电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

硕士学位论文

MASTER DISSERTATION

论文题目 基于 PowerPC 在 VxWorks 下的嵌入式仿真系统的设计

学科专业 光学工程

学　　号 201321010202

作者姓名 向　往

指导教师 邱　昆　　教　授

分类号 _____ 密级 内部

UDC [注1] _____

学 位 论 文

基于 PowerPC 在 VxWorks 下的嵌入式仿真系统的设计

（题名和副题名）

向 往

（作者姓名）

指导教师 邱 昆 教 授

电子科技大学 成 都

（姓名、职称、单位名称）

申请学位级别 硕士 学科专业 光学工程

提交论文日期 2016.3.17 论文答辩日期 2016.5.17

学位授予单位和日期 电子科技大学 2016 年 6 月

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

The Design of Embedded Simulation System Base on Vxworks Operating System With Powerpc

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Major: **Optical Engineering**

Author: **XiangWang**

Supervisor: **Prof. Qiu Kun**

School: **School of Communication & Information Engineering**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名：_____ 日期：　　年　　月　　日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名：_____ 导师签名：_____

日期：　　年　　月　　日

摘　要

随着信息化战争时代的来临，信息掌握以及处理能力的大小将是今后空战致胜的关键，航空电子系统肩负着航空平台“突破信息障”的重任，负责所有信息的采集、处理、传输和应用，而其中综合核心处理系统又居于整个航空电子系统的核心位置，是航电系统的大脑和神经。

本文以仿真模拟综合核心处理机系统的设计与实现为目的，主要对整体系统架构，关键器件的选取进行研究，通过分析航电系统网络通信机制，在 VxWorks 嵌入式实时系统下进行各硬件驱动和 BSP 的开发，提出并解决了以 RapidIO 高速通道承载 FC 消息的方案，并通过 RapidIO 交换机以实现 FC 网络消息的交换。实现了一套完整的模拟综合核心处理机，并最终投入实际联调应用。

本文主要内容分为四个部分。首先，介绍了本仿真系统的操作系统支持—VxWorks，系统的两个关键器件 CPU 和交换芯片，并且针对最为重要的两个协议 FC 协议和 RapidIO 协议进行了描述。接着，针对综合核心处理机系统的需求，对仿真系统进行总体架构设计，并对关键模块的设计进行了详细阐述。之后，分别对仿真系统的关键技术进行了详细阐述，涉及本文的核心内容为：BSP 包的设计和 RapidIO 驱动，以及 RapidIO 承载 FC 模块。最后，分三个阶段对本仿真系统进行充分测试，在分析结果确认达标后，最终将仿真系统接入实验室机载网络环境中投入应用。

关键词：综合核心处理机，VxWorks，RapidIO 总线，FC，仿真系统

ABSTRACT

With the advent of the information warfare era, information grasping and concerning processing ability is key to future air battle winning. Avionics system is bound for the "Breakthrough the Information Barriers" task of airborne platform, which is responsible for all the information acquisition, processing, transmission and application, and the comprehensive system core processing occupies the core position of the entire avionics system, which is the brain and nerves for the entire avionics system.

In this thesis, the design and realization of the simulation analog integrated core processor system is the focus, mainly by researching the overall system architecture,the selection of key devices, the analysis of the avionics system network communication mechanism. The VxWorks embedded real-time systems are the hardware driver and BSP development, and the thesis presents a solution, the RapidIO high-speed channel carrys FC news. And through RapidIO switch, it achieves the exchange of FC network news. A complete set of analog integrated core processor is implemented, and the application of the actual joint modulation is finally put into practical application.

The main content of this thesis is divided into four parts. First of all,the operating system support of the simulation system ,VxWorks. Two key devices CPU and exchange chips of the system, for the most important two protocols FC protocol and RapidIO protocol, are described. Then, according to the requirements of the integrated core processor system, the overall architecture of the simulation system is designed, and the key modules of the design are described in detail. Later, the key technologies of the simulation system are described in detail, and the core content of this thesis is as follows: the design of BSP package and RapidIO driver, as well as the RapidIO carrying FC module. Finally, the simulation system is fully tested in three stages. After the result is confirmed to the standard, the simulation system is put into application in the laboratory airborne network environment.

