果队列管理没有剩余空间供完整 MAC_RAW 帧存储，表明上层应用数据量异常，此时需要丢弃数据帧，返回初态。

4.1.4 入队申请模块

（1）模块功能概述

入队申请模块从 COM 数据组帧模块和 SAP 数据帧头替换模块接收数据及信息，与队列管理模块交互，确认是否有足够空间供完整数据帧存储，从而决定传输数据帧还是丢弃数据帧。

（2）模块接口框图

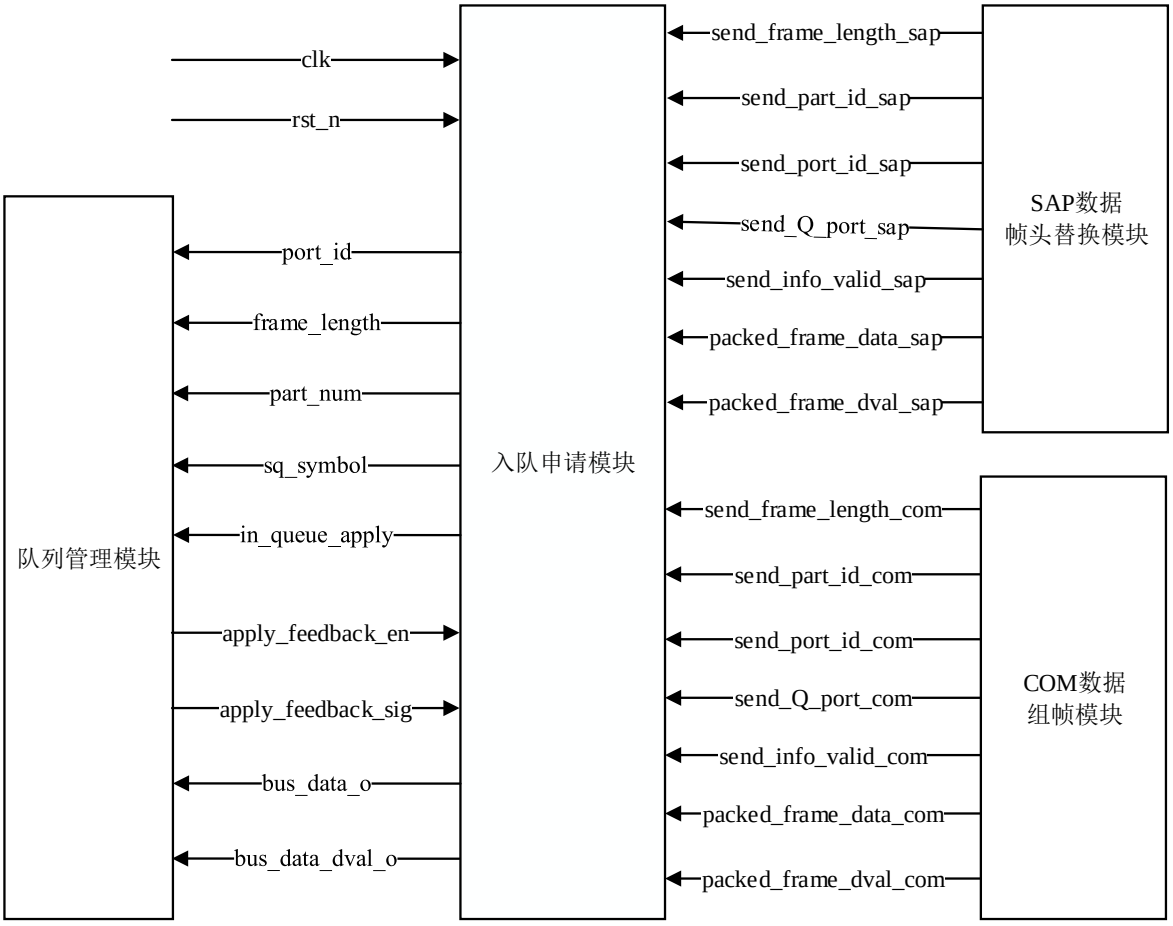

图4.8 入队申请模块接口

（3）模块详细设计

入队申请模块会先存储从 COM 数据组帧模块和 SAP 数据帧头替换模块接收到的数据及信息，以避免一组数据正在传输时另一组数据的丢失。

如果缓存中数据及数据信息 FIFO 皆非空，入队申请模块可以与队列管理模块开始交互，首先将诸如端口号、帧长、分区号等信息与入队申请信号同时提交给队列管理模块，队列管理模块计算结束后会返回反馈信号，此模块根据反馈信号决定数据的传输与舍弃。

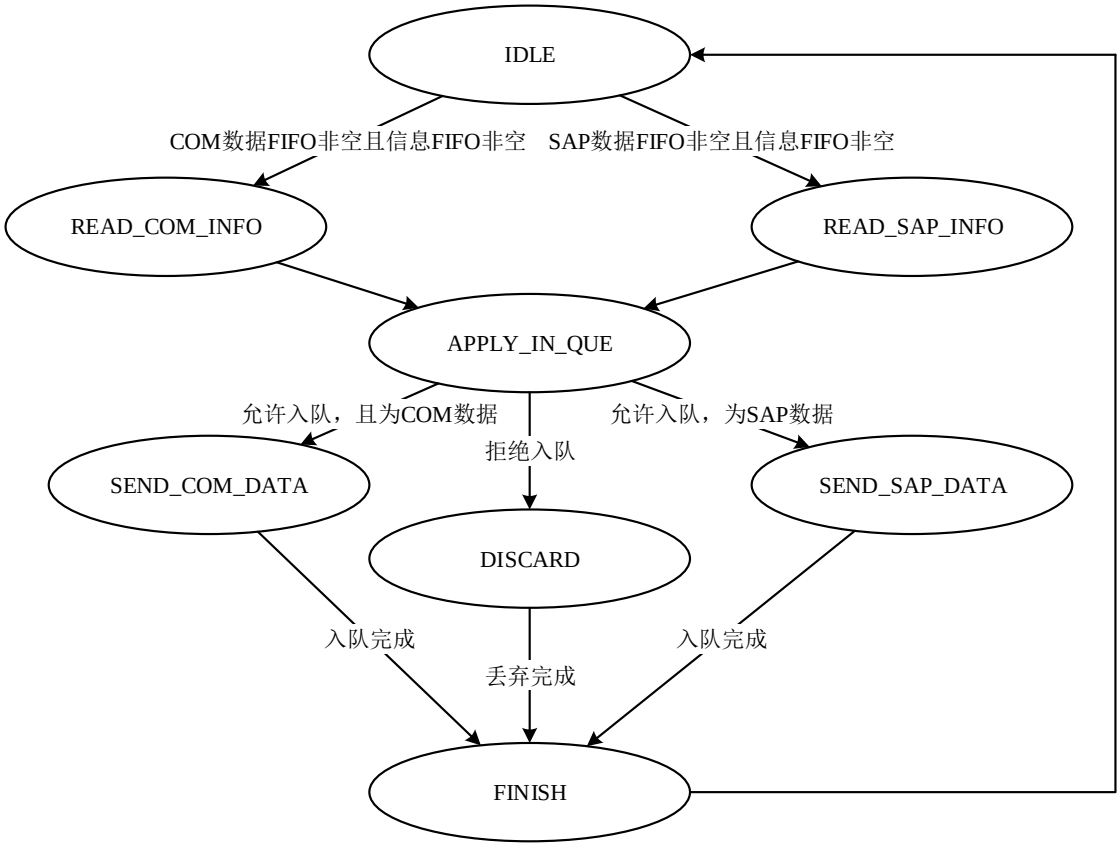

图4.9 入队申请状态机

IDLE：初态，根据存储 COM 数据、SAP 数据及它们信息的 FIFO 是否为空决定下一步跳入哪个状态。

READ_COM_INFO：从 COM 数据帧信息 FIFO 中提取需要的信息并加以寄存。

READ_SAP_INFO：从 SAP 数据帧信息 FIFO 中提取需要的信息并加以寄存。

APPLY_IN_QUE：向队列管理模块发送入队请求，等待队列管理模块给出的反馈，并依据反馈信号决定跳入 SEND_COM_DATA 或 SEND_SAP_DATA 还是 DISCARD 状态。

SEND_COM_DATA：如果队列管理有剩余空间供完整 COM 数据存储，将完整数据帧传送给队列管理模块，传输完成后跳入 FINISH 状态。

SEND_SAP_DATA：如果队列管理有剩余空间供完整 SAP 数据存储，将完整数据帧传送给队列管理模块，传输完成后跳入 FINISH 状态。

DISCARD：如果队列管理没有剩余空间供完整 COM 数据或 SAP 数据存储，表明上层应用数据量异常，此时需要丢弃数据帧，丢弃完成后跳入 FINISH 状态。

FINISH：状态机结束，返回初态。

4.1.5 队列管理模块

（1）模块功能概述

对于不同类型的数据，队列管理采用不同的存储方式。COM 端口采样数据的旧数据会被覆盖，队列只容纳一帧，故而利用静态缓存事先规定每个端口采样数据的起始存储地址即可。MAC_RAW（可简单理解成 BE 帧）的业务优先级较低，帧长不固定，且不存在多个端口号，也利用静态缓存，划分一块固定的存储区即可。

其他类型的业务数据，队列中应能容纳几帧数据，但端口号众多，所以可以利用共享缓存[47]方式减少资源浪费，这种方法需要管理每个数据帧的帧长、头尾指针等信息。每存储一个数据帧或释放一个数据帧，队列管理都会更新信息，和其他模块交互，以确认存储区内是否还能容纳新的数据帧或是否可以有数据帧被调度。

（2）模块接口框图

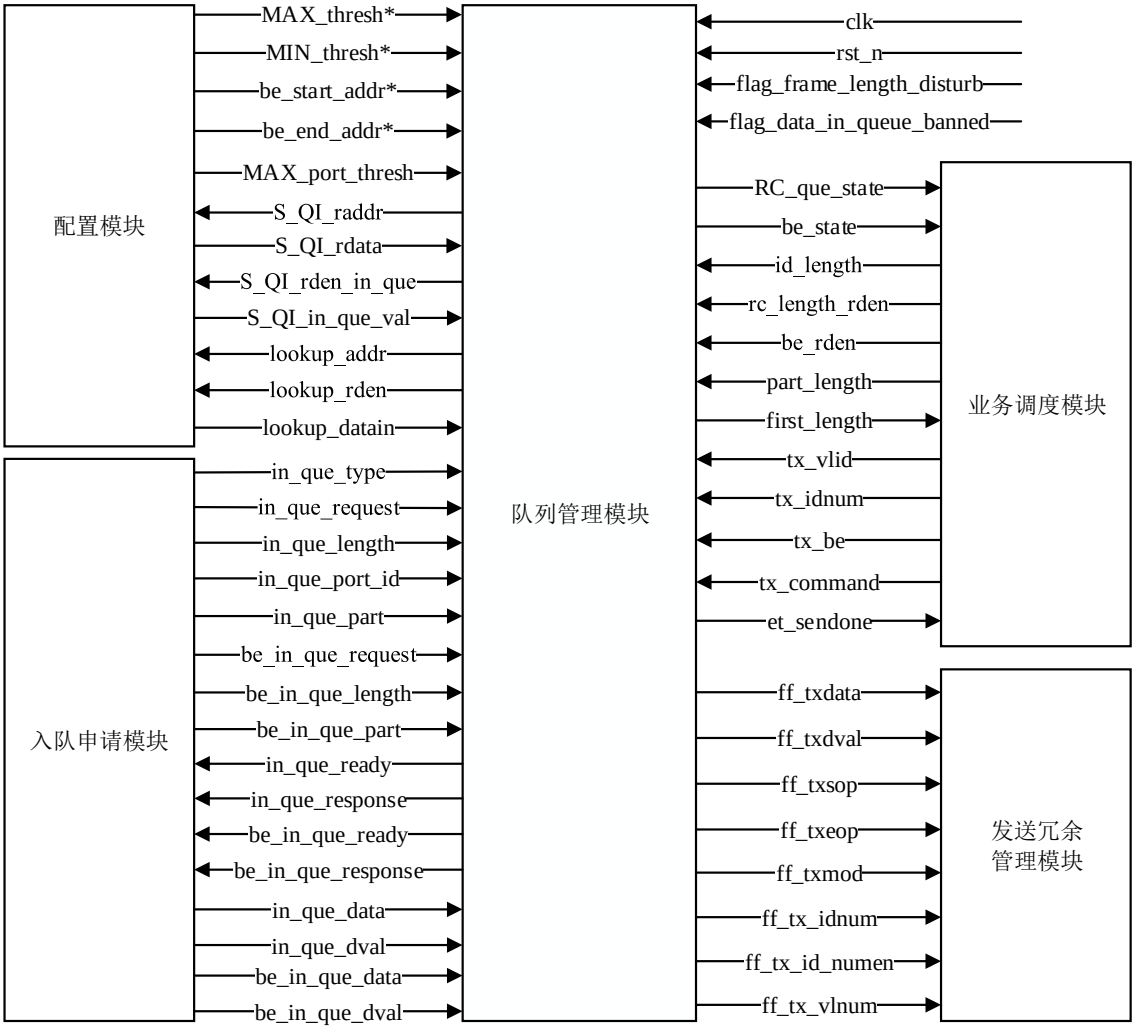

图4.10 队列管理模块接口

（3）模块详细设计

队列管理模块维护着所有数据的信息，例如每个端口数据帧的长度、头尾指针、已用的缓存区大小等等。当有数据申请入队时，首先查询该端口号下的剩余空间是否足够用于完整数据帧的存储，如果足够，则取出该端口队列的尾指针，从尾指针开始存储数据帧，待存储完成后将数据的所有信息更新，例如尾指针信息，新数据入队后采用共享缓存方式存储的数据尾指针变化，必须及时更新，既是为下一帧的入队做准备，也能在有数据出队时把释放的空闲缓存链接到当前空闲存储区之后。

对于非 S 数据和 BE 数据，因为采用共享缓存存储方式，以 16 个字节作为一个存储块，这种存储块被称为缓存描述符（Buffer Description，BD[48]）块，每个 BD 块存储数据并不是固定的，也不一定连续，所以当每一帧数据的存储耗费多个 BD 块时，必须要以链表的方式将这些散乱的存储块连接，以保证数据的连续性。

每接收一帧数据，队列管理模块都会更新，并把各队列的状态发送给调度模块，调度模块根据队列状态进行调度，直至发起调度请求和待调度的端口号，队列管理模块接收到后，开始查询队列信息，即此端口数据的头指针、帧长等，将数据从缓存区搬移至输出总线后更新旧信息。

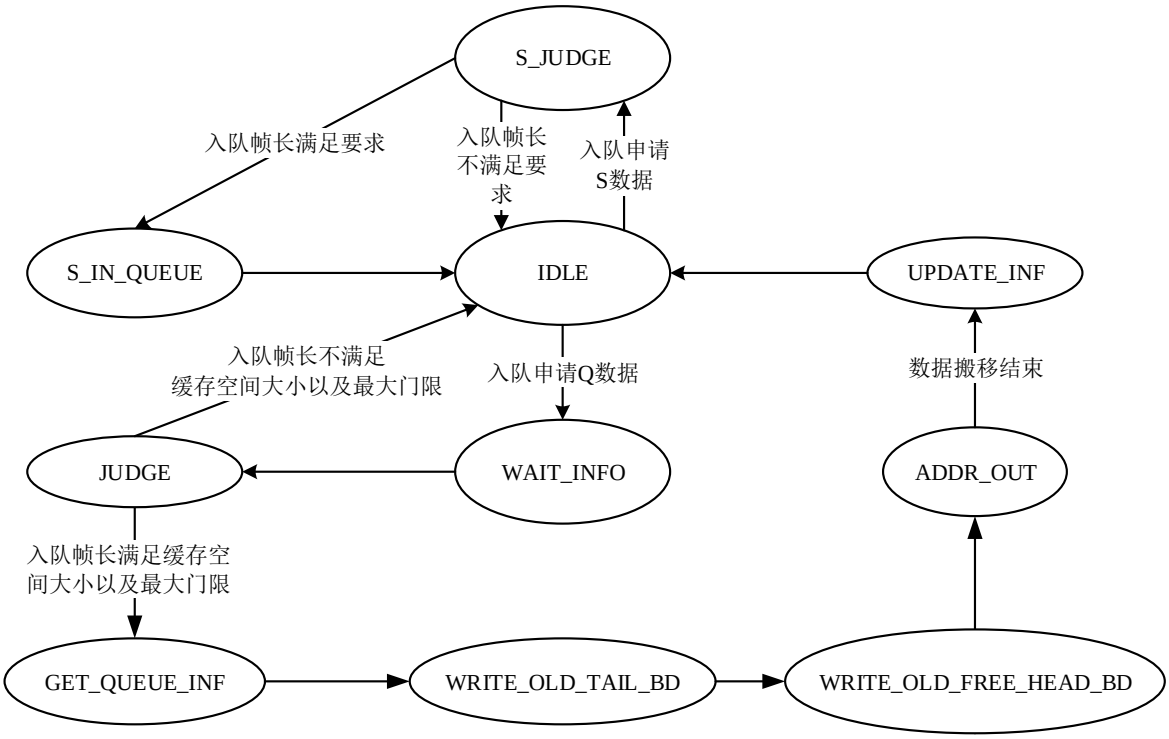

图4.11 队列管理入队状态机

IDLE：初态，根据入队申请模块传来的入队请求信号和 Q 标识决定跳入 WAIT_INFO 还是 S_JUDGE 状态。

WAIT_INFO：如果申请入队数据不是 S 数据，则进入此状态，此时读取队列信

息，跳入下一状态。

S_JUDGE：如果申请入队数据是 S 数据则进入此状态，此时读取 S 数据起始地址存储表，判断将入队数据帧长是否符合要求，符合就跳入下一状态，否则返回初态。

S_IN_QUE：S 数据申请入队成功，记录此数据信息为出队所用，进入下一状态。

JUDGE：根据申请入队数据的帧长和上一状态读出的队列缓存区使用情况，判断是否能接纳新的数据帧，如果可以就跳入 GET_QUEUE_INF 状态，否则返回初态。

GET_QUEUE_INF：读取该端口队列的信息为数据入队做准备，跳入下一状态。

WRITE_OLD_TAIL_BD：新数据入队导致队列的尾指针发生变化，需将新数据的存储起始地址与旧的尾指针相连接，跳入下一状态。

WRITE_OLD_FREE_HEAD：记录数据帧长，此状态会在新数据的存储起始地址处写入帧长，之后跳入下一状态。

ADDR_OUT：给出新数据的存储地址，待存储完成之后跳入下一状态。

UPDATE_INF：因有新数据入队，原有信息都要更新替换，更新完成后返回初态。

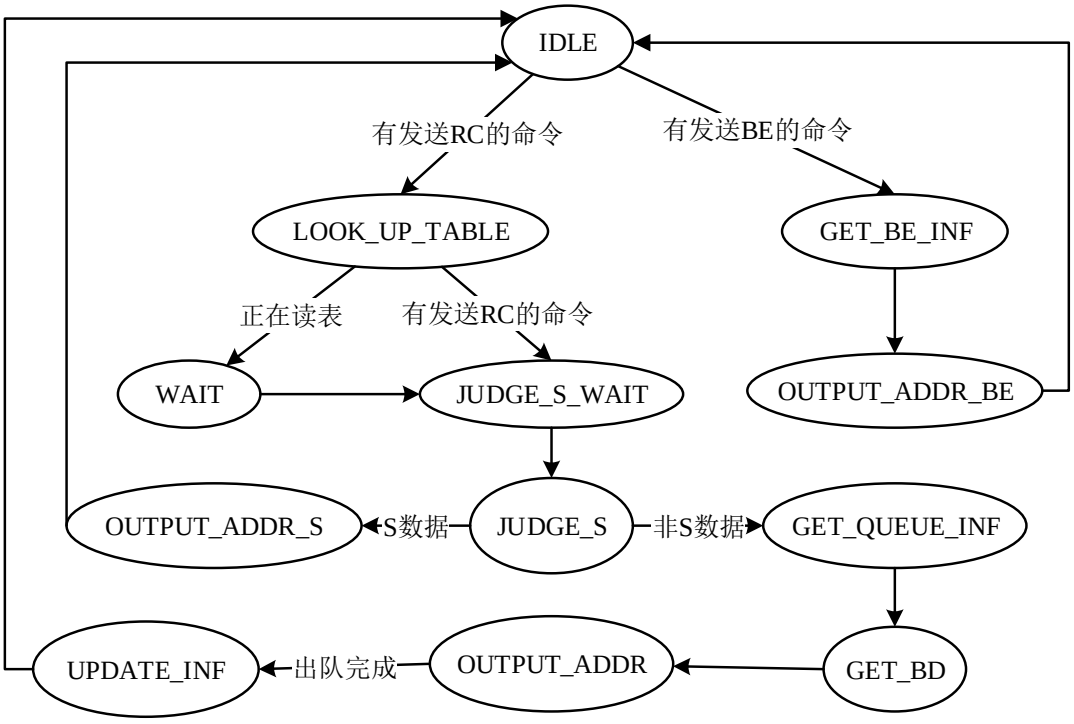

图4.12 队列管理出队状态机

IDLE：初态，如果业务调度模块传来 RC 发送指令，跳入 LOOK_UP_TABLE 状态，传来 BE 发送指令则跳入 GET_BE_INF 状态。

LOOK_UP_TABLE：根据调度端口号获取调度数据的帧长，之后可进入下一状态。

WAIT：如入队出队同时使用 S 数据起始地址存储表，则做等待，进入下一状态。

JUDGE_S_WAIT：获取 S 数据起始地址存储表表项内容，进入下一状态。

JUDGE_S：由上一状态获知的表项内容，可确定待调度数据帧是否为 S 数据，倘若是则进入 OUTPUT_ADDR_S 状态，否则进入 GET_QUEUE_INF 状态。

OUTPUT_ADDR_S：根据从 S 数据起始地址存储表中获知的数据存储起始地址，将 S 数据从队列中搬移出来，而后返回初态。

GET_QUEUE_INF：获取待调度端口的信息为数据出队做准备，进入下一状态。

GET_BD：由上一状态可获取当前数据帧存储的起始地址，进入下一状态。

OUTPUT_ADDR：据存储链表将完整数据帧从队列中搬移出来，进入下一状态。

UPDATE_INF：因有数据出队，原有信息都要更新替换，更新完成后返回初态。

GET_BE_INF：读取 BE 数据帧队列中第一帧的长度等信息，进入下一状态。

OUTPUT_ADDR_BE：根据上一状态获知的信息将 BE 数据从队列中搬移出来，更新原有信息后返回初态。

4.1.6 业务调度模块

（1）模块功能概述

端系统内有两种业务：RC 业务和 BE 业务，前者的优先级比后者的优先级更高。当队列管理将 RC 的队列信息和 BE 的分区信息传递而来时，业务调度模块会根据这些数据是否满足调度条件，同时注意 MAC 核是否已准备好，如若条件都满足则会给队列管理模块发送调度请求和端口号或分区号。

（2）模块接口框图

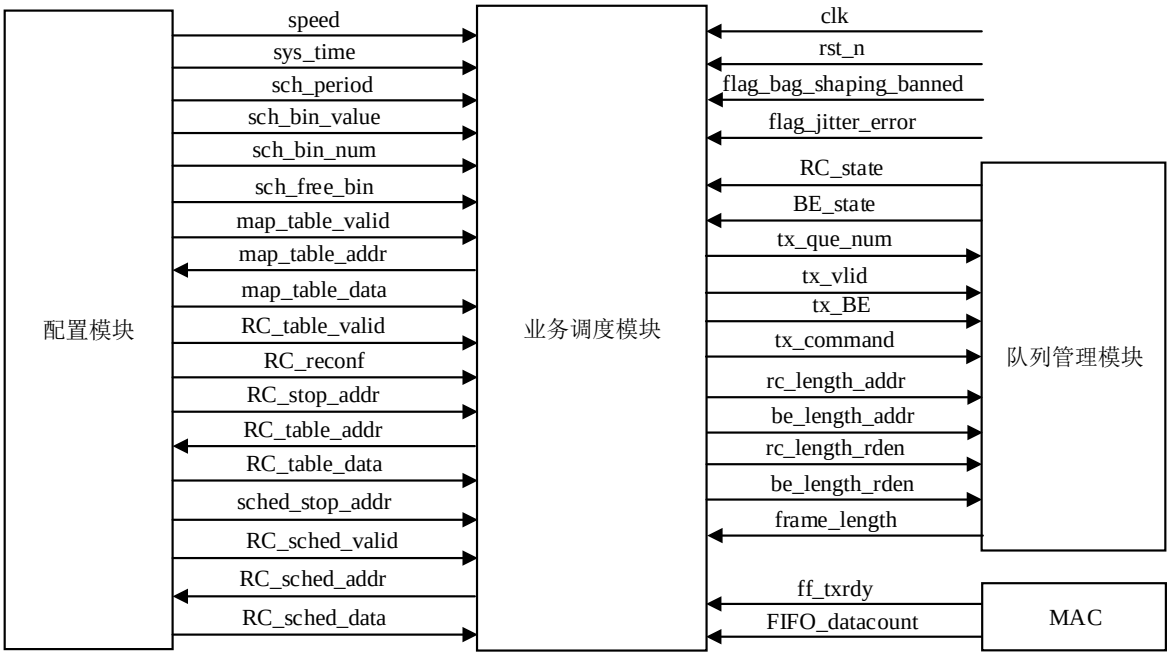

图4.13 业务调度模块接口

（3）模块详细设计

业务调度模块会根据时间计数值确定当前处于第几个调度片，并在进入调度片的起始点读取该调度片的业务调度表表项，从中获取待调度 RC 数据的端口号。而后查询该端口号的数据帧间的时间间隙是否已经满足 BAG 要求，如不满足则不能调度，如果满足则查询子 VL 映射表，采用公平轮询的方式，结合 RC 队列状态表判别队列中是否存在数据帧。倘若有数据帧，在向队列管理模块发送调度请求和端口号之前，需要判别 MAC 核可不可以接收数据帧，可以的话即向队列管理模块发送读使能，读取该端口号的数据帧首帧长，以预留足够的时间供数据帧搬移。

BE 数据的优先级比 RC 数据低，当业务调度表中某一条表项未规划 RC 业务或调度片长度大于帧长，这些时间间隙都可以被利用于 BE 数据的调度。

BE 调度的流程很简单，因为没有 BAG、子 VL 的概念，所以只需根据 BE 分区状态查看是否有 BE 数据，获取 BE 帧长，MAC 核能否接收数据帧即可向队列管理模块发送调度请求和分区号。

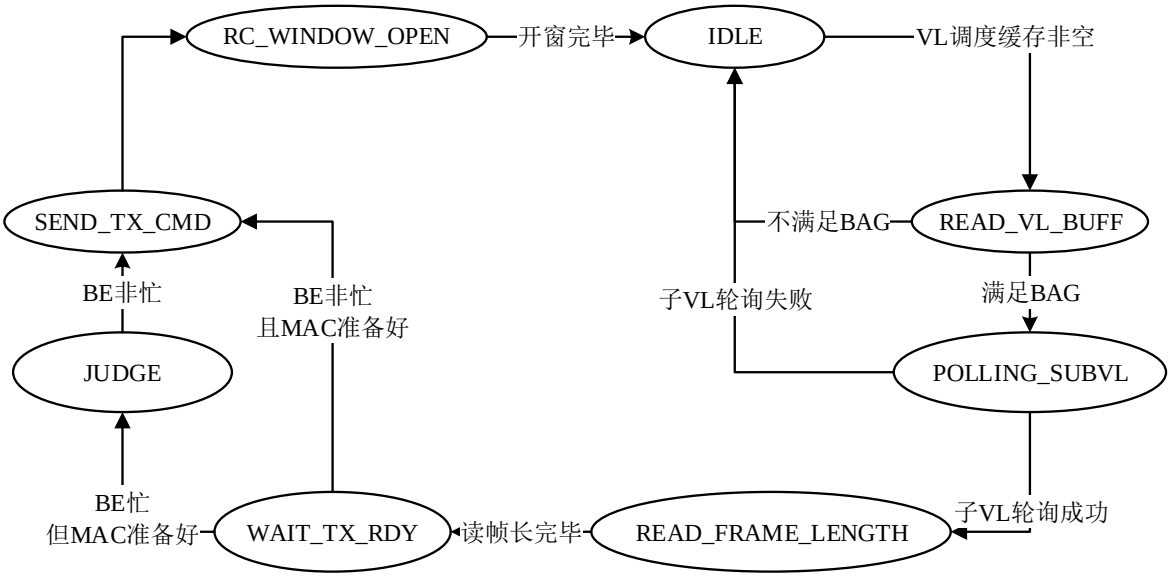

图4.14 业务调度状态机

IDLE：初态，当业务调度表中有 RC 数据帧被规划时跳入下一状态。

READ_VL_BUFF：获取待调度数据的端口号，同时确认此端口号的数据帧间的时间间隙，如果时间间隙大于配置的 BAG 则跳入 POLLING_SUBVL，否则返回初态。

POLLING_SUBVL：由上一状态获取的端口号，根据子 VL 映射表公平轮询，查询队列状态表，确认子 VL 中是否存有数据，如有则跳入 READ_FRAME_LENGTH 状态，否则返回初态。

READ_FRAME_LENGTH：从队列管理模块获取数据帧长，以计算时间窗长，跳入下一状态。

WAIT_TX_RDY：根据 MAC 传递来的信号，确认其是否可以接收并发送新的数据帧，如果可以接纳新数据且此时未有数据在搬移则跳入 SEND_TX_CMD 状态，否则跳入 JUDGE 状态进行等待。

JUDGE：等待尚在搬移的数据帧完成操作，直至此时新的数据帧可以开始搬移，跳入下一状态。

SEND_TX_CMD：向队列管理模块发送调度指令，跳入下一状态。

RC_WINDOW_OPEN：开始搬移数据帧，千兆速率下数据帧搬移长度与从队列管理模块获取的数据帧长相等则可返回初态，其他速率下则需与时间窗长相等。

4.1.7 发送冗余模块

（1）模块接口框图

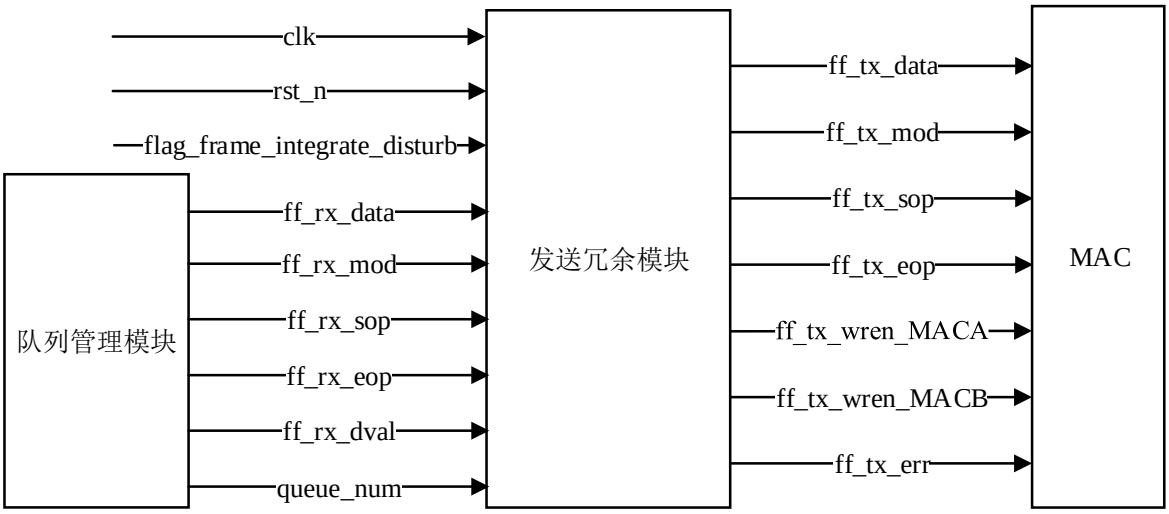

图4.15 发送冗余模块接口

（2）模块详细设计

此模块功能实现较为简单，根据调度结果，数据帧从队列管理模块搬移结束后，需要定位数据帧的起始、结束并提供有效字节数、队列号等信息，之后在 RC 数据帧的尾部添加序列号，同一虚拟链路号的数据帧序列号循环递增，不同虚拟链路号数据帧之间互不影响。同时，此模块会复制数据帧，两个网口的数据写使能都有效，完成冗余功能。BE 数据帧没有序列号一说，在这个模块只需要解析 MAC 地址，确认从哪个网口发送出去即可。

4.2 接收侧模块划分

4.2.1 预处理模块

（1）模块功能概述

预处理模块包括序列号移除、帧信息提取、完整性检查和接收冗余管理功能，负责处理数据帧，并完成部分帧过滤功能。当数据帧为不符合要求的异常帧时，向存储控制模块发送丢弃指令，清空相应缓存，否则把数据信息往后级传输。

（2）模块接口框图

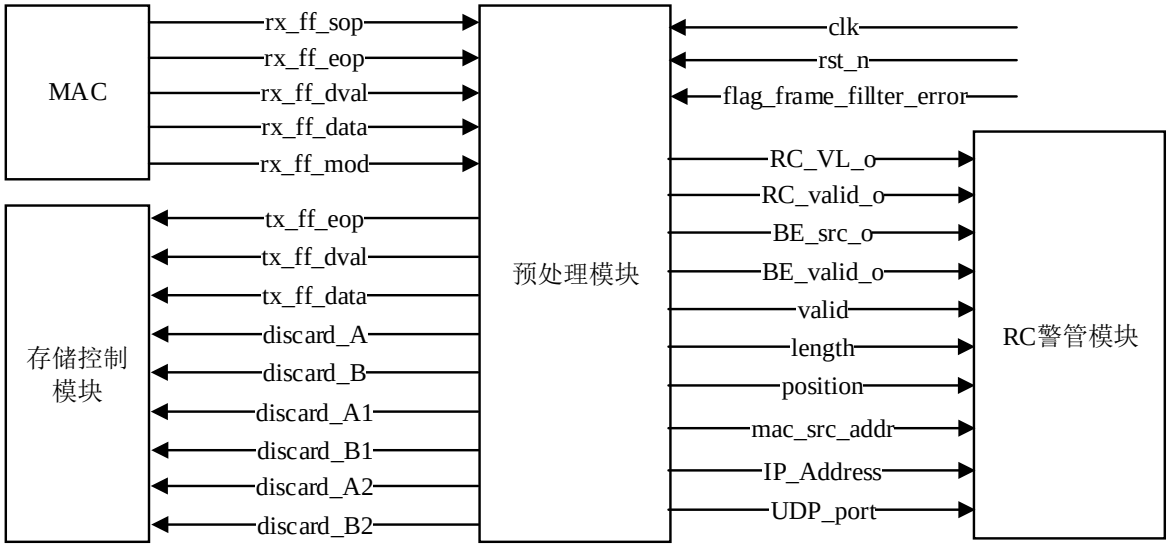

图4.16 预处理模块接口

（3）模块详细设计

数据帧中携带很多后续模块所需的帧信息，例如虚拟链路号、MAC 地址、IP 地址等等，帧信息提取子模块会从 MAC 侧接收数据，提取帧信息给后续模块。同时此模块还会进行 MAC 地址过滤，如发现不符合要求的数据帧，将会给存储控制模块传递丢弃信息以清空存储该帧的缓存。

发送侧发送冗余模块会为 RC 数据帧增加序列号，帧信息提取子模块从数据帧中解析出数据帧隶属 RC 业务还是 BE 业务的标识后，将标识信息传递给 SN 移除子模块，在将 RC 数据帧帧尾的序列号移除完后，把数据帧上交给存储控制模块暂存，而 BE 数据帧不携带序列号，故而将被直接传递给存储控制模块。

正常情况下，同个虚拟链路号下的数据帧应该是连续的，完整性检查子模块根据帧信息提取模块提供的序列号，检查接收的数据帧的序列号是否连续，假设上一个数据帧的序列号为 SN1，而当前接收数据帧的序列号为 SN1+1 或 SN1+2，则判定数据帧完整性检查通过，否则将会给存储控制模块传递丢弃信息，从而清空存储该帧的缓存[49][50]。

端系统发送侧会将数据帧复制成两份，经两个网口发送，对于同一个接收端，当网络 A 和网络 B 都正常时，会收到两份相同的数据，势必要丢弃一组重复数据，此时可根据完整性检查子模块传递而来的数据信息，主要是虚拟链路号和序列号，两个先后到达的数据帧，先接收前一帧，把相关信息存储，待下一帧到达，对比发现数据信息相同时，向存储控制模块发送丢弃指令，清空相对应的缓存区[51]。

4.2.2 RC 警管模块

（1）模块功能概述

发送端发送 RC 数据帧时，同虚拟链路号下相邻两帧的时间间隙必须符合 BAG 要求，而在接收端也要检查同虚拟链路号下的 RC 数据帧帧间间隙，如不符合要求则可视为异常帧，需被丢弃。

（2）模块接口框图

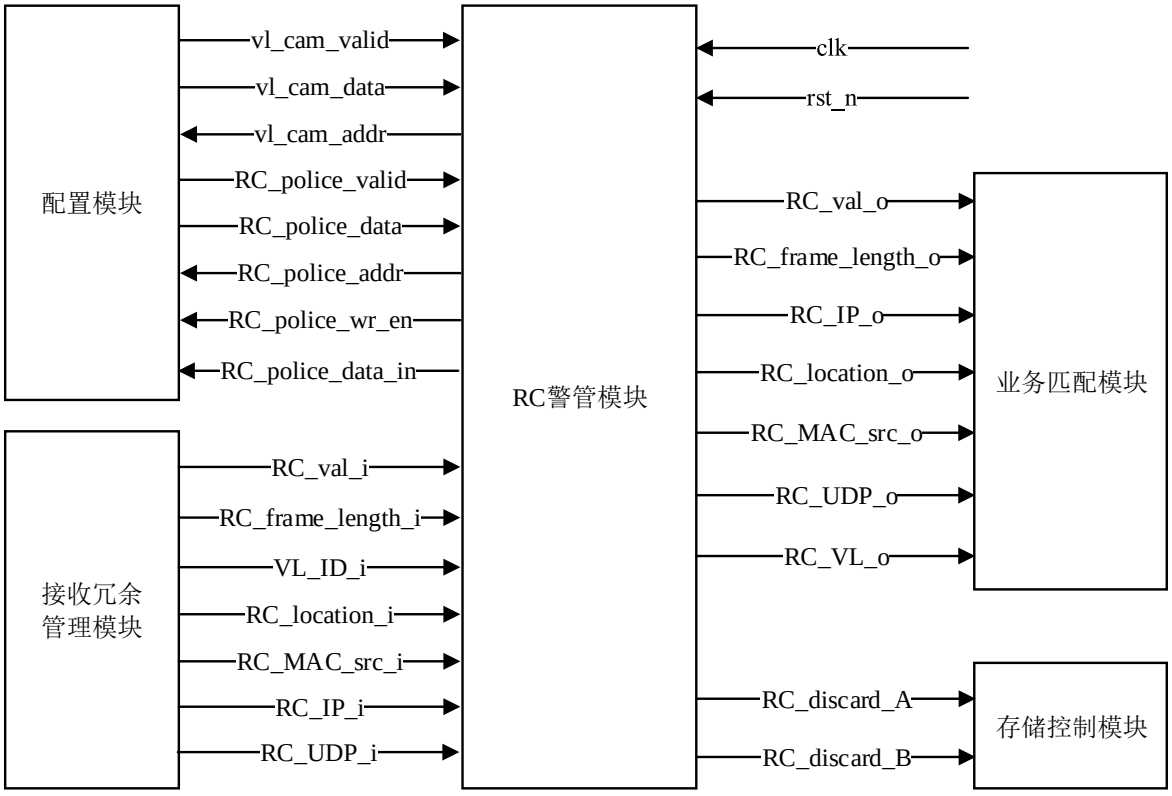

图4.17 RC 警管模块接口

（3）模块详细设计

每个虚拟链路号下第一帧 RC 数据永远满足 BAG 的要求，不需要警管，但每接收一帧数据，都要记录当前的时间计数值。下一帧到达时，开始计算相同虚拟链路号下 RC 数据帧的帧间间隙。

当前时间计数值加上剩余信用值，再减去上一帧数据到达的时间，等于两帧间的

时间间隙。如果差值大于预先配置的 BAG，表明时间间隙符合 BAG 要求，警管完成，该帧符合要求，否则警管失败，该帧数据异常，不允许传输至后级模块，此时要给存储控制模块传递丢弃信号，清空相应缓存[52][53]。