Keywords: Integrated Core Processor System, VxWorks, RapidIO， FC, Simulation System

目 录

图目录

表目录

主要符号表

英文缩写	英文全称	中文释义
AE	Avionic Environment	航空电子环境
IMA	Integrated Modular Avionics	综合模块化航空电子系统
API	Application Program Interface	应用编程接口
ASM	Anonymous Subscriber Messaging	匿名订户消息
CRC	Cyclic Redundancy Check	循环冗余校验
DMA	Direct Memory Access	直接存储器存取
ELS	Extended Link Service	扩展连接服务
SOF	Start of Frame	帧起始符
EOF	End of Frame	帧结束符
FC	Fibre Channel	光纤通道
BSP	Board Support Package	板级支持包
FS	Framing and Signaling	帧与信号处理
ISR	Interrupt Service Routine	中断服务例程
PCI	Peripheral Component Interconnect	外部设备互联标准
RT	Remote Terminal	远程终端
RTC	Real Time Clock	实时时钟
WDT	Watch Dog Timer	看门狗定时器

第一章 绪 论

1.1 研究工作的背景与意义

1.1.1 航电系统现状以及发展态势

近三十年来，现代半导体，雷达技术以及新一代网络技术的飞速发展，使得军用航空电子系统也进入了崭新的时代。对未来航电通讯网络系统也提出了更加严格的要求，在可靠性、实时性、低延时等基本要求的基础上，更加注重高传输速率、开放式系统及统一网络。而 FC(Fibre Channel)光纤通道协议具有高带宽、低延迟、远距离传输和灵活的拓扑结构等优点，它的出现合适地满足未来航电系统互联的所有主要要求[1]。所以，FC 协议将会在未来的航电系统互联起到至关重要的作用。

美国标准化委员会（ANSI）小组于 1998 年制定了一种高速串行通信协议—光纤通道协议（简称 FC 协议）。本协议同时具有快速的、可靠的信道技术和灵活的、可扩展的网络拓扑结构，两者被有机的结合在了一起[2]。FC 协议发展十分迅速，迄今为止已经能够支持很多上层应用协议和指令集以及能够支持多种物理介质。FC 协议能的通信模式包括全双工、半双工和单工等多种通信模式。FC 协议的基本特点有：灵活的、可扩展的结构格局、低位错率、高可靠性、高带宽、高效性、开放性、低迟延[3]。

嵌入式系统出现于 20 世纪 60 年代，经历了几十年的发展，嵌入式系统在各行各业应用的扩张速度急剧加大，各领域对嵌入式的应用需求和要求使得各类嵌入式系统的功能需求愈发提升，使得其设计要求变得越来越复杂。为了适应各界的要求，增强系统的处理能力，目前主流的 32 位微处理器在嵌入式开发领域得到了广泛的使用，与此同时，为了发挥处理器的最大性能，相应的软件开发也需要更加灵活可靠。目前，嵌入式处理器主要有 PowerPC、SC-400、MIPS、68000、ARM 等系列。在工业控制、电子产品、交通系统、通信系统、网络系统、无线系统、生活应用等多个领域得到了广泛应用，且呈现出递增趋势[4]。

嵌入式系统将领先的计算机技术、半导体技术和电子技术融合进各行业中，嵌入式系统在航电方向的应用更是势在必行。这一特质决定了它必然是一个技术密集、高度分散、不断创新、持续维护的应用集成系统；嵌入式系统要求硬件器件的高性能，高集成性，而且为了整个系统的功能集中，性能优越，还需要对软件进行高效的设计与开发，才能最大程度上发挥硬件的高效性，也就是说嵌入式

系统需要软硬件高度契合，才能在相关应用中发挥专项作用[5]。

鉴于此，开发一个嵌入式开发/测试平台用于飞机网络的监控和仿真是十分有必要的。建立一个航空电子系统通信嵌入式仿真平台可以实现对航空电子设备的检测和故障定位，保障飞机的飞行安全，以及对飞行员的日常训练起到模拟作用，同时还可以为航空电子设备开发人员提供一个调试开发平台，使其能更好的理解通信网络系统的工作原理以便加速系统设备的开发和维护。

1.1.2 本嵌入式仿真系统的研究意义

图 1-1 中给出了综合核心处理机的功能示意图，综合核心处理机负责飞机上所有传感器、受动器和飞机资源的控制、数据的处理、通信以及人机交互。本嵌入式仿真系统是对综合核心处理机系统进行仿真模拟，用于代替真实昂贵的综合核心处理机，以满足实现与真实综合核心处理机想同的应用需求。此系统可以仿真航电系统的通信网络及多个终端，通过部署上层应用程序，可以对飞机飞行各种状态，各种飞行数据进行模拟，还能针对各种飞机其他系统，例如雷达、导弹等应用场景和数据进行模拟，可以对航电系统的功能与性能进行前期验证，以及在系统的调试过程中发挥仿真验证的作用。并也可以用于飞行员的地面模拟飞行训练。