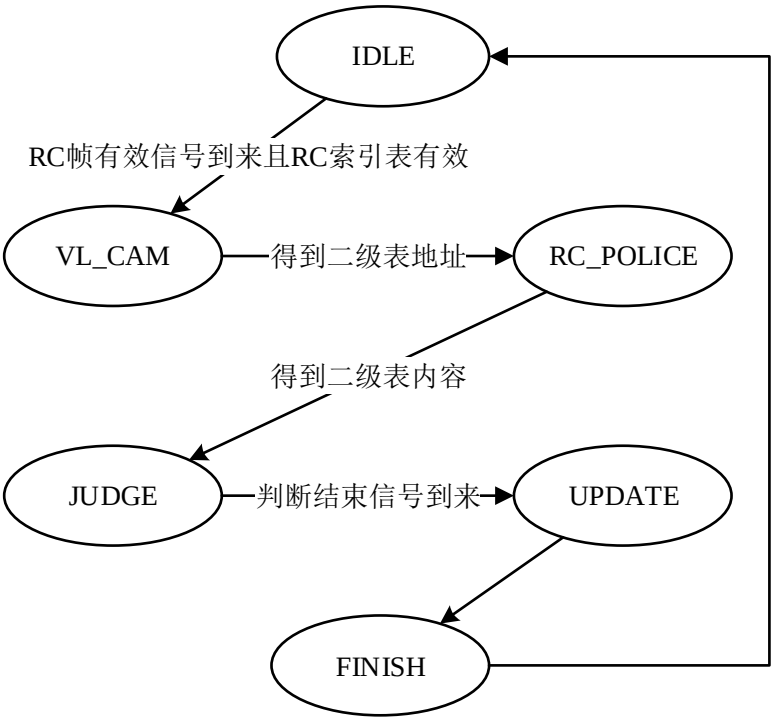

图4.18 RC 警管状态机

IDLE：初态，当 RC 数据帧信息到来时跳入下一状态。

VL_CAM：将 RC 数据帧虚拟链路号作为查表地址，查询 RC 索引表，进入下一状态。

RC_POLICE：将索引表中的表项内容作为 RC 警管表的查表地址，得到表项内容，进入下一状态。

JUDGE：通过 RC 警管表中的表项内容，计算当前数据帧与上一数据帧之间的时间间隙是否满足 BAG 要求，如果满足 BAG 要求，则该数据帧正常，否则丢弃，进入下一状态。

UPDATE：如果当前数据帧正常，则记录此时的剩余信用值、到达时间等信息，更新 RC 警管表。

FINISH：完成 RC 数据的警管，返回初态。

4.2.3 业务类型匹配模块

（1）模块功能概述

从数据帧中提取虚拟链路号、UDP 端口号、IP 地址，唯一确定与应用端口对应

的数据帧，分辨出数据属于 COM 数据还是 SAP 数据，属于 S 数据还是 Q 数据，并从业务类型匹配表、VL 业务匹配表中查询接收端口号等信息，随数据流传递到下一模块。如发现表项中无此业务，则需进行丢弃操作。本模块是使 AFDX 端系统可面向多业务场景的关键模块之一。

（2）模块接口框图

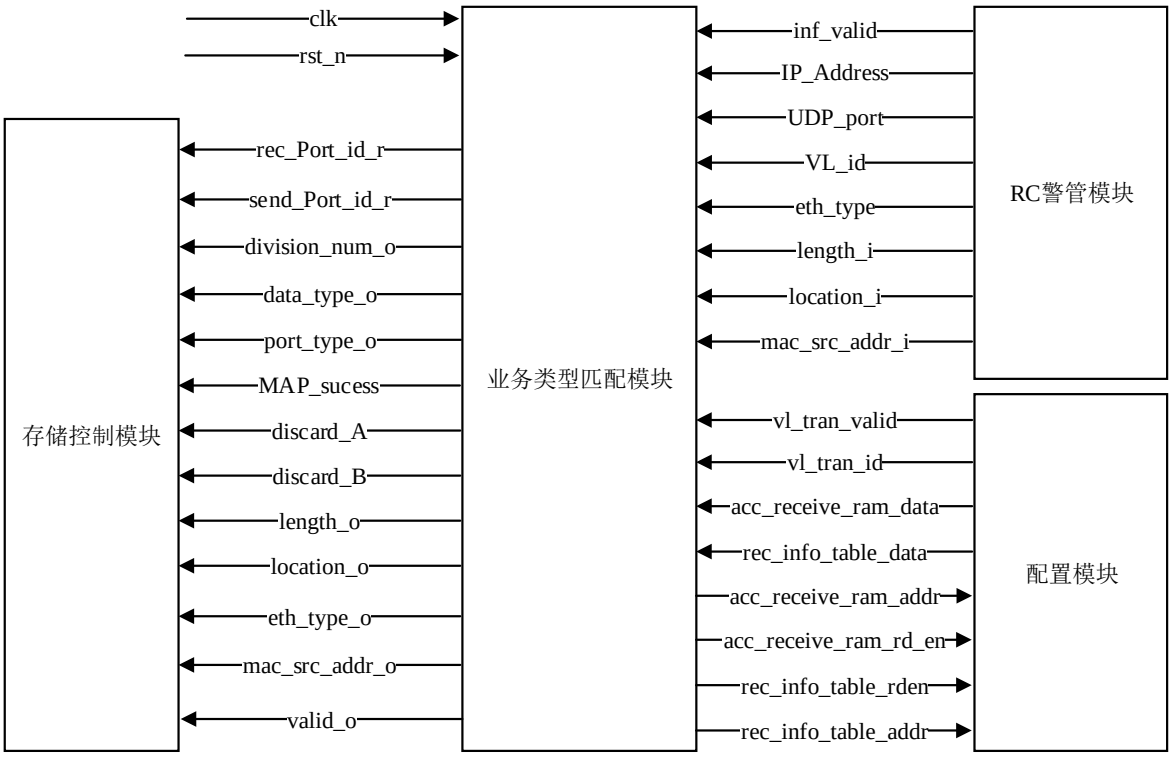

图4.19 业务类型匹配模块接口

（3）模块详细设计

端系统接收侧需要在接收数据帧后查询其业务类型，以便数据能被上层固定端口接收，而不同数据帧携带的信息繁复，如何从纷杂的信息中搜查特定的信息，就需要依赖查表方法。常见查表方法有顺序查找、二分查找、分块查找、哈希查找[54]等。

顺序查找从表格的源头开始查询，地址逐一递增，直至能够查找到匹配项，如果所有表项均与目标不相符，则表示查找失败，顺序查找实现方法简单，但会耗费过多时间。二分查找需要事先知道表格深度，通过比对表项内容与目标内容的结果，决定新一次二分查找的上限和下限，直至查表成功或失败，它的查表操作更为灵活，但应用场景仍有所限制，依然相当于一种盲目查询。分块查找需要在查找表之前添加索引表，索引表中包含每个关键词的最大值和所处的第一个位置，且根据关键词进行排序，分块查找更像顺序查找和二分查找的复合体，能较快的查找到匹配表项。

而哈希查找中内容的存储位置既不是从起始点随意存储，也不是按照顺序存储，而是由哈希函数指定，哈希函数对输入的信息进行计算，计算的结果即为存储地址，

这种存储方式能够在查表时快速查到地址，但同时仍存在一个问题，不同数据的哈希函数结果可能相同。由于端系统在接收数据时，数据流是源源不断的，不可能等着上一个数据处理完成后才往后处理，所以适合采用哈希查找方式。

哈希冲突问题可以采用线性探测法，即当多个数据的存储位置相同时，可以利用某个空闲位置作为冲突存储位置，且在第一次查询的地址表项内容里指定冲突下一跳地址，这种处理方式符合项目实际需求，故而内部将采用哈希查表法，另外由于信息过长，哈希生成式采用循环冗余检查（Cyclic Redundancy Check ，CRC[55]）生成式，将长信息转化为短地址。

当数据帧信息传递至业务类型匹配模块时，首先会根据虚拟链路号查询此数据是否为 SAP 数据，如果是则通过 VL 索引表将虚拟链路号映射为 VL 业务查找表的查找地址，从中获取该数据帧业务的接收端口号、发送端口号、分区号等信息，之后把信息传递给存储控制模块。

如果不是 SAP 数据，则通过虚拟链路号、IP 地址、UDP 端口号计算出业务类型匹配表的查表地址，从查表内容中获知该数据隶属于 S 数据还是 Q 数据，并比对表项内容和数据信息，如果信息相同表明匹配成功，如发生冲突则进入下一跳地址继续查询，匹配成功后将表项内容和数据信息一同传递给存储控制模块。

如果最后发现 VL 业务查找表和业务匹配查找表中都没有表项和数据信息匹配，则表明当前数据帧并不是预期数据，很可能是系统异常而导致端系统接收到错误数据帧，针对这种情况，端系统内部应该执行过滤操作，向存储控制模块传递丢弃信号，清空相应缓存。

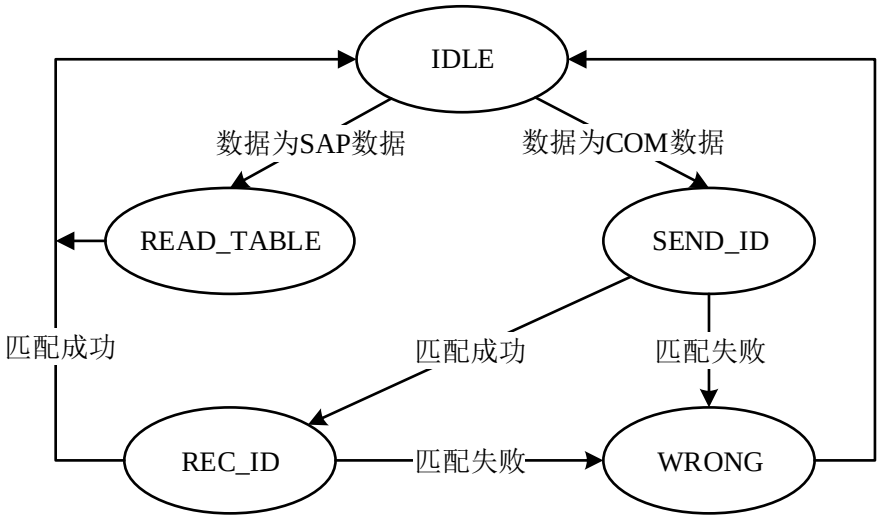

图4.20 业务类型匹配状态机

IDLE：初态，当查询到 VL 索引表中表项内容有效时，说明数据帧隶属 SAP 类型，进入 READ_TABLE 状态，否则进入 SEND_ID 状态。

READ_TABLE：将 VL 索引表的表项内容作为查表地址，查询 VL 业务查找表，获取表项内容后传递给后续模块，返回初态。

SEND_ID：通过 UDP、IP 及虚拟链路号计算查表地址，查询业务类型匹配表，得到数据信息并比对，如比对成功则进入 REC_ID 状态，否则进入 WRONG 状态。

REC_ID：继续查表并匹配，比对数据信息及表项内容，如果失败则进入 WRONG 状态，成功则将信息传递给后续模块，返回初态。

WRONG：VL 业务查找表和业务类型匹配表均未与当前数据帧匹配成功，业务类型匹配模块向存储控制模块发送丢弃指令，之后返回初态。

4.2.4 存储控制模块

（1）模块功能概述

存储控制模块内部设有两个 FIFO，用于缓存从 A 网络和 B 网络接收的数据，根据预处理模块等前级模块的结果决定数据帧应该被丢弃还是传输至下一级模块进行处理。

（2）模块接口框图

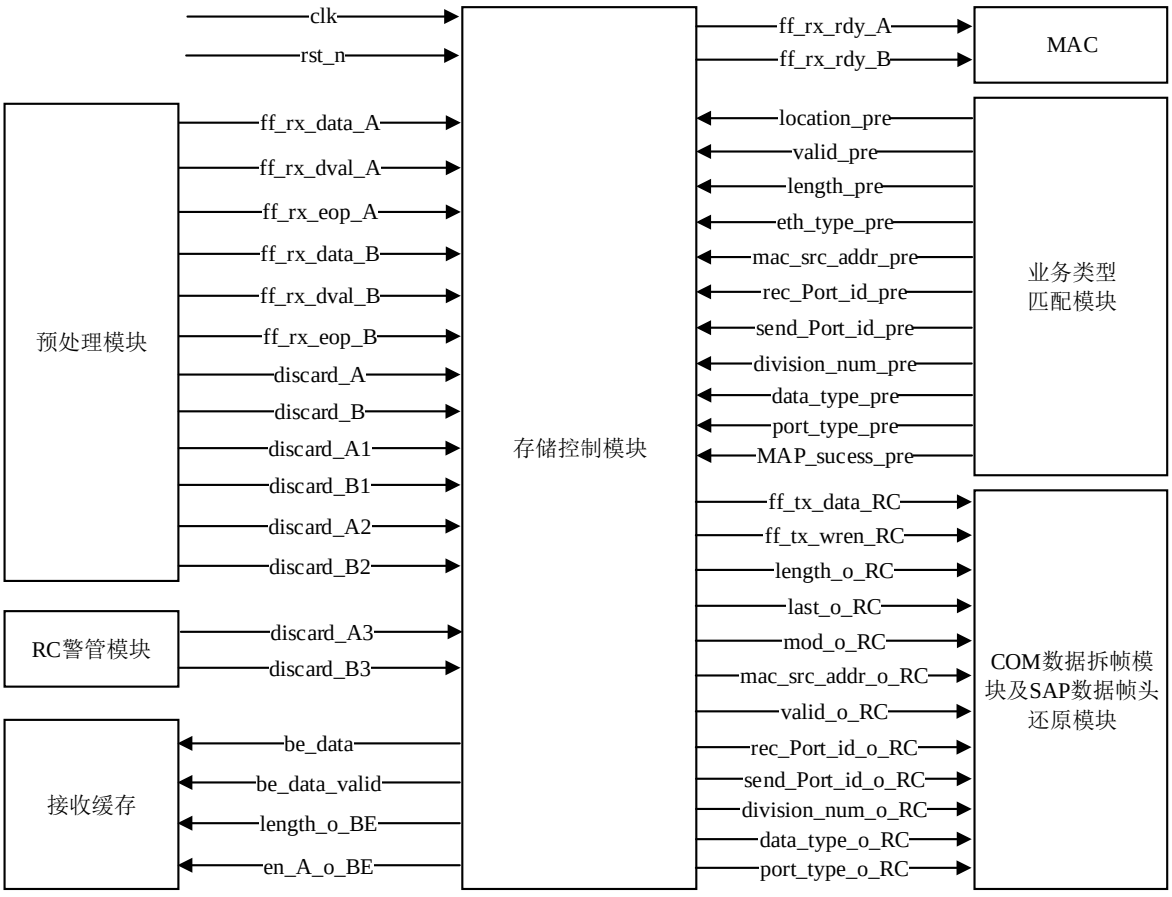

图4.21 存储控制模块接口

（3）模块详细设计

预处理模块、RC 警管模块、业务类型匹配模块等皆与存储控制模块相关联。为防止端系统内部模块处理数据信息时间较长而导致数据帧被覆盖，存储控制模块内部设有乒乓 FIFO，每个 FIFO 支持最长帧的存储，不同网口传输的数据使用的 FIFO 不同，两个 FIFO 交替存储，时刻更新标志信号以供异常数据的丢弃和正确数据的读取。

经 SN 移除后，数据帧存入乒乓 FIFO 中，此时帧信息提取等模块开始处理该数据帧，判别此数据帧是否正常，如不正常则发送丢弃指令，存储控制模块根据丢弃指令及乒乓 FIFO 存储的标志信号丢弃相应数据帧，如果数据帧正常，则会从业务类型匹配模块中获取数据帧的帧信息和查表得到的表项内容，并将乒乓 FIFO 中的数据帧取出，一同传递给下级模块。

另外，如果数据已经知晓为 MAC_RAW 类型数据（BE 数据帧），因为不需要进行任何处理，故而直接通向接收缓存待上层应用读取。而 RC 数据帧则进入 COM 数据拆帧模块及 SAP 数据帧头还原模块处理。

4.2.5 COM 数据拆帧及 SAP 数据帧头还原模块

（1）模块功能概述

由于接收侧 COM 数据拆帧及 SAP 数据帧头还原处理过程有一定程度的重合，故而接收侧合并为一个模块处理，与发送侧功能相反，接收侧会去除 COM 数据帧的帧头，将其还原为裸数据，SAP 数据会把原来的帧头替换回来，以保持应用下发数据与接收数据的一致性。本模块是使 AFDX 端系统可面向多业务场景的关键模块之一。

（2）模块接口框图

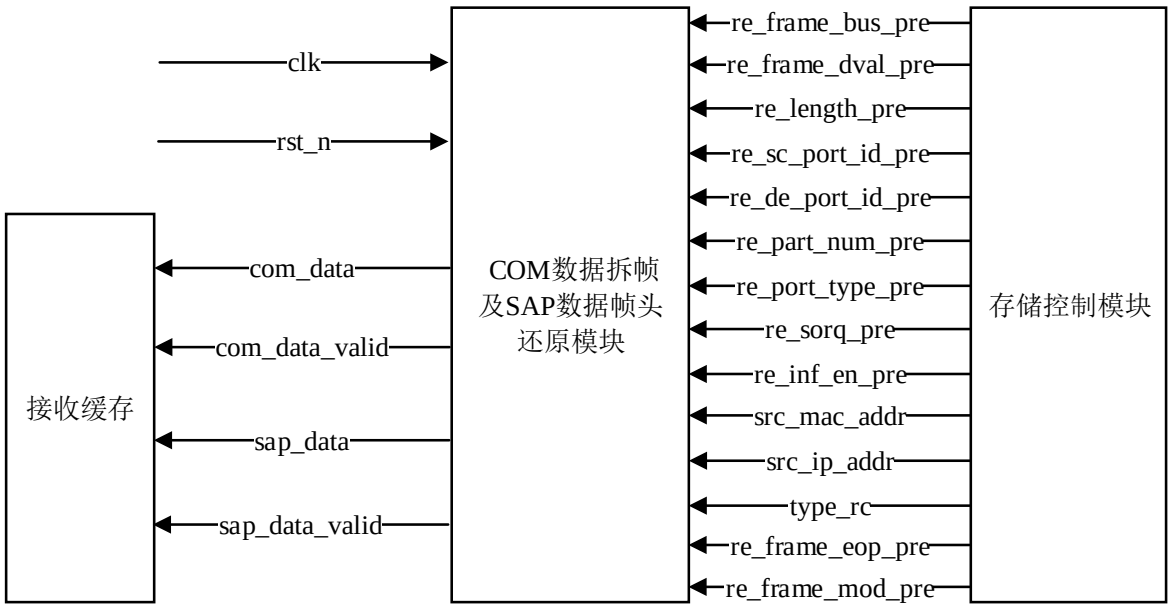

图4.22 COM 数据拆帧及 SAP 数据帧头还原模块接口

（3）模块详细设计

从存储控制模块接收数据及数据信息后，首先会把数据及信息存放入模块内缓存中，根据帧信息可以判别数据隶属于 COM 类型还是 SAP 类型。

如果是 COM 数据，则需要把数据载荷以及帧头部递交给模块内部 IP 头校验子模块和 UDP 头检验子模块。IP 头校验子模块和 UDP 头检验子模块重新计算数据的校验和，与接收数据自带的校验和比较，如果相等则说明数据帧传输过程没有出错，可以将 COM 数据的帧头拆除。

从何处开始拆除与数据业务类型相关，MAC_COM 需要去除 MAC 头，IP_COM 需要去除 MAC 头和 IP 头，UDP_COM 需要去除 MAC 头、IP 头和 UDP 头。COM 数据去除完帧头之后，应该在帧首加上 8 字节帧头信息，即帧起始标识、分区号等信息，这些信息都需要传递给应用以便于数据帧识别。

SAP 帧的处理流程要简单一些，只需要根据数据隶属于 MAC_SAP、IP_SAP 还是 UDP_SAP，相应地替换数据帧头的源 MAC 地址、源 IP 地址和源 UDP 地址即可，而这些信息业务类型匹配模块已给出。之后像 COM 数据的处理流程一样，添加帧起始标识、分区号等信息，把添加好帧头信息的数据传送至接收缓存，等待上层应用读取。

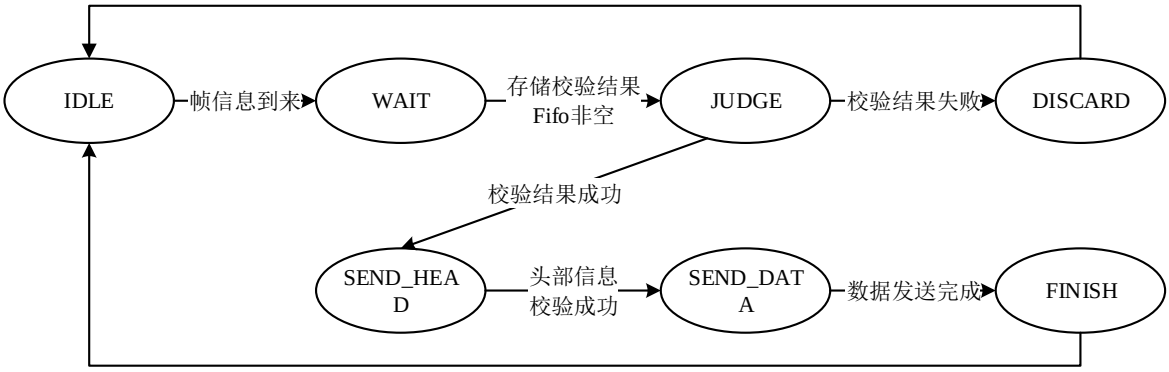

图4.23 COM 数据拆帧及 SAP 数据帧头还原状态机

IDLE：初态。存储控制模块将 RC 数据帧及帧信息传递至此模块后，可跳入下一状态。

WAIT：将数据帧、UDP 头及 IP 头送至校验模块，计算校验和，进入下一状态。

JUDGE：检查计算得到的检验和与数据帧中的校验和是否相等，若相等则可进入 SEND_HEAD 状态，否则表明这是一个错误数据帧，进入 DISCARD 状态。

DISCARD：针对错误的数据帧，不能被上传到应用层，而是要被丢弃，之后返回初态。

SEND_HEAD：根据数据帧所属的类型，如果是 COM 数据帧，丢弃原有帧头，

并添加 8 字节的帧头信息，如果是 SAP 数据帧，则需在对应位置替换帧头，之后进入 SEND_DATA 状态。

SEND_DATA：从缓存区中取出完整数据载荷，进入下一状态。

FINISH：完成数据的传输，返回初态。

4.3 故障注入模块

（1）模块功能概述

为模拟设备在真实情况下可能遇见的问题，如单粒子翻转、模块出错等故障，设计故障注入模块，与多模块耦合，人为制造差错，以便观察故障发生后的现象，故障注入模块能产生多个故障激励，传递到硬件板卡相应模块，影响系统正常运行。

（2）模块接口框图

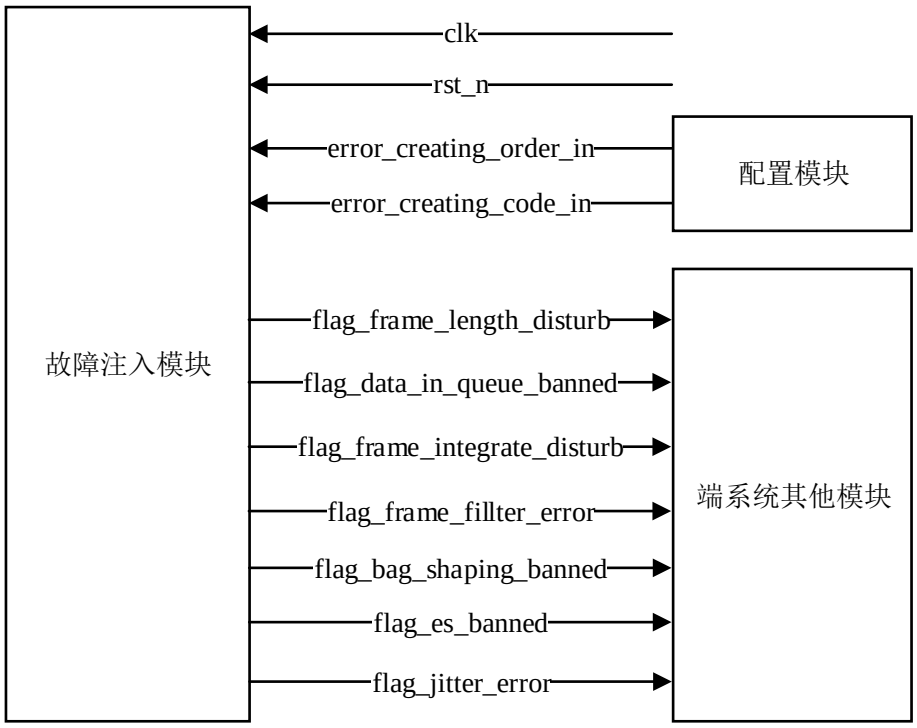

图4.24 故障注入模块接口

（3）模块详细设计

目前已设置七种类型的故障，注入故障时，首先要使能故障注入标识，并且选择注入故障的类型，故障指令可以通过简单网络管理协议（Simple Network Management Protocol，SNMP[56]）方式传输至多个设备，操作便捷。设备检测到故障指令后，判别故障类型，扰乱相应模块正常运行。

入队帧长信息错误和数据入队干扰与发送侧队列管理模块相关联，当入队帧长信息错误使能有效时，数据入队后，将给队列管理模块提供错误的帧长信息，而数据出

队之前要读取帧长信息，以判断搬移次数，此故障的作用为扰乱数据正常出队。

当数据入队干扰使能有效时，原本应当入队的数据没有入队，但与数据帧有关的信息都已正常记录，造成结果为帧信息个数和实际存储的帧数目不对应，扰乱数据正常出队。

帧完整性缺失与发送侧发送冗余模块相关联，正常情况下同一虚拟链路号下的 RC 数据帧帧尾序列号连续递增，注入帧完整性缺失故障后，序列号将不再连续，直接影响接收端完整性检查模块。

RC 整形失效与时间抖动异常也都是在发送侧注入故障，和业务调度模块相关联，当 RC 整形失效使能有效时，同一虚拟链路号的 RC 数据帧帧间隙不再满足 BAG 的要求，直接影响接收端 RC 警管模块。当时间抖动异常使能有效时，情况类似，也将影响 RC 警管模块。

帧过滤异常与接收侧帧信息提取模块相关联，原先 MAC 地址符合要求的数据应当传输至下一模块，当帧过滤异常使能有效时，会把正常帧当成异常帧处理，通过存储控制模块丢弃该帧。另外，端系统失效表示整个端系统下线，即网口不可发送数据。

4.4 配置及监控模块

（1）模块功能概述

端系统内部可设定固定的运行参数及表项，但由于不同用户的需求不同，为扩大端系统应用范围，增设配置模块，可随时更改表项内容及运行参数，以适应复杂可变的拓扑环境。

配置完成后，端系统开始处理数据，此时相应模块开始统计数据信息，给到监控模块，供上层读取状态。监控的作用是既可以便于设计者调试，同时也便于使用者更直观地观测端系统的运行状态。

（2）模块接口框图

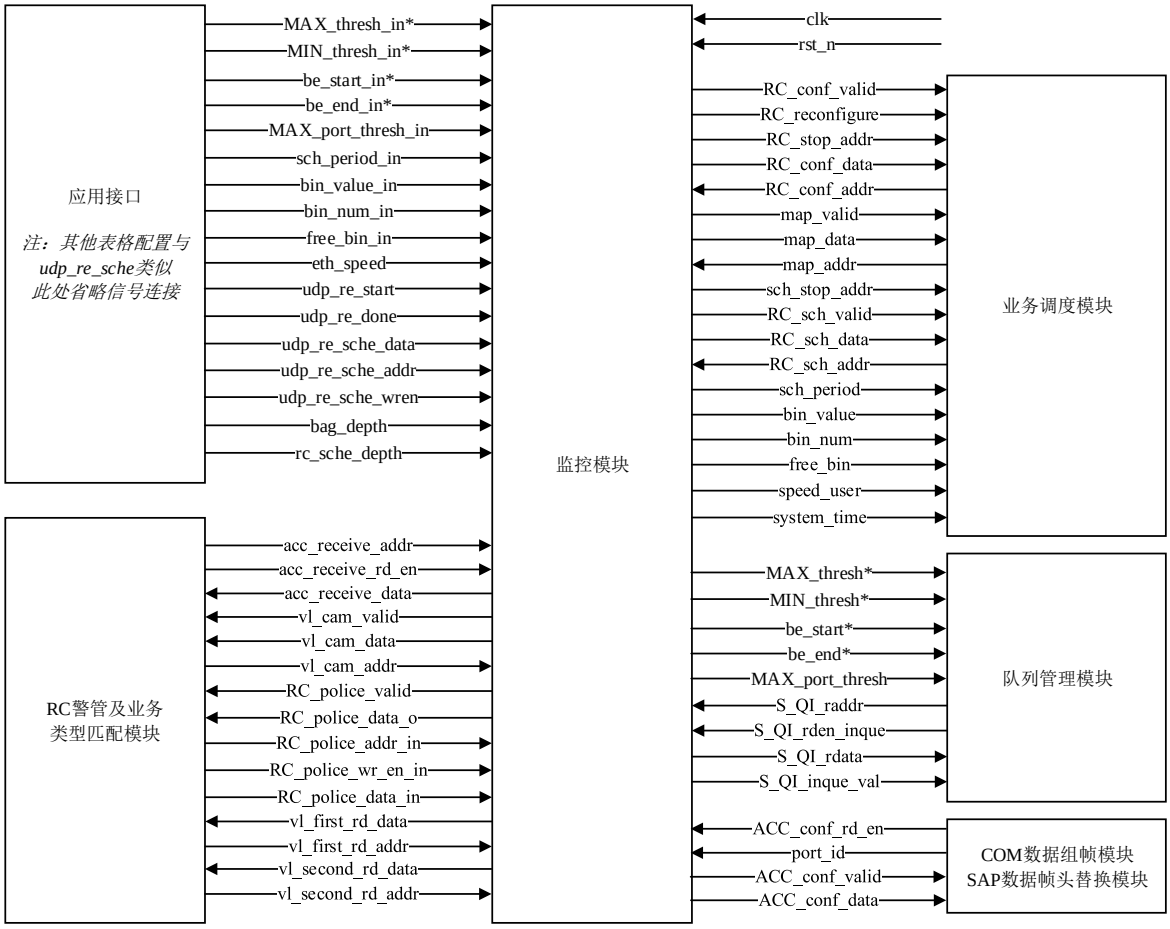

图4.25 配置及监控模块接口

（3）模块详细设计

根据实际规划的业务，用户提供表格文件，应用经驱动向配置模块发起配置指令，端系统识别到配置指令后，开始向随机存取存储器（Ramdom Access Memory，RAM）中写入表项内容，每个表项都有固定的寄存器，从接收到配置起始信号，直至配置完成信号到来，一个表格才算配置结束，例如硬件协处理器加速表、业务类型匹配表等，接口的配置过程大致相似，值得注意的是业务调度表和 BAG 表还需要上层操作系统告知端系统表项深度为多少，以便于相关模块顺序读取表项时能及时停止。

另外如队列管理模块、业务调度模块，运行前也要事先配置一些参数，像给每个队列 RC 数据帧划分的最大存储空间、调度片的长度等，这些参数在收到配置指令后直接采用寄存器寄存，接口也可以反读寄存器以确定内部工作参数。端系统在运行之前，所有配置必须完成，否则会影响端系统的正常运行。

内部配置结束之后，端系统可以开始正常运行，运行过程中，及时记录发送侧与接收侧的状态，比如 RC 数据帧发帧数目、收帧数目等，通过对这些状态的观察，能

直接了解运行过程中是否有数据帧丢失，既方便验证端系统性能，也方便知晓系统中的节点有无正常工作。

与端系统内部运行参数的读取过程不同，上层操作系统会周期性地读取状态寄存器，将统计结果在终端应用界面上直观表现，无需繁琐操作，可以实时清晰地观测结果。

4.5 接口模块

本节从两方面介绍接口模块，分别是与上层应用的接口及与网口侧的接口，端系统对上采用 EMIF 接口，实现过程较为简单，故而介绍时不做过分说明，对下采用 SGMII 接口，实现逻辑为 MAC 逻辑输出千兆介质无关接口（Gigabit Media Independent Interface，GMII）信号，经 SGMII IP 核变换成高速串行数据流，SGMII IP 核包含的部分关键技术和 IP 核构建已在第二章有所叙述，本章不再赘述，仅着重说明 MAC 部分及 PHY 芯片的配置。

4.5.1 EMIF 总线接口模块

（1）模块功能概述

通过 24 位寄存器空间的合理分配，EMIF 接口既参与数据传输工作也能完成参数配置和状态监控。本模块设有两个子模块：接口基础模块、插入捕获模块。前者作为数据的收发、配置及监控通路；后者用于完成用户需求的其他应用。

（2）模块接口框图

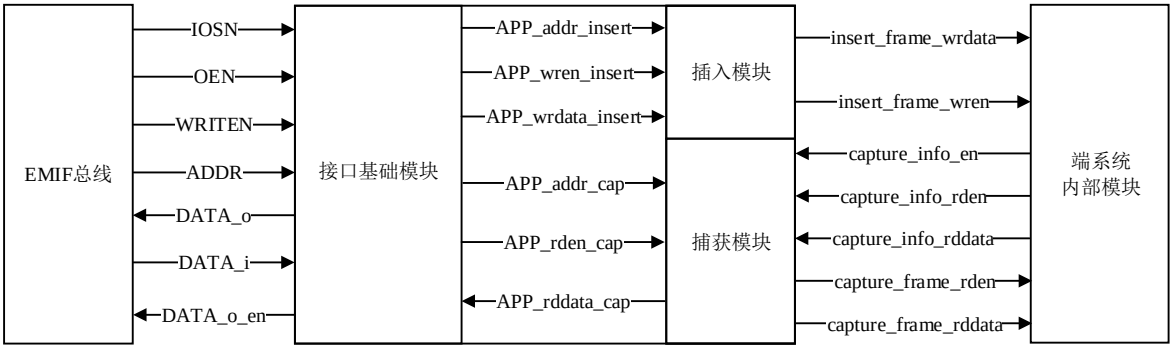

图4.26 EMIF 总线接口

（3）模块详细设计

接口在进行写操作时第一步为片选解析，IOSN 为低时检测 OEN 或 WRITEN 信号是否拉低，若 WRITEN 信号拉低则说明此次操作为写操作，寄存 WRITEN 信号拉低时的 ADDR 和 DATA_i，即此次写操作的地址和数据。完成操作后片选 IOSN 置高，释放片选。

接口在进行读操作时第一步依然是片选解析，IOSN 为低时检测 OEN 或 WRITEN 信号是否拉低，若 OEN 信号拉低则说明此次操作为读操作，寄存 OEN 信号拉低时的 ADDR 和 DATA_o，两拍之后将数据放入数据总线作为此次操作的读数据。完成操作后片选 IOSN 置高，释放片选。

插入模块的数据处理流程如下：驱动有数据需要搬移时先往数据量寄存器里写入此次搬移的数据量，接着驱动往数据寄存器里写入 16 位数据，最后在驱动写数据完成之后往写数据结束寄存器里写全 0，表示此次操作结束。

捕获模块的数据处理流程如下：上层定期轮询端系统接收缓存是否有帧需要搬移，即读取数据量寄存器，若读回的值为 0，表明无数据可搬移，继续轮询，若读回的值不为 0，则硬件有数据需要搬移，记录需要搬移的数据量，接着往数据量寄存器中写入此次读回的数据量，告知硬件开始读数据。在发送完读数据开始标志之后，开始搬移数据，根据搬移的数据量计算需要搬移的次数，从数据寄存器中读取数据，最后在读取完成之后往读取结束寄存器中写入全 0，告诉硬件本次搬移完成，驱动继续进入轮询状态。

4.5.2 MAC 核及 PHY 配置模块

（1）模块功能概述

MAC 核负责 SGMII IP 核与端系统用户逻辑的数据交流及处理，端系统发送侧根据网口速率为千兆、百兆还是十兆，将 LocalLink（即有帧起始、帧结束等标识）格式的数据经 MAC 核转化为速率合适的 GMII 格式的数据，端系统接收侧进行逆向操作。同时 MAC 核内部还负责配置 PHY 芯片的速率。

（2）模块接口框图

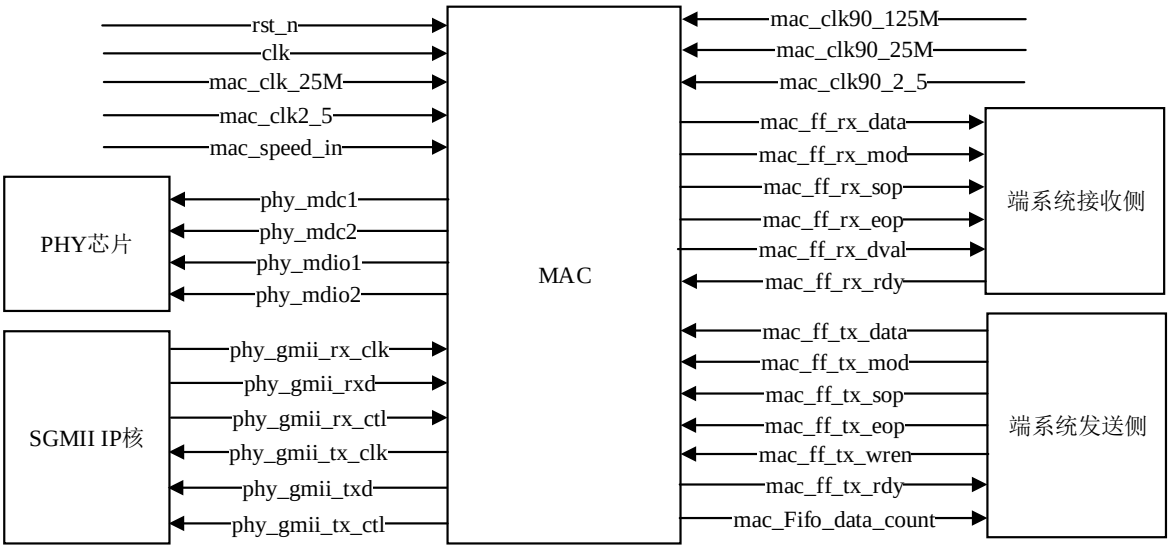

图4.27 MAC 核及 PHY 配置模块接口

（3）模块详细设计

端系统发送数据时先把 LocalLink 形式的数据递交给 MAC，将用户侧内部使用的 32 bit 位宽数据转化为 8 bit，由于用户侧时钟和网口侧速率可能不同，所以在 MAC 中需进行跨时钟处理，处理结束后再经过 GMII 模块，将 8 bit 数据转化为上下边沿皆有效的数据，而 SGMII IP 核也会把并行数据转化为串行数据流传给 PHY 芯片。

接收端处理流程类似，SGMII IP 核恢复时钟，将串行数据流转化为 GMII 格式的数据后，需要根据当前网口速率确认上下边沿哪些数据有效，转化为单边沿数据后，把 8 bit 数据通往 MAC 逻辑中进行跨时钟处理和位宽转换，最后传输至接收侧用户逻辑。

端系统 MAC 需要完成的功能并不复杂，例如位宽转换、跨时钟域和数据格式转换都是较为简单的模块，但数据传输过程中也要及时与发送侧、接收侧交互，避免因数据处理速度和数据流量不匹配导致数据传输出错。

另外在此简单介绍 PHY 芯片相关配置过程，一般情况下，PHY 芯片的默认速率为千兆，如需改变网口工作速率，必须配置 PHY 芯片。PHY 芯片中有多个寄存器，通过芯片手册可知晓与网口速率相关的寄存器和配置数值。PHY 芯片的配置数据以串行方式输入，且时钟频率很低。配置、读取操作需遵循一定规则，如图 4.28 所示：

32 bit 前导码	帧起始符	帧操作符	PHY芯片设备地址	PHY芯片寄存器地址
11111111	01	10/01	00000	00000
转换符	数据域			初始状态
z0/10	0001001100000000（示例）			11111111

图4.28 PHY 配置帧格式

PHY 芯片配置模块发送串行配置数据需按照数据组成格式，首先发送 32 bit 前导码，接着发送 2 bit 的帧起始符，以通知 PHY 芯片将开始被内部逻辑配置，而后是 2 bit 的帧操作符，“10”表明此次配置为读操作，“01”则为写操作。当板卡有多个 PHY 芯片时，用 5 bit 的 PHY 芯片设备地址区分，而具体哪种功能的开与关则与 5 bit 的 PHY 芯片寄存器地址有关，2 bit 转换符中“z0”代表读操作，“10”代表写操作。16 bit 的数据域是配置内容或读取内容。

写操作流程如下：帧操作符为“01”，转换符为“10”，之后与配置时钟同步，以串行数据流的形式传递给 PHY 芯片即可。为查询 PHY 芯片当前工作状态，也可以读取内部寄存器，读操作流程如下：帧操作符为“10”，转换符为“z0”，传递给 PHY 芯

片，PHY 芯片提取寄存器数据传回至硬件逻辑。

4.6 本章小结

基于本文第三章提出的总体设计方案，本章从发送侧模块、接收侧模块、故障注入模块、配置与监控模块以及接口模块这五个部分，从模块功能、详细设计、接口信号、状态机等方面给出了新型 AFDX 端系统的详细设计，为保证本文完整性的同时又要突出创新点，故而与第一代端系统相似的部分仅做简略介绍，而关于例如 COM 数据及 SAP 数据处理、配置与监控功能等部分则有更详尽的说明，各个模块采用 Verilog 语言实现。

第五章 仿真验证与板级验证

本章将验证测试第四章设计的新型 AFDX 端系统模块功能是否符合项目需求。首先介绍仿真工具及仿真环境，接着侧重于本文创新点，从 COM 业务数据处理、SAP 业务数据处理、故障注入和配置及监控四部分，利用仿真平台，验证理想情况下模块功能的正确性。最后采用主机、网线以及项目合作方要求的 FPGA 板卡等搭建板级测试环境，验证真实环境下新型 AFDX 端系统的性能是否达到设计目的。

5.1 仿真验证

5.1.1 仿真工具说明

本文描述的新型 AFDX 端系统已在 Actel 公司的 Libero v12.6 及 Xilinx 公司的 Vivado 2019.2[57][58]软件上设计完成。两者总体设计流程大致相同，区别在于设计实现时两家公司的 IP 核不同。仿真时两种软件均可调用外部软件，本文采用的仿真软件为 Mentor 公司的 QuestaSim-10.6c。相比于 Vivado 2019.2，Libero v12.6 软件使用人数较少，且使用时仿真编译、抓取信号、烧录比特流等皆需调用其他公司的软件，操作麻烦，为提升开发者后续上板验证的便捷性以及使用者的易操作性，本文将结合 Xilinx 平台及 QuestaSim 仿真工具验证新型 AFDX 端系统是否满足设计需求。

5.1.2 仿真环境说明

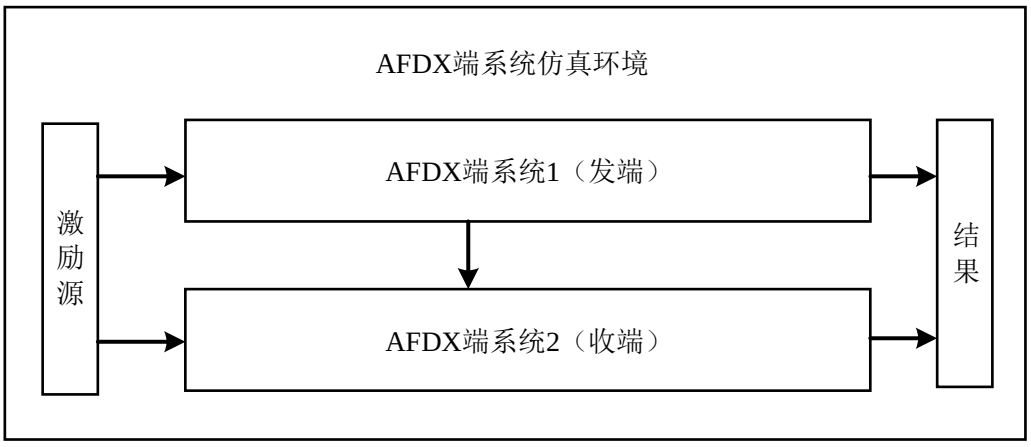

图5.1 AFDX 端系统仿真环境

如图 5.1 所示，仿真环境由四部分组成，即激励源、AFDX 端系统 1、AFDX 端系统 2 及结果组成。其中激励源模拟真实环境中的上位机，负责产生验证端系统功能所需的数据激励或者信号激励，激励源可产生 COM 业务数据、SAP 业务数据、

MAC_RAW（BE）业务数据，也可以产生多种故障激励。数据激励递交给接口模块，信号激励则直接作用于内部相应模块。AFDX 端系统 1 及 AFDX 端系统 2 用以模拟真实环境下的收发设备，两者皆与激励源相连，因为故障注入模块激励既可能影响发端也可能影响收端，为简便起见，数据激励均输入至 AFDX 端系统 1，经发送侧处理，过 MAC 后传输至 AFDX 端系统 2。通过调整激励源，利用仿真平台观察收发端的结果，以验证端系统功能是否达到设计需求。

5.1.3 COM 业务数据处理仿真验证

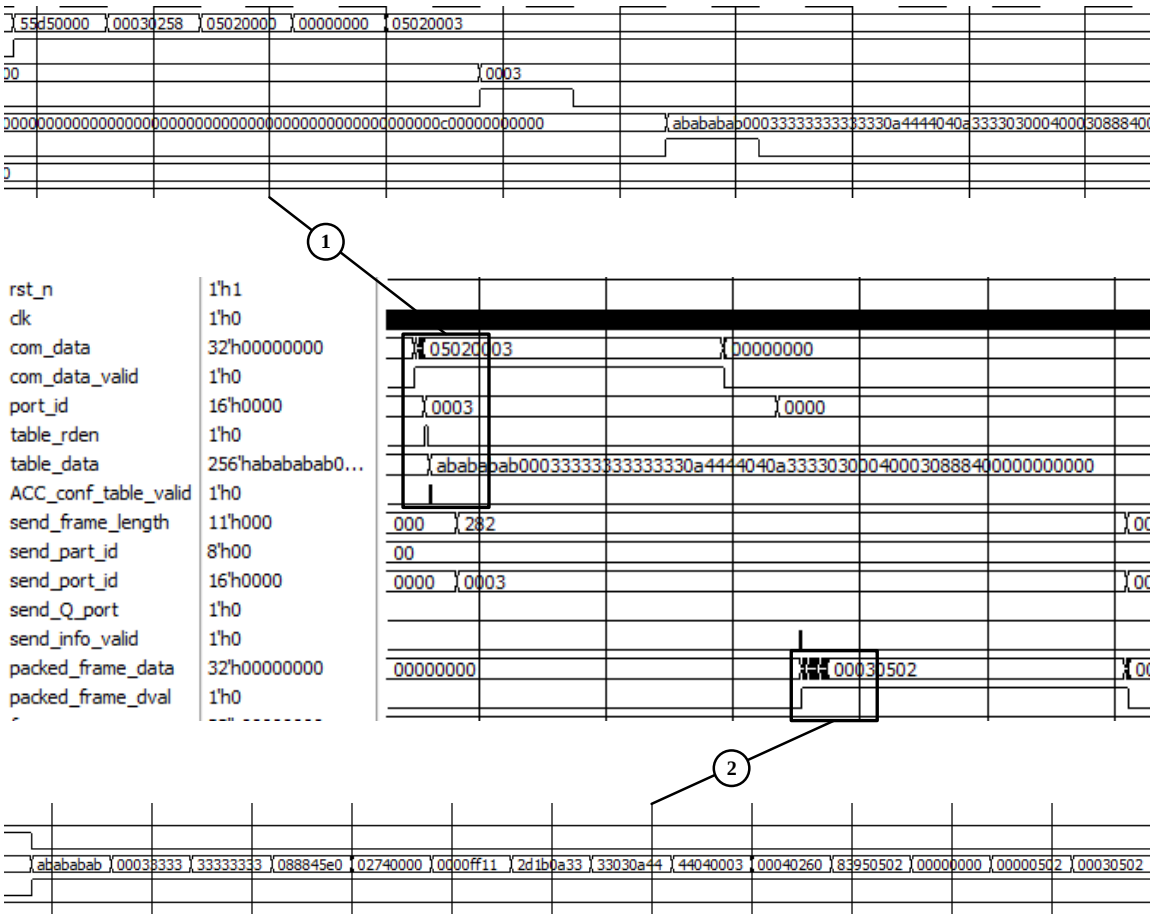

图5.2 COM 业务发端数据处理仿真波形图

数据经数据分类模块后，将 COM 数据传递至发送侧 COM 数据组帧模块，如图 5.2 所示，编号“1”为模块数据输入，编号“2”为模块数据输出。波形图以 UDP_COM 类型数据为例，com_data（输入 COM 业务数据）前 8 字节“0x55d50000 00030258”为数据帧信息，仅参与数据处理，在组帧完成后应该被丢弃，除数据帧信息外，仅有裸数据。数据输入后，用 port_id（端口号）查询表格，得到 table_data（硬件协处理器加速表表项内容），从中获知 MAC 地址、IP 地址、UDP 地址及业务类型等内容。IP 头和 UDP 头检验和计算完成后，开始组帧，由 packed_frame_data（已组帧 COM

数据）可见原始 COM 数据帧前的 8 字节数据信息已被去除，send_frame_length（已组帧 COM 数据帧长）、send_port_id（已组帧 COM 数据端口号）等信号寄存帧信息并传递给下级模块，而 COM 数据在裸数据前已添加好从表项中获知的 MAC 地址、IP 地址等内容。

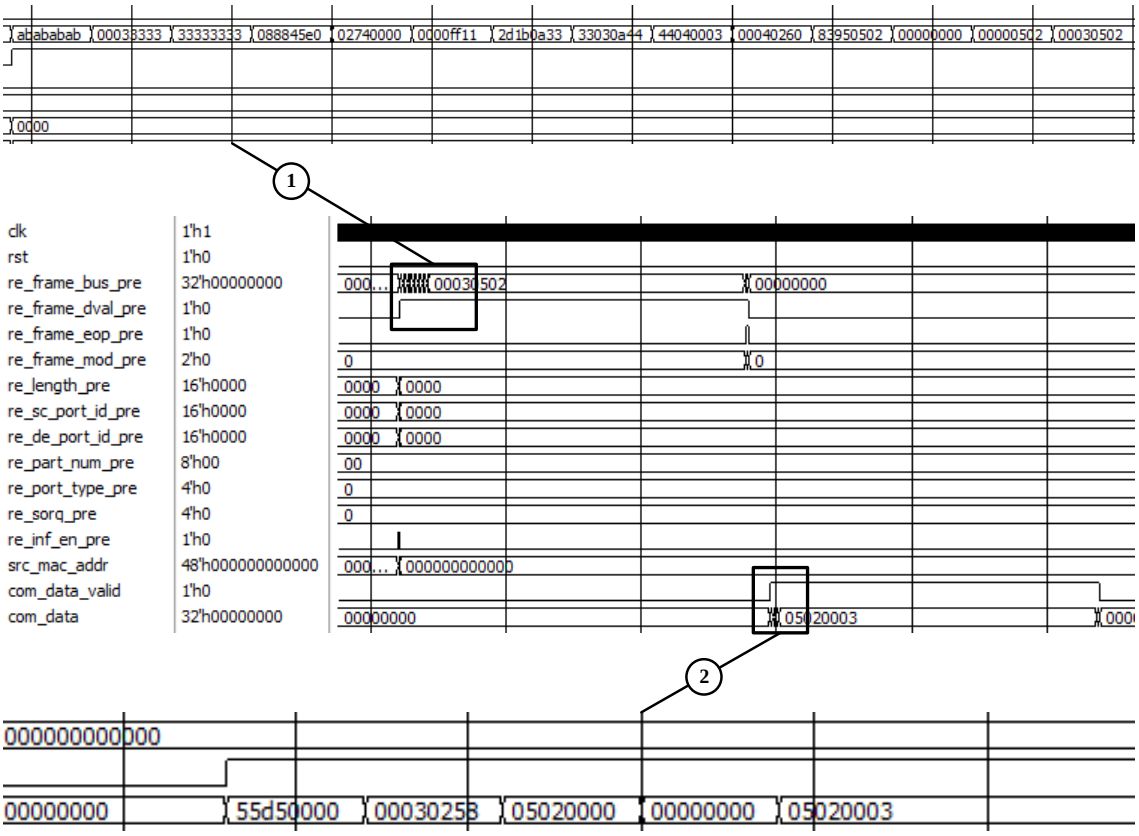

图5.3 COM 业务收端数据处理仿真波形图

组帧完成的数据经其他模块，最后会传输至接收侧的 COM 数据拆帧及 SAP 数据帧头还原模块。如图 5.3 所示，编号“1”为模块数据输入，编号“2”为模块数据输出。与发送侧对应，re_frame_bus_pre（输入 UDP_COM 业务数据）为携带 MAC 头、UDP 头、IP 头等信息的完整数据帧，同时上级模块也传递 re_length_pre（输入 UDP_COM 业务数据帧长）、re_port_type（输入数据类型）等信息，COM 数据拆帧及 SAP 数据帧头还原模块接收到数据及数据信息后并不会马上丢弃帧头，而是比较数据帧的 IP 检验和以及 UDP 检验和是否与数据帧帧头的相同，故而从数据输入到数据输出耗费了一些时间。由 com_data（已拆帧 COM 数据）可见数据帧的 MAC 头、IP 头、UDP 头等信息已去除，同时帧头添加 8 字节的数据帧信息，对比发送侧的输入输出数据，两者相同，表明数据从发端到收端数据处理正确，另外 UDP_COM 数据的正确处理流程为发端为裸数据增加 MAC 头、IP 头和 UDP 头，收端去除数据帧的 MAC 头、IP 头和 UDP 头，模块功能正确。

5.1.4 SAP 业务数据处理仿真验证

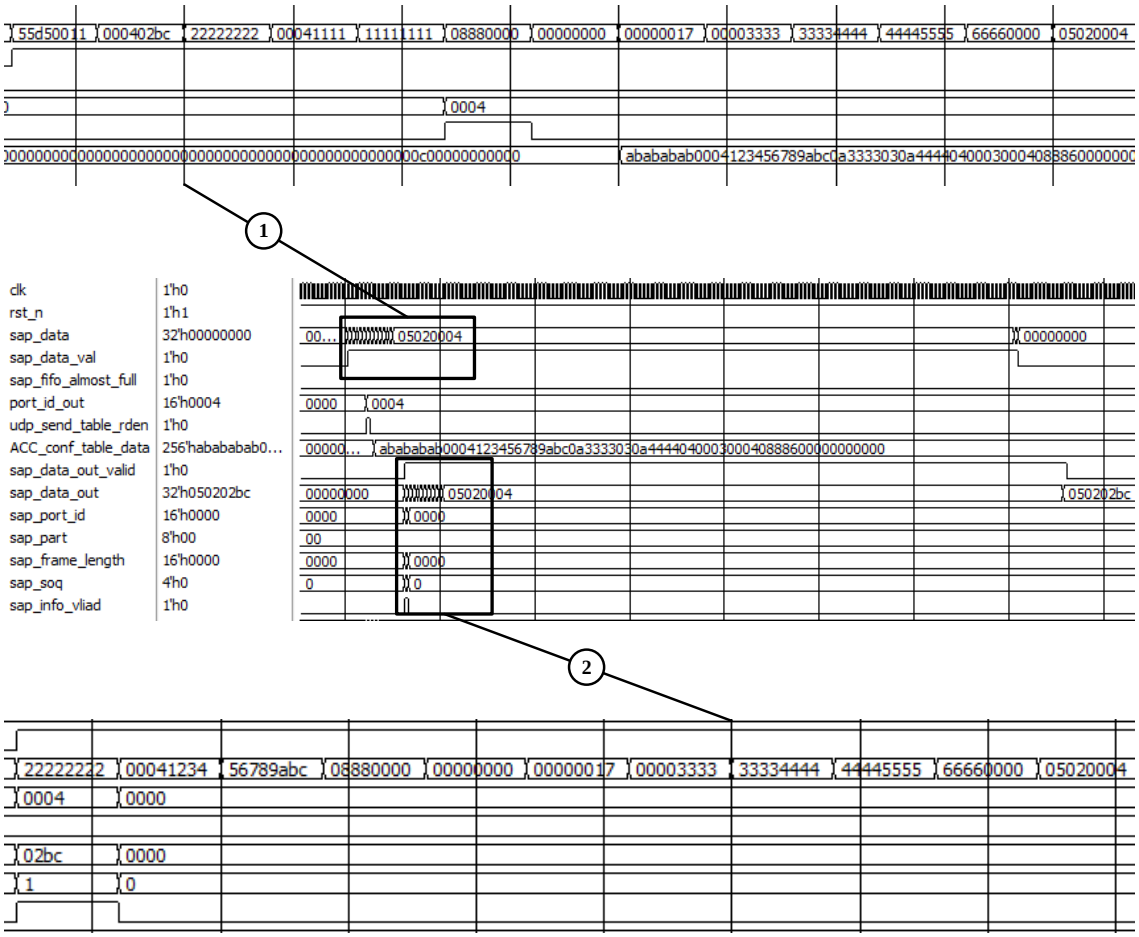

图5.4 SAP 业务发端数据处理仿真波形图

数据经数据分类模块后，对于不是 COM 类型的数据，将会传递至发送侧 SAP 数据帧头替换模块。如图 5.4 所示，编号“1”为模块数据输入，编号“2”为模块数据输出。波形图以 MAC_SAP 类型数据为例，sap_data（输入非 COM 业务数据）前 8 字节“0x55d50011 000402bc”为数据帧信息，仅参与数据处理，在组帧完成后需要从数据帧中删除，除数据帧信息外，MAC_SAP 业务类型数据已含有 MAC 头。模块接收到数据后，利用 port_id（端口号）查询表格，得到 ACC_conf_table_data（硬件协处理器加速表表项内容），从中获知 MAC 地址及业务类型等内容，SAP 类型的数据只需进行源地址的替换，处理前数据帧的 MAC 源地址为“0x1111 11111111”，经替换后，由 sap_data_out（已替换帧头 SAP 数据）可见原始 SAP 数据帧前的 8 字节数据信息已被去除，源地址被替换为表项内容中的“0x1234 56789abc”。sap_frame_length（已替换帧头 SAP 数据帧长）、sap_port_id（已替换帧头 SAP 数据端口号）等信号寄存帧信息并传递给下级模块。

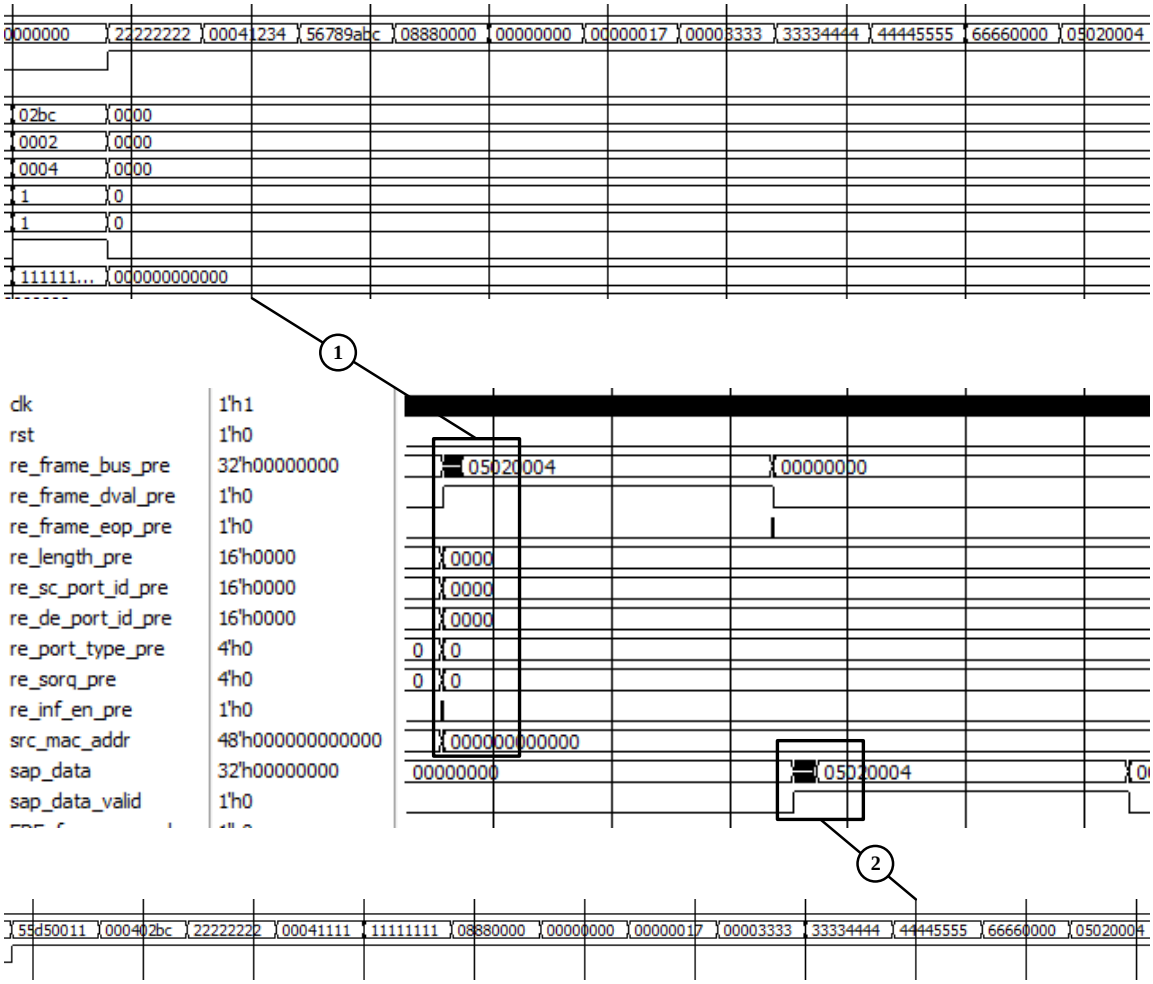

图5.5 SAP 业务收端数据处理仿真波形图

与 COM 数据处理流程相同，帧头替换完成的 SAP 数据经其他模块，最后也会传输至接收侧的 COM 数据拆帧及 SAP 数据帧头还原模块。如图 5.5 所示，编号“1”为模块数据输入，编号“2”为模块数据输出。re_frame_bus_pre（输入 MAC_SAP 业务数据）为已替换 MAC 源地址的完整数据帧，并且上级模块也把 re_length_pre（输入 MAC_SAP 业务数据帧长）、re_port_type（输入数据类型）、src_mac_addr（发端源 MAC 地址）等信息传输至此模块，COM 数据拆帧及 SAP 数据帧头还原模块接收到 SAP 数据及数据信息后，首先还原源 MAC 地址，之后耗费一些时间，比较数据帧的 IP 检验和以及 UDP 检验和是否与帧头的相同。由 sap_data（已还原帧头 SAP 数据）可见数据帧的 MAC 源地址已由“0x1234 56789abc”还原为“0x1111 11111111”，另外帧头添加 8 字节的数据帧信息，对比发送侧的输入输出数据，两者相同，表明数据从发端到收端处理正确，MAC_SAP 数据的正确处理流程为发端替换 MAC 头源地址，收端还原 MAC 头源地址，模块功能正确。

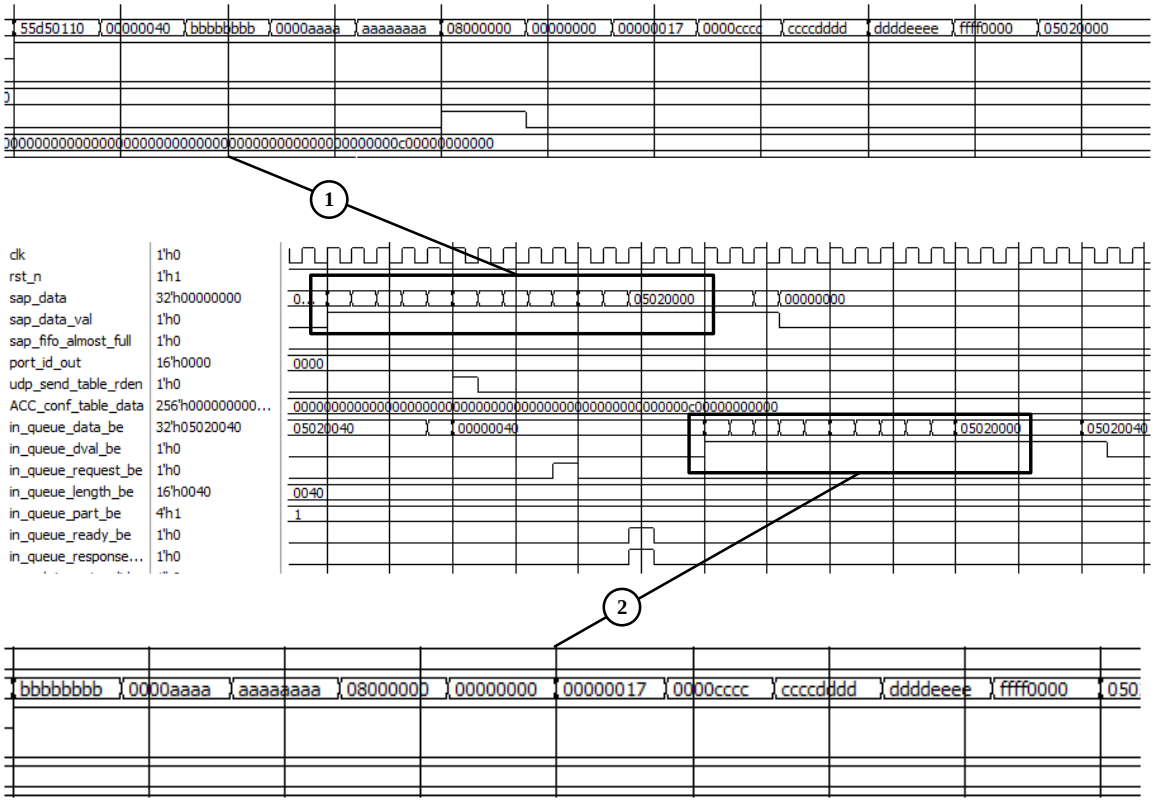

图5.6 MAC_RAW 业务发端数据处理仿真波形图

SAP 数据帧头替换模块除处理 SAP 数据，也会处理 MAC_RAW（一般为 BE 数据帧）类型的数据。如图 5.6 所示，编号“1”为模块数据输入，编号“2”为模块数据输出。sap_data（此处为 BE 数据）前 8 字节“0x55d50110 00000040”为数据帧信息，用于数据处理，传递给下一级模块时需要从数据帧中删除，从 in_queue_data_be（传递给下一级的 BE 数据）信号波形中可见前 8 字节的帧信息已经去除。对于 MAC_RAW 类型的数据帧，在 SAP 数据帧头替换模块也需要查表以确认类型。另外，数据帧还和队列管理模块有交互，当接收到数据帧后，先发送 in_queue_request_be（BE 数据入队申请）以及 in_queue_length_be（BE 数据帧长）等信息，队列管理有空间供完整帧存储则回传 in_queue_response_be（BE 数据入队反馈）和 in_queue_ready_be（BE 数据允许入队）信号，接着开始传输数据。

MAC_RAW 类型数据在接收端不会经过 COM 数据拆帧及 SAP 数据帧头还原模块，而是在业务类型匹配模块查询到业务类型后，协同其他信息传递至存储控制模块，即 length_pre（数据帧长）、eth_type_pre（帧类型属于 BE 还是 RC）等信号，存储控制模块从缓存中取出数据，直接递交给接收缓存供上层应用读取。如图 5.7 所示， 编号“1”为模块数据输入，编号“2”为模块数据输出。ff_rx_data_A 与 ff_tx_data_BE 相同，即模块输入输出数据完全相同，且与发送侧数据对应，表明模块功能正常。另外，与 COM 数据和 SAP 数据有别，MAC_RAW 类型数据的 8 字节帧头在数据帧进

入接收缓存后才加。

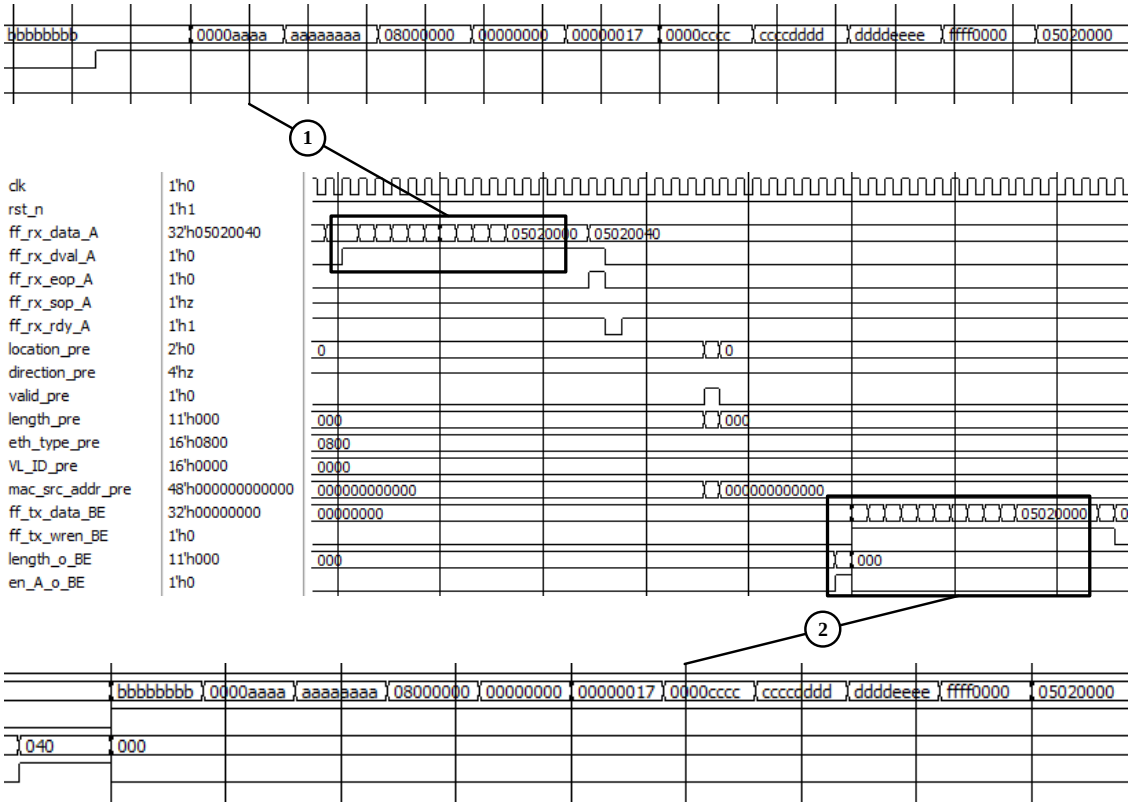

图5.7 MAC_RAW 业务收端数据处理仿真波形图

5.1.5 故障注入模块仿真验证

故障的模式有很多，以帧完整性缺失为示例。图 5.8 为故障注入之前的波形图，图 5.9 为故障注入之后的波形图。故障注入之前，error_creating_order（故障注入使能）和 error_creating_order（故障注入类型）均为“0x0”，接收侧从 MAC 接收数据，编号“1”的矩形框中可见 SN_r（数据帧序列号）从“0x00”开始递增，符合完整性要求，ff_rx_data（存储控制接收的数据帧）成功通过完整性检查模块，编号“2”的矩形框中 ff_tx_data_RC（递交给下级的数据）传输的数据量和 ff_rx_data 相同。

故障注入之后，图 5.9 编号为“3”的矩形框中可见 error_creating_order 置为“0x1”，error_creating_order 数值为“0x04”，表示注入故障为帧完整性缺失。编号为“4”的矩形框中可见 SN_r 递增到“0x0e”后变为“0x01”，完整性检查时，如果序列号恒不变，代表数据帧出错，相应模块向存储控制模块递交丢弃信号，编号为“5”的矩形框中可见序列号出错的数据帧到来时 discard_A（A 网络数据帧丢弃指令）和 discard_B（B 网络数据帧丢弃指令）拉高，存储控制模块不再向下级模块递交数据，ff_tx_data_RC 无有效数据。

需要说明的是帧完整性缺失在发送端注入，所以图中显示注入故障后会延时一段

时间，才会在接收端出现结果。

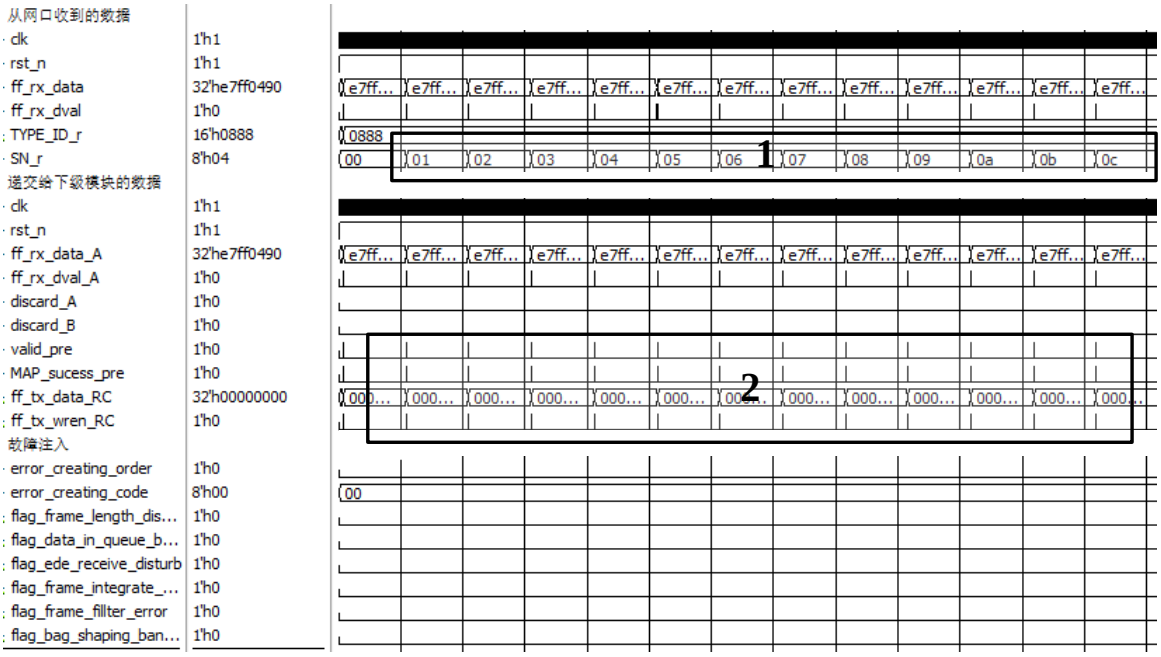

图5.8 故障注入前的仿真波形图

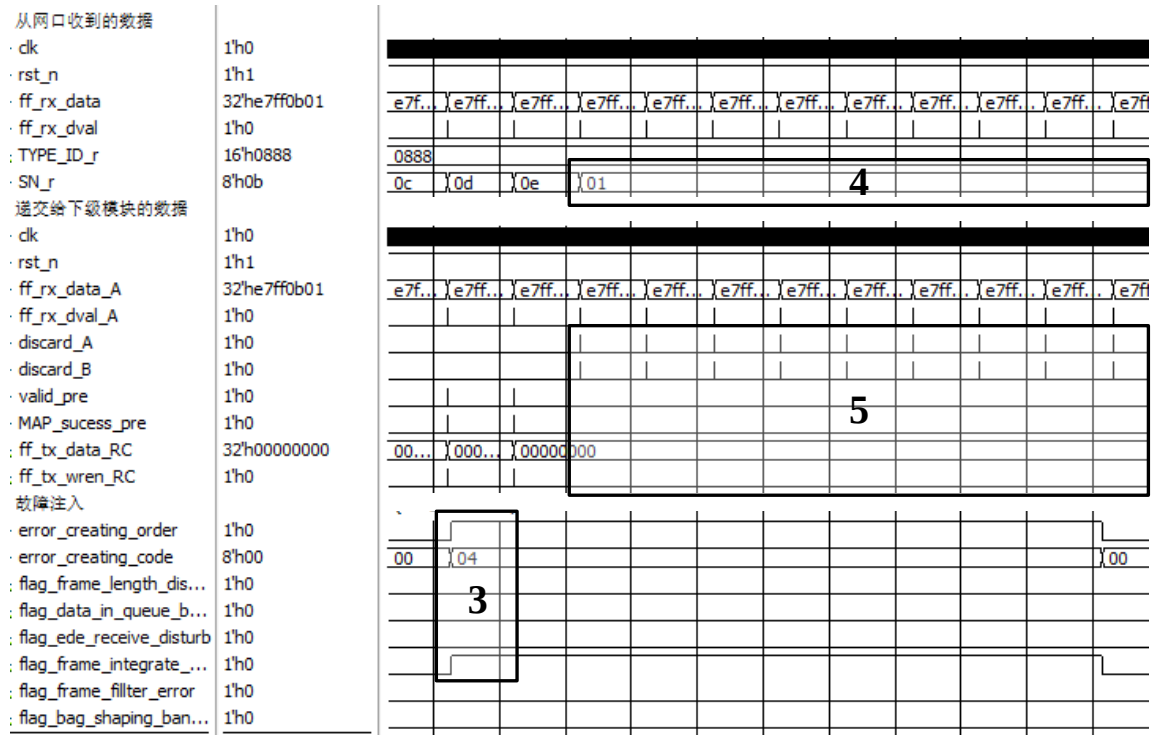

图5.9 故障注入后的仿真波形图

5.1.6 配置及监控模块仿真验证

激励源发送端口号为“0x2”、“0x3”、“0x4”的 RC 数据帧，图 5.10 为一些监控参数，编号为“1”的矩形框中为各端口号 RC 数据帧的发送数目，编号为“2”的矩

形框中为各端口号 RC 数据帧的接收数目，由于没有发送端口号为“0x1”的 RC 数据帧，故而可见波形图中 RC_FRAME_SENDNUM_1（端口号为 1 的 RC 数据帧发送数目）以及 RC_FRAME_RCVNUM_1（端口号为 1 的 RC 数据帧接收数目）无有效数据，而 RC_FRAME_SENDNUM_2 至 RC_FRAME_SENDNUM_4 数值一直在递增，并且和 RC_FRAME_RCVNUM_2 至 RC_FRAME_RCVNUM_4 数值相等，显而易见，监控功能的存在可以更好观测端系统的运行状态。另外，RC_SENDRATE（RC 数据帧发送速率）与 RC_RCVRATE（RC 数据帧接收速率）数值没有递增是因为每累积一秒才计算一次，但 RC_SENDRATE_pre（RC 数据帧发送速率累计）与 RC_RCVRATE_pre（RC 数据帧接收速率累计）在有相应端口 RC 数据帧发送后数值增加。

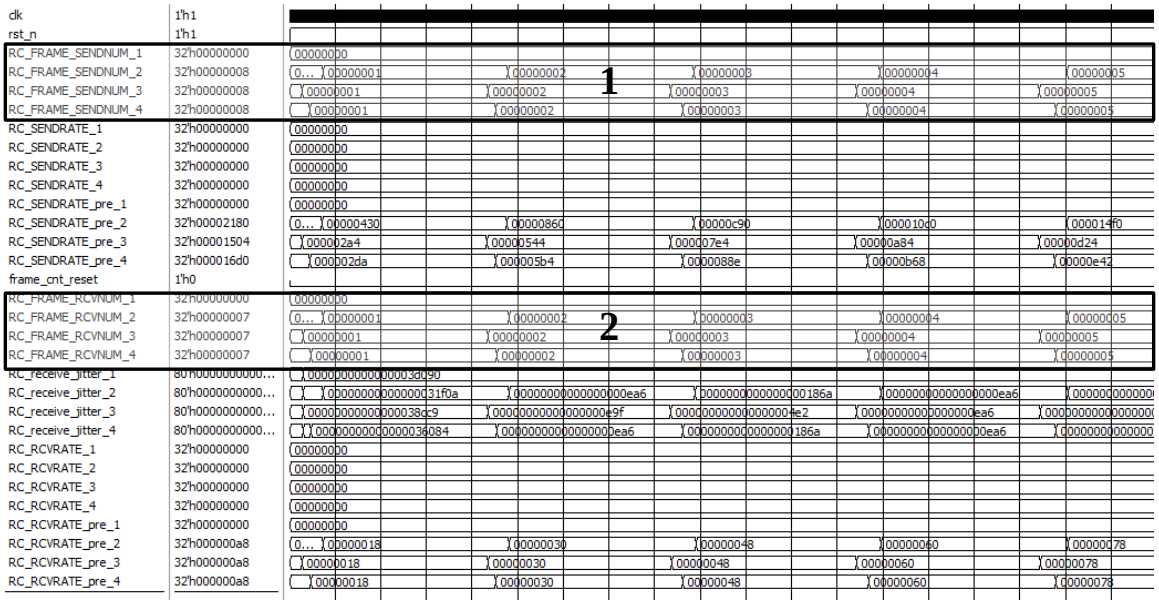

图5.10 监控模块仿真波形图

配置模块以速率配置为例，如图所示，Speed_in（速率配置）为“0x4”时，代表端系统网口工作在千兆模式下，编号“1”可见数据经 MAC 处理后的 GMII 数据符合千兆要求，发送数据较快。数据保持发送，更改网口工作速率，Speed_in 为“0x2”代表网口工作在百兆模式下，编号“2”可见数据经 MAC 跨时钟域处理后的 RGMII 数据符合百兆要求，相同数据发送耗费的时间比千兆情况下更长，配置功能正常。配置模块提升了系统的适用度。

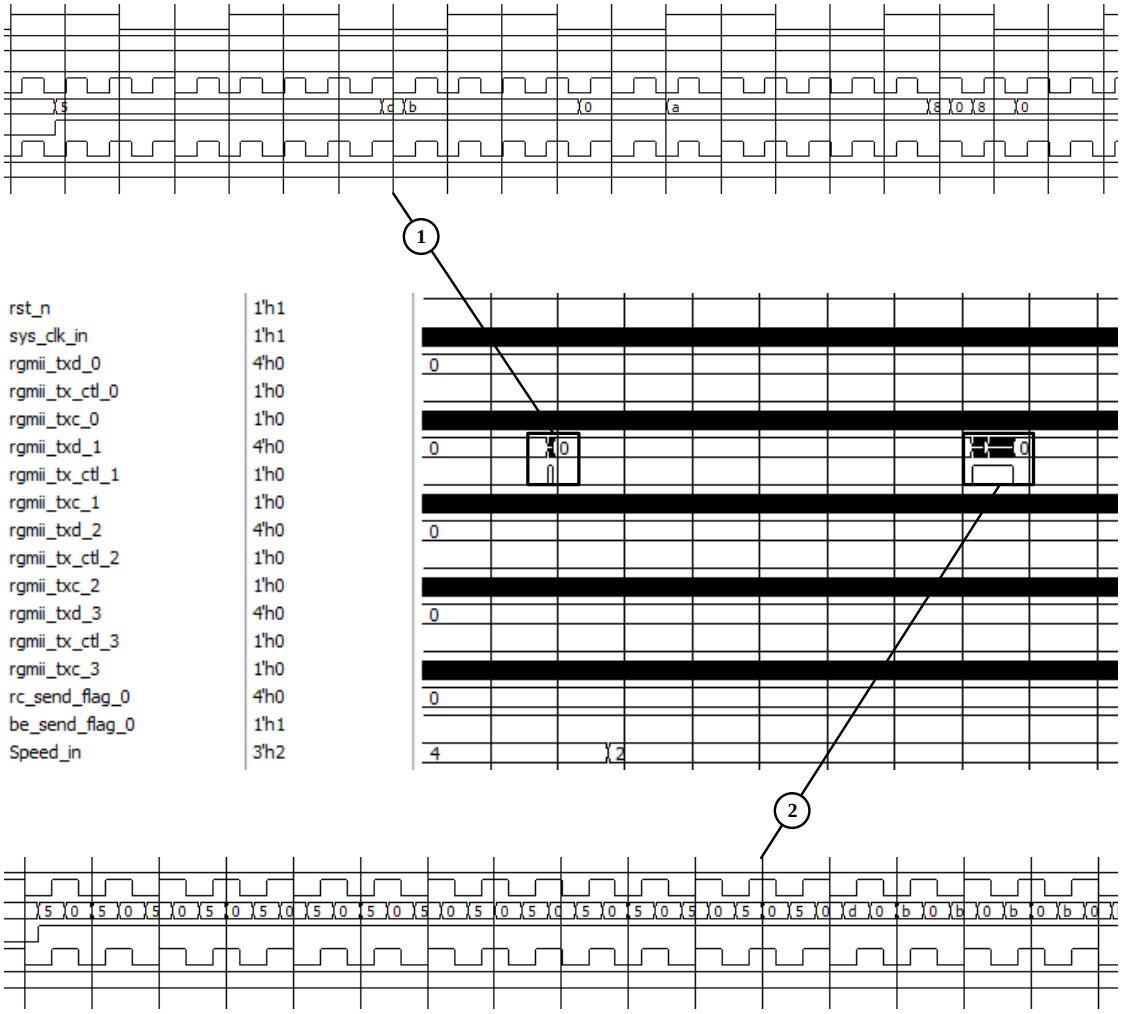

图5.11 配置模块仿真波形图

5.1.7 仿真验证小结

仿真中以 UDP_COM、MAC_SAP 和 MAC_RAW 数据为典例，从收发两侧分别验证了 COM 业务数据处理、SAP 业务数据处理功能的正确性。而在故障注入的验证中则以完整性缺失为典例，对比故障注入前后的波形图，结果与预期一致。监控与配置的仿真验证则分别以收发帧数和网口速率为例，同样表明模块功能符合预期。通过观测仿真中的波形信号变化，完善代码设计，达到预期设计目的，为板级验证作铺垫。

5.2 板级验证

5.2.1 板级环境说明

本文板级验证采用的软件平台和硬件板卡分别为 Vivado 2019.2 和实验室自研板卡，AFDX 端系统为 AFDX 系统中的一部分，为尽量真实地还原使用场景，故而搭建板级验证环境时，配合使用实验室设计开发的 AFDX 交换机[59]和上层应用软件[60]。

如图 5.12 所示，板级验证环境包括以下组成部分：终端主机五台、FPGA 硬件板卡四个、AFDX 交换机两台。其中四个 FPGA 硬件板卡上分别烧录 AFDX 端系统 1 至 AFDX 端系统 4 的比特流文件，与四台终端主机相连，四台终端主机上运行 AFDX 收发包软件和故障现象显示软件。剩余一台终端主机作为外网设备与任一主机相连，负责监控及配置 AFDX 系统。四个 AFDX 端系统与两个交换机相连，构成冗余网络。

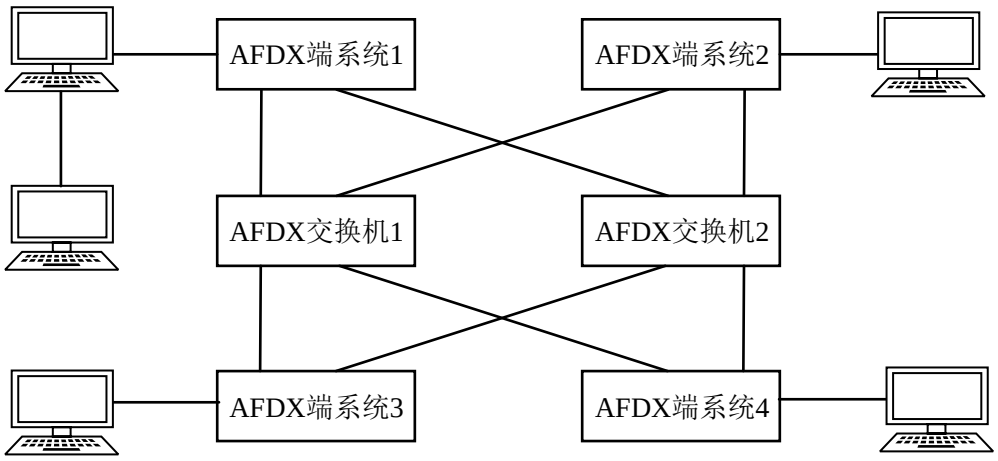

图5.12 板级验证环境拓扑

根据图 5.12 拓扑图连接好实物图，在开始收发数据之前，必须要配置好操作系统端口与端系统板卡之间的映射表，端口映射表分为 IO 发送表和 IO 接收表，用于指示端口数据帧的类型、下发数据帧的端口号、数据类型等信息。如图 5.13 所示，端系统接口映射表配置成功。

图5.13 端口映射表配置界面

5.2.2 关键功能测试

（1）监控与配置功能

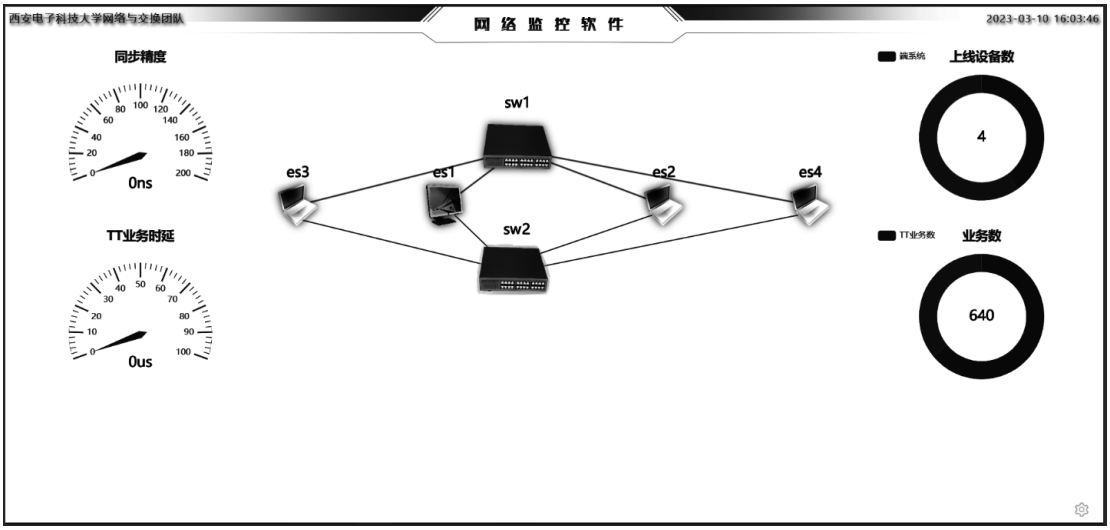

图5.14 监控界面首页

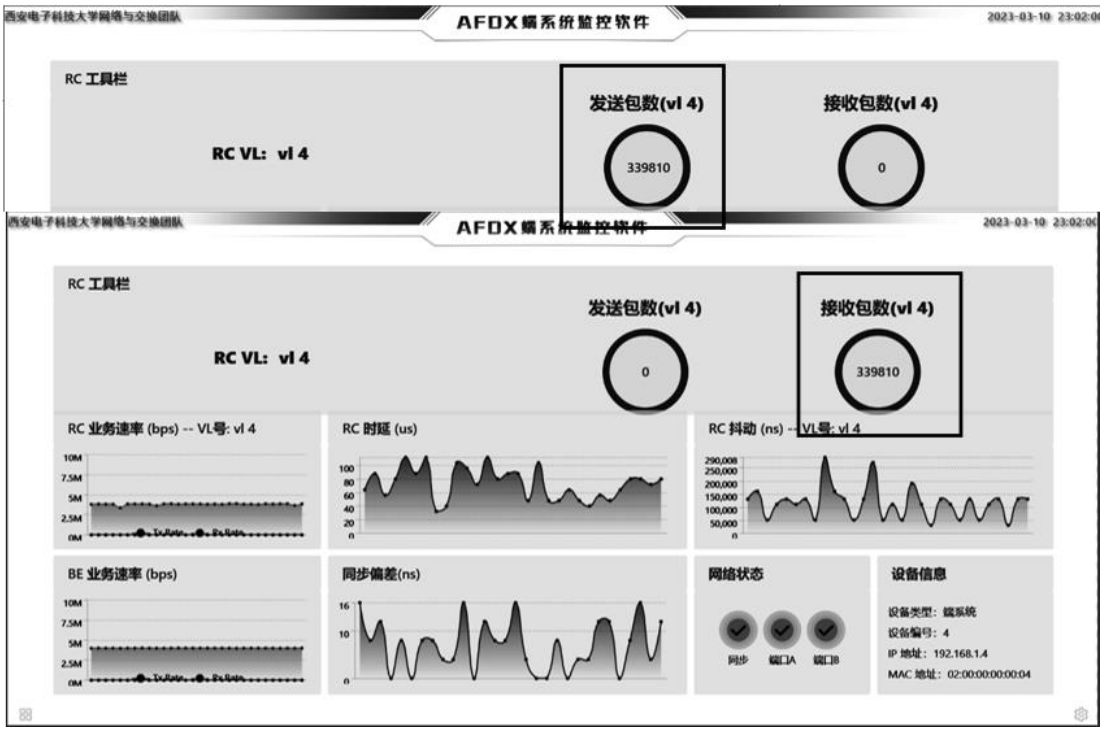

图5.15 数据业务监控

AFDX 端系统的监控功能是本文的创新点之一，通过终端监控软件可直观观测端系统的运行状态，如图 5.14 所示，打开监控界面可见目前拓扑连接中有四台端系统和两台交换机，与实际拓扑连接相同。现将 es3（端系统 3）设置为发送端，es4（端系统 4）设置为接收端，端系统 3 向端系统 4 发送虚拟链路号为“0x4”的 RC 数据

帧，BAG 设置为 2ms。

初始化端系统，配置完成后开始发送业务。打开端系统 3 的监控界面图，如图 5.15 上半部分，可见端系统 3 发送了 339810 个虚拟链路号为“4”的 RC 数据帧。同时打开端系统 4 的监控界面图，如图 5.15 下半部分，可见端系统 4 接收了 339810 个虚拟链路号为“4”的 RC 数据帧，两者数据量相同，说明数据传输过程中没有数据帧丢失，端系统运行正常。

另外，从图中可见 RC 数据帧的实时业务速率、RC 时延、RC 抖动等状态信息，时延小于 150 微秒，符合技术要求，RC 业务速率为 4 兆左右，与配置的 BAG 匹配，也证明端系统功能正常，达到预期。

图5.16 配置软件界面

如图 5.16 所示，配置软件既可配置端系统运行所需的表格，也可以配置一些功能的开关和运行参数，以网口速率配置为例，连接好设备拓扑图后，将网口配置为千兆，点击“SET”，右侧方框处可见“SET Successfully”，表明网口速率配置成功。配置软件界面还可以获取硬件板卡的状态信息，再次确认配置是否正确，点击“GET”，右侧方框处可见“Interface mode：1000Mbps”，即此时端系统网口速率为千兆，由此可见配置功能符合预期设计目标。

（2）故障注入功能

图5.17 故障注入界面

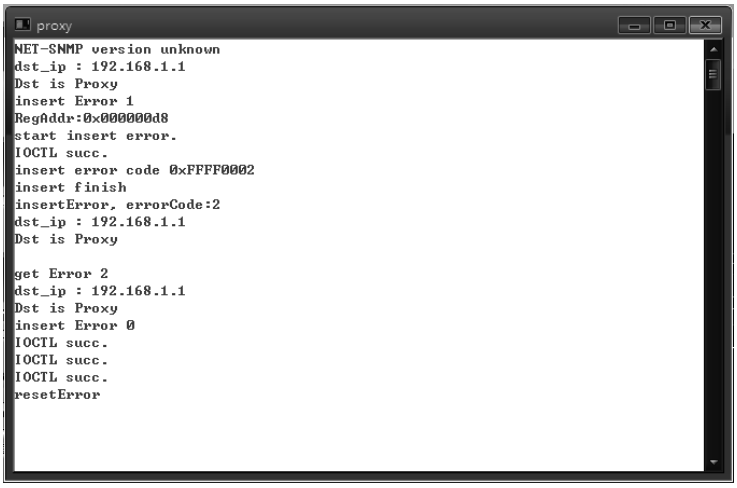

图5.18 故障注入结果

如图 5.17 所示为故障注入界面，以注入数据入队干扰为例，初始化配置结束，开始发送数据，而后点击“注入”可在右侧方框内观测到“SET Successfully”，即已注入故障，点击“获取”，显示“DATA_ENABLE_ERROR:inserted”，即故障已注入成功。如若置位，则只需点击“复位”。图 5.18 为故障注入端的打印结果，显示故障已注入成功。图 5.19 为故障注入前现象，可见数据帧信息、CRC、序数完整性等状态灯都显示通过，右侧健康状态一直在发生变化，而健康状态是通过数据接收的时间计算的，说明一直有数据处于接收状态。

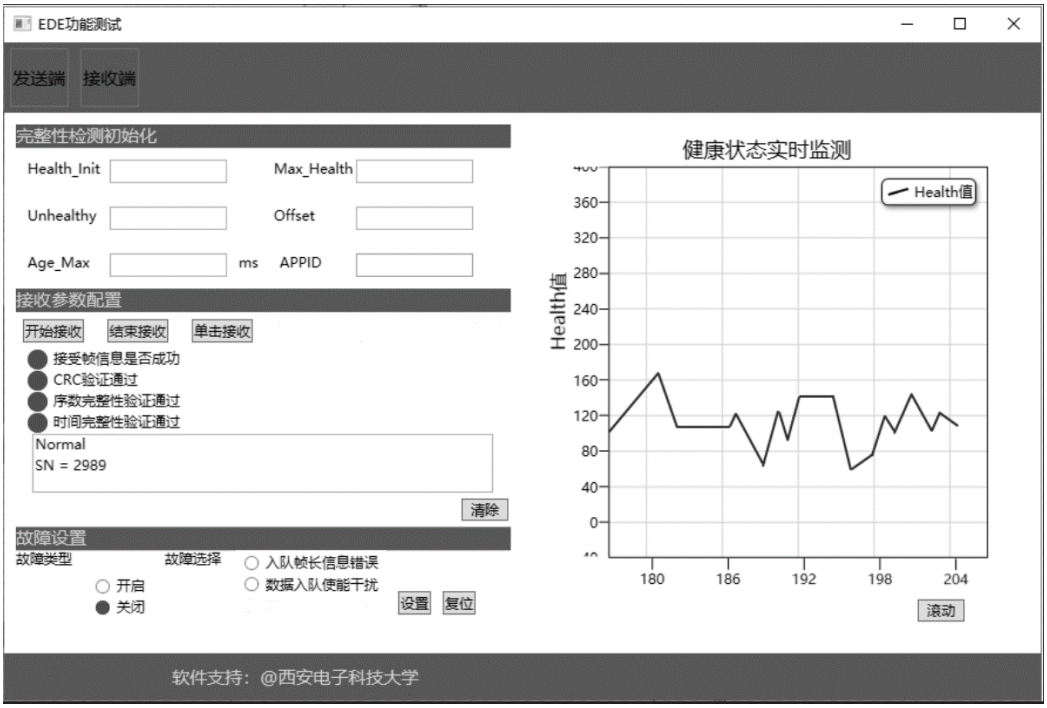

图5.19 故障注入前现象

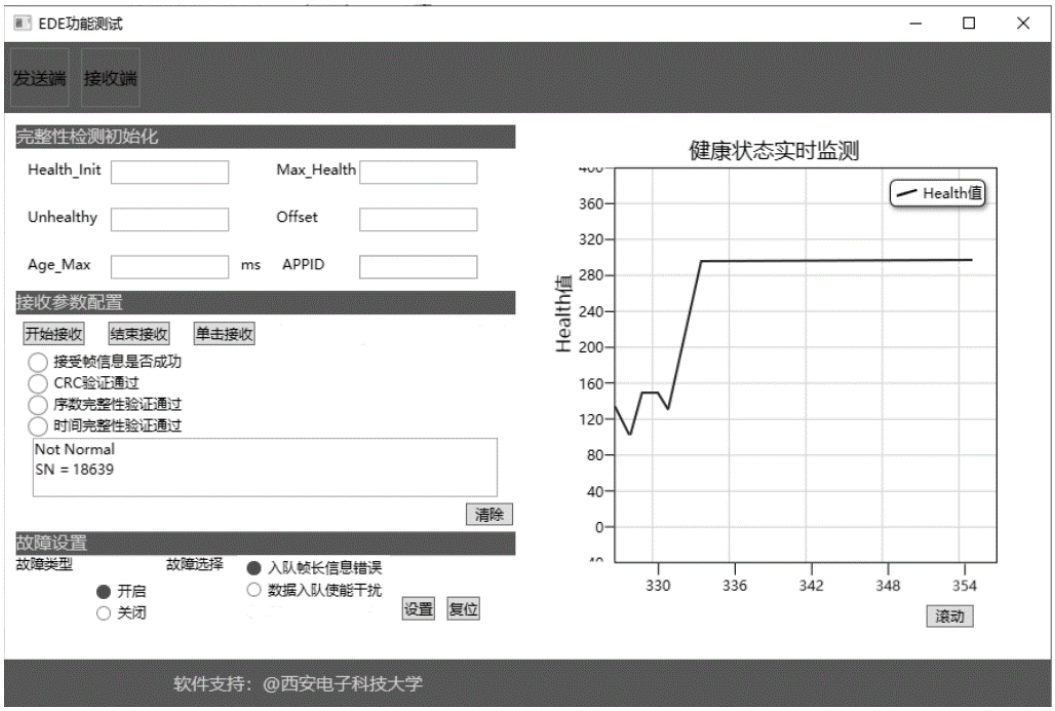

图5.20 故障注入后现象

如图 5.20 所示，开启故障注入并选择入队帧长信息错误后，由于队管维护的数据信息出错，接收端将无法接收正确数据，故而帧信息、CRC、序数完整性等状态指示灯不再亮，因为没有数据接收，左侧界面显示“Not Normal”，SN 号不再增加，同时数据的时间将维持不变，故而健康状态为定值。对比图 5.19 和图 5.20，可见故障

注入功能达到设计目标，且现象十分明显，有助于使用者直观地了解设备在真实环境下发生故障的现象，并且衡量故障发生后的结果是否在可控范围内。

（3）数据处理

图5.21 端 1 BE 数据收发界面

图5.22 端 2 BE 数据收发界面

如图 5.21 和图 5.22 所示，端系统 1 和端系统 2 互发一段时间 BE 数据帧，且两

者都是每隔 2500 微秒发送一个帧长为 1000 字节的数据帧，理论上发包速率和收包速率应该为 400 帧每秒，考虑到端系统调度 RC 数据帧会花费一定时间，交换机处理 BE 数据帧时间也有一定差异，实际速率应该小于但接近 400 帧每秒，而图中可见端系统 1 和端系统 2 的收发包速率值符合理论。另外，对比端系统 1 发送的数据帧和端系统 2 接收的数据帧，互发时间内两者数值相等，同样，端系统 2 发送的数据帧和端系统 1 接收的数据帧数值也相等，表明端系统的 BE 数据帧处理相关的功能正确，达到设计目的。

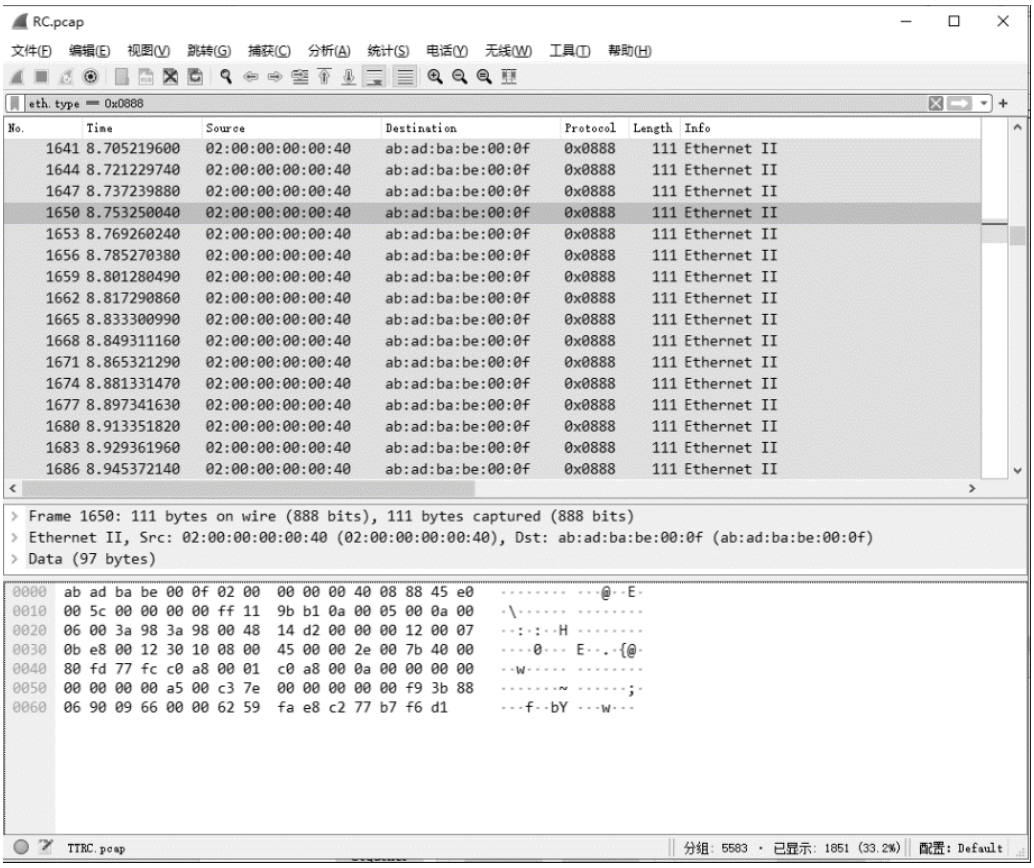

图5.23 端 1 RC 数据发送数据

端系统初始化配置完成后，利用软件作为数据源，图 5.15 已表明接收数据帧和发送数据帧的数目相等，数据传输过程中没有丢帧的情况，在此通过 WireShark 抓包软件验证 COM 数据帧和 SAP 数据帧的操作流程准确性。

如图 5.23 所示，为端系统 1 发送的 MAC_SAP 类型的 RC 数据帧，端系统 MAC 地址为“0x0200 0000 0001”，但从板卡网口发送的数据帧显示其源 MAC 地址为“0x0200 0000 0040”，表明数据在端系统板卡内已经替换源 MAC 地址，验证了 SAP 数据处理相关模块的正确性。

如图 5.24 所示，为端系统 2 发送的 MAC_COM 类型的 RC 数据帧，端系统 MAC 地址为“0x0200 0000 0002”，但从板卡网口发送的数据帧显示其源 MAC 地址为

“0x0200 0000 0140”，且数据不具备 IP 头部和 UDP 头部，表明数据在端系统板卡内仅添加 MAC 头部，符合 COM 数据的处理预期，验证了 COM 数据处理相关模块的正确性。

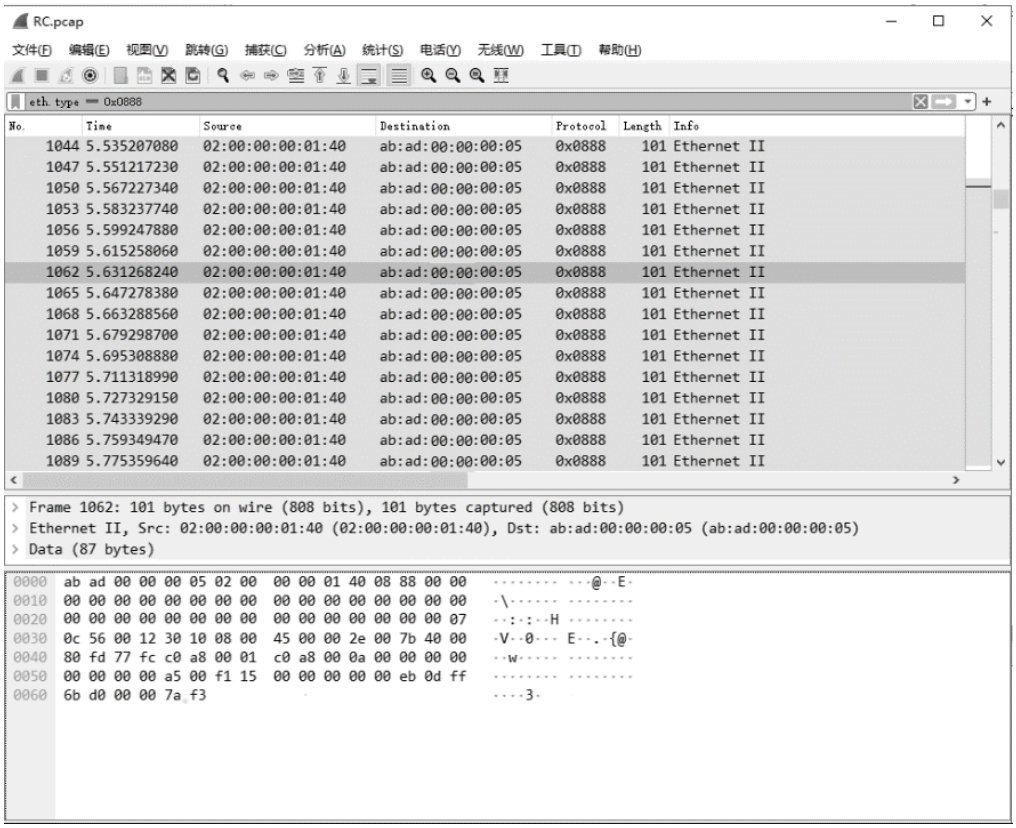

图5.24 端 2 RC 数据发送数据

5.3 测试问题及解决方法

5.3.1 跨时钟域问题

（1）问题概述：在故障注入和运行参数配置时，软件操作并不总是能够立即生效，有时候需要多次配置硬件板卡才能识别软件指令。

（2）问题分析：上层操作系统和板卡内部的时钟源不同，板卡内部逻辑由晶振经过倍频得到，为 125MHz，而操作系统下发数据的时钟由接口提供，为 62.5MHz，两者之间频率不同，故而存在跨时钟域问题。故障注入和运行参数配置指令有效周期为一个接口时钟周期，由于电路中可能存在亚稳态等因素，导致硬件板卡逻辑无法采集到有效信号。

（3）问题解决：故障注入和运行参数配置指令为单比特信号时，应该先展宽再于板卡内部逻辑时钟域下采样，多比特信号则需要采用多周期路径规划，修改后，故障注入和运行参数配置立即生效。

5.3.2 代码设计问题

（1）问题概述：上板验证时，如果发送端同时发送 RC 数据帧和 BE 数据帧，接收端经常卡死并且无法恢复，而且这种异常情况并非偶然，需要重新上电端系统才能正常工作，但发送端却不受任何影响。

（2）问题分析：由于发送端不受影响，故而问题出在接收端，且发送端单发 RC 数据帧和 BE 数据帧都不会导致异常情况，经抓信号定位，最后发现接收端存储控制模块卡死，初步猜测问题出在两个数据靠得太近，数据信息处理耗费时钟周期稍长导致 FIFO 切换不及时。

（3）问题解决：仿真时设置 BE 数据源和 RC 数据源，且 BE 数据源帧长不固定，打高速流，发现这种情况确实会导致 FIFO 切换出错，更改代码逻辑，如果数据信息还未处理结束，不允许切换 FIFO，另外将数据隶属于 RC 类型还是 BE 类型的判断逻辑放置于 MAC 内以节省时延，改动完成再上板，原异常情况不再发生。

5.3.3 软硬件联调问题

（1）问题概述：和软件人员联调，进行配置、数据收发测试时，发现系统并不能正常工作。

（2）问题分析：仿真时端系统内部内置数据源并不会出现问题，而通过操作系统下发指令的方式却有差错，显然是因为操作流程不规范，或者内部寄存器规范不同。

（3）问题解决：与软件人员核对寄存器的地址、配置流程和收发数据流程，发现部分功能软硬件定义的寄存器地址不同，配置和收发数据之前软件没有向硬件发送操作起始命令。经修改后，软硬件联调不再出现问题。

5.4 本章小结

本章从仿真验证和板级验证两方面测试了新型 AFDX 端系统功能是否符合设计需求，达到设计目标。其中，仿真验证从 COM 数据处理、SAP 数据处理、故障注入以及监控和配置四个层面表明了设计功能模块的正确性，板级验证从监控和配置、故障注入以及数据处理三个层面表明了真实环境下设备能够如同预期一样正常运行。本章结尾处还针对测试过程中的问题进行总结和思考。

第六章 总结与展望

6.1 工作总结

在发展国产大飞机的背景下，AFDX 技术在新一代机载网络系统的应用潜力显得更为突出，依托于实验室科研项目“AFDX 端系统 IP 核及驱动软件研制外包”，本文主要介绍基于 ARINC664 协议和实际项目需求的 AFDX 端系统 IP 核的架构，并就架构着重介绍关键模块的详细设计。

本文的工作主要集中于以下几点：AFDX 技术的研究、面向多业务场景的 AFDX 端系统 IP 核架构的设计、面向多业务场景的 AFDX 端系统 IP 核的具体实现。

第一点：AFDX 技术的研究。以时间为维度，清晰简明地概述以太网技术发展的过程，亦即 AFDX 技术产生的原因。接着比对国内外这项技术发展的差异性以表明本文的意义。之后，简单介绍 AFDX 系统的组成和业务的划分，并讲解 AFDX 端系统中用到的部分相关技术。

第二点：面向多业务场景的 AFDX 端系统 IP 核架构的设计。首先给出面向多业务场景的 AFDX 端系统的整体架构，接着与第一代端系统比较，通过差异点体现新型端系统的优异之处，同时引出本文的创新点，再简略介绍各模块功能、数据处理流程、帧格式和表项含义，以便读者对新型端系统工作方式有大致的理解。

第三点：面向多业务场景的 AFDX 端系统 IP 核的具体实现。详细介绍各模块的功能、设计，辅以框图和状态机，行文时着重于架构的创新之处，即通信端口业务、服务访问点业务的处理、故障注入、监控与配置。之后，通过仿真测试的手段验证端系统主要功能点，并搭建板级验证环境，证明新型端系统应用于真实环境下的可行性。

最后，仿真和上板的结果表明，本文设计的面向多业务场景的 AFDX 端系统 IP 核达到预期的设计目标，既符合协议规定，又满足设计需求，同时融入诸如监控、配置等功能，便于用户调试和观测，整体优点突出。

6.2 研究展望

在对本文设计的面向多业务场景的 AFDX 端系统 IP 核进行仿真及上板验证时，笔者观察思考后，发掘到有几处设计不够优异，仍有待相关研究者日后能不断改善与提升，

第一点：部分模块耗费时钟周期过多。接收端存储控制模块存储数据后，数据信息需经过多个模块才能判断此数据帧是否符合要求，尤其是哈希查表可能耗费多个时钟周期，不利于数据流的快速处理，可能成为端系统速率提升、升级换代之路上的绊

脚石，针对此问题，改进方向为将原有哈希查表更改为双哈希查表，通过两个互补的生成式，可以完全避免哈希冲突问题，另外在前级模块处理数据信息时，可以同时为后级模块做准备工作，例如在进行完整性检查时，就可以计算业务类型匹配表的查表地址，会减少操作时延。

第二点：部分模块资源利用过多。例如业务调度和队列管理模块，使用了大量的存储资源和查找表，可能导致现有端系统 IP 核在应用于部分资源较少的硬件板卡时出现时序无法收敛等问题。以 RC 数据调度的判别条件为例，每一条虚拟链路号都要维护一个 BAG 判别模块，耗费大量的查找表资源，很有可能引起布局布线拥塞，导致建立时间违例。针对这种情况，由于 BAG 由用户事先规划配置，可以尝试把 BAG 规划入业务调度表中，让调度片的长度和 BAG 挂钩，没达到 BAG 要求前不在调度片上规划某个虚拟链路号的 RC 数据帧。

第三点：增加故障注入和可监控状态的个数。当前设计仅为满足项目合作方要求，而面向多业务场景的 AFDX 端系统 IP 核未来肯定会继续提升性能，应用场景也会越来越多，建议结合设备运行环境，增加故障注入的类型，并且考虑增加设备能自检并从故障中恢复的功能。另外，可以监控端系统内部模块的关键状态机，便于端系统升级时的调试。

参考文献

[1] 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/OL]. 计算机应用研究: 1-4[2022-04-05].

[2] IEEE Standard for Ethernet Amendment 5: Specification and Management Parameters for Interspersing Express Traffic[C]// IEEE Std 802.3br-2016 (Amendment to IEEE Std 802.3-2015 as amended by IEEE St802.3bw-2015, IEEE Std 802.3by-2016, IEEE Std 802.3bq-2016, and IEEE Std 802.3bp-2016). IEEE, 2017.

[3] 范超, 赵琳, 段海军. 航空数据总线技术研究[J]. 信息技术与信息化, 2022, No.265(04): 160-163.

[4] 张思军, 冉玲. 计算机局域网技术发展及维护研究探析构建[J]. 科技创新与应用, 2020, No.293(01): 141-142.

[5] 朱其新. 网络控制系统的建模、分析与控制[D]. 南京航空航天大学, 2003.

[6] 张森, 严小双, 晏恺晨等. 机载总线技术应用综述及其对飞机性能的影响[J]. 电子世界, 2019, No.579(21): 37-38.

[7] 逯计划. 机载总线技术发展研究[J]. 电子测试, 2017, (7): 72-73.

[8] Airlines Electronic Engineering Committee. ARINC 664 P7 - AIRCRAFT DATA NETWORK PART 7 AVIONICS FULL-DUPLEX SWITCHED ETHERNET NETWORK[S]. AERONAUTICAL RADIO, INC, 2005.

[9] Developing AFDX solution[Z]. Actel Corporation, 2005.

[10] 张涵中. AFDX 航空通信协议及其核心技术分析[J]. 电子世界, 2022, No.631(01): 9-10.

[11] 黎石磊. AFDX 调度算法研究及其在飞行仿真系统中的应用[D]. 陕西: 西安电子科技大学, 2011.

[12] 屈静. 航电机载网络的业务调度与性能评价技术研究[D]. 西安电子科技大学, 2020.

[13] 张志平, 陈长胜, 白杨等. 时间触发数据帧冗余管理方法: CN202010467575. X[P]. 2020-09-22.

[14] 张丽. AFDX 端系统虚拟链路层的研究与仿真[J]. 工业控制计算机, 2022, 35(02): 8-10.

[15] 莫文冬. IP 电话服务质量与带宽利用率的关系研究[D]. 北京: 北京邮电大学, 2003.

[16] 杨峰, 洪元佳, 夏杰, 等. AFDX 网络技术综述[J]. 电子技术应用, 2016, 42(4): 4-6, 10.

[17] 王九龙, 郭中伟, 田庄. 我国载人航天器信息系统技术发展[J]. 航天器工程, 2022, 31(06): 105-116.

[18] 梁晓辉, 黄骅, 张骥等. 工业网络新技术研究与趋势展望[J]. 通信与信息技术, 2022(S2): 43-47.

[19] 孔成磊. 确定性以太网瘦终端的设计与实现[D]. 西安电子科技大学, 2020.

[20] 黄韬, 汪硕, 黄玉栋等. 确定性网络研究综述[J]. 通信学报, 2019, 40(6): 160-176.

[21] 施太平, 娄莉, 田泽. AFDX 协议及关键技术的实现[J]. 测控技术, 2012, 31(10): 81-84.

[22] 赵永库, 李贞, 唐来胜. AFDX 网络协议研究[J]. 计算机测量与控制, 2012, 20(01): 8-10+30.

[23] 马萌. 航空专用数据总线技术研究[J]. 数字技术与应用, 2013(10): 61-63.

[24] 肖红玉. AFDX 航空通信协议及其核心技术分析[J]. 无线互联科技, 2021, 18(08): 7-8.

[25] 刘马飞, 曾学文, 倪宏. 一种基于碰撞等待的 BE 业务带宽请求方案[J]. 微计算机应用, 2010, 31(8): 13-18.

[26] 田泽, 姜丽云, 陈伟等. AFDX 网络关键协议分析与研究[J]. 电子技术应用, 2016, 42(4): 7-10, 14.

[27] 周德新, 王彦翔. AFDX 网络分析及带宽分配间隔设定策略[J]. 科学技术与工程, 2016, 16(9): 226-230.

[28] 栾昌海, 马艳. Serdes 技术发展介绍以及未来的挑战[J]. 中国集成电路, 2022, 31(11): 49-53.

[29] 白光媛. 网络芯片物理编码子层关键电路的设计及验证[D]. 西安电子科技大学, 2015.

[30] 王元勋, 张双, 孔德岐等. AFDX 网络端系统测试与分析[C]// 中国航空学会. 2020（第九届）民用飞机航电国际论坛论文集. 科学普及出版社, 2020: 6.

[31] 张杨阳. 航电通信网络中简单文件传输协议的实现研究[J]. 信息通信, 2019, No.193(01): 82-83.

[32] 张希遥. TFTP 以太网通信[J]. 赤峰学院学报(自然科学版), 2013, 29(16): 21-23.

[33] 张晓丽, 彭寒, 景月娟. 综合航电分区间通信元模型设计研究[J]. 计算技术与自动化, 2019, 38(04): 162-166.

[34] 付强. 机载网络数据总线仿真技术研究[J]. 软件导刊, 2018, 17(02): 164-167.

[35] NAVID R, CHEN E H, HOSSAIN M, et al. A 40 Gb/s serial link transceiver in 28 nm CMOS technology [J]. IEEE J Sol Sta Circ, 2015, 50(4): 814-827.

[36] 李岚, 苏敏, 程丽彬等. 基于 FPGA 多接口的千兆以太网 IP 核设计[J]. 电子信息对抗技术, 2020, 35(02): 82-88.

[37] 郑红党, 孙彦景, 李松等. 基于 FPGA 技术的 8B/10B 信道编码设计[J]. 实验技术与管理, 2021, 38(03): 40-44.

[38] 兰雨娇, 侯伶俐, 岳宏卫等. 一种高速串行信号线性均衡电路[J]. 微电子学, 2020, 50(04): 514-520.

[39] 朱佳. 千兆以太网的 SerDes 接口电路设计[D].江南大学, 2021.

[40] 舒畅. 高速 SerDes 系统中多相延迟锁相环的设计[D]. 杭州电子科技大学, 2022.

[41] 高速双模 SerDes 接收机关键技术研究[D]. 上海交通大学, 2020.

[42] 谢希仁. 计算机网络[M]. 第 5 版. 北京: 国防工业出版社, 2009.

[43] 石华. 基于 FPGA 的万兆以太网 TCP/IP 卸载引擎与硬件系统设计[D]. 华东师范大学, 2020.

[44] 张丽, 陈利云, 李润青. 基于 FPGA 的 AFDX 发送端虚拟链路层的研究[C]// 中国航空学会, 中国航空研究院. 第八届民用飞机航电国际论坛论文集. 航空工业出版社, 2019: 6.

[45] 李健, 张激, 施刚. 面向航空电子的分区操作系统[J]. 计算机工程, 2008, 34(z1): 69-71.

[46] 范毓洋, 肖洪, 李子航. 基于时间窗的 AFDX 端系统调度策略设计实现[J]. 电光与控制, 2023, 30(01): 109-112+119.

[47] 王雷淘, 乔庐峰, 续欣. 一种应用于星载交换机的 DDR3 共享存储交换结构的设计与实现[J]. 通信技术, 2020, 53(6): 1546-1553.

[48] 许晶. 10G 专用交换队列管理单元的设计与实现[D]. 西安电子科技大学, 2017.

[49] 陈长胜, 一种余度管理电路及其管理方法. 陕西省, 中国航空工业集团公司第 631 研究所, 2010-12-01.

[50] 杜宏伟, 马捷中. 航空电子全双工交换式以太网及其关键技术研究[J]. 测控技术, 2008, 27(12): 65-67.

[51] 钟杰, 何民, 王怀胜等. AFDX 构架及协议研究[J]. 电讯技术, 2010, 50(1): 65-70.

[52] 陈昕, 周拥军, 万剑雄. AFDX 端系统关键技术的设计与实现[J]. 计算机工程, 2009, 35{5}: 1-3.

[53] 刘晓胜, 刘建平, 刘博. 基于 FPGA 的 AFDX 虚拟链路层实现方法[J]. 计算机工程, 2012, 38(19): 233-237.

[54] 郑博文. 基于 FPGA 的哈希算法加速器设计与实现[D]. 江南大学, 2022.

[55] 李宥谋, 房鼎益. CRC 编码算法研究与实现[J]. 西北大学学报(自然科学版), 2006(06): 895-898.

[56] 张季. 针对 SNMP 协议的研究[J]. 电脑迷, 2017(06): 165.

[57] Sanjay Churiwala. Designing with Xilinx® FPGAs[M]. Switzerland: Springer International Publishing, 2017.

[58] 高亚军. Vivado 从此开始, 进阶篇[M]. 北京: 电子工业出版社, 2020.

[59] 王鹏. AFDX 交换机及交换芯片中关键模块的设计[D]. 西安电子科技大学, 2008.

[60] 王媛媛. TTE 端系统驱动软件设计与实现[D]. 西安电子科技大学, 2021.