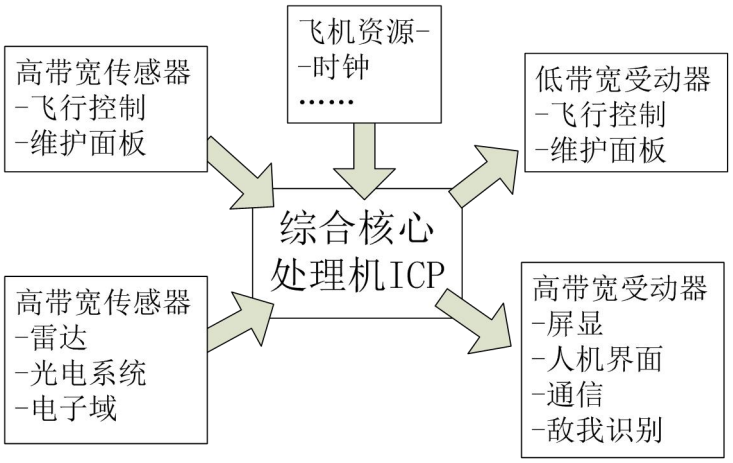


图 1-1 综合核心处理机功能示意图

1.2 综合核心处理机的研究历史与现状

综合核心处理机系统（ICP）作为航空电子系统的核心组成部分，相当于人体中的“大脑”和“神经”，对于实现航电系统的功能综合化、性能高效化、高容错性以及飞机的作战能力和生存能力的提升，保证飞行人员的安全，提高飞机的

可操控性上面都起到关键性的作用。

航空电子系统作为飞机上的核心，对飞机的控制、高性能、安全可靠、以及作战能力都起到至关重要的作用，经历了长时间的发展，航空电子系统经历了几次关键的革命，每一次都使现代飞机各方面迅速发展，并且为下一次航空电子技术的变革起到了铺垫作用。航电系统经历了分布式、联合式、综合式、先进综合式四个发展阶段，航电系统先进性和各方便的性能都得到不断提升。而航电系统性能、功能增长的过程之中，最重要的体现就是在对综合核心处理机的技术更新之上，因此综合核心处理系统在整个航电系统的发展过程中不断的在更新。在 F-16“战隼”战斗机上的联合式航电系统中，核心部分是一台承担系统运行管理、控制及总线调度的系统管理部件。发展到 F-22“猛禽”战斗机的高度综合航电系统后，系统中综合核心处理机系统是多种先进技术的聚集地，所有计算、处理、控制和管理的功能都在综合核心处理机中完成。

计算机技术和数字技术的飞速发展，使得航空电子技术迅速成长，变得更加的智能化和综合化。其中，航电系统网络决定了航电系统的拓扑结构，而航电综合处理机决定了航电系统网络的功能和性能，因此直接影响了航电系统综合化的进程。航空电子系统作为航电网络空间的一个重要节点，担当了整个飞行过程中最为繁重的任务，其中综合核心处理机负责整个飞机上所有功能的集成、机载数据处理和传输、整个飞机的系统的监测与检测，以及最后的应用控制，成为飞机名副其实的中枢神经系统，成为了保障航空电子系统正常、稳定、高效工作的必要组成部分。

1.3 本文的主要贡献与创新

课题来源于某研究所横向项目“模拟 ICP 设备”。本文主要对航电综合核心处理机系统的仿真模拟进行研究，进行了整体系统架构的设计、各模块间的通信机制的设计。其中主要创新集中在此系统的 RapidIO 驱动设计开发，利用 RapidIO 高速通信承载 FC 消息以实现 FC 消息的交换，通过分析 RapidIO 消息类型与格式，FC 帧格式，将 FC 帧承载在 RapidIO 消息的载荷中，再通过地址转换算法，将 FC 地址和 RapidIO 地址进行对应转换，在 RapidIO 交换的基础上实现了 FC 的交换。最终形成一套完整的机载集成处理仿真系统，并投入实际应用。

1.4 本论文的结构安排

本文的章节结构安排如下：

第一章从航电系统的发展现状和发展态势和本嵌入式系统的研究意义两个方

面，引出本课题研究工作的背景及意义。之后介绍了综合核心处理机的研究历史与现状，以及本文的主要贡献与创新。

第二章分别介绍本仿真系统的操作系统—VxWorks，以及仿真系统中最为重要的两个器件，PowerPC8640 和 CPS-1616 交换芯片的特征以及为何选用这些器件。接着从层次划分和帧结构这两方便详细介绍了 FC 协议，最后对 RapidIO 协议进行了介绍。

第三章针对航电机载集成处理系统需要解决的问题，对嵌入式仿真系统进行了总体上的设计。首先叙述系统整体功能及性能需求，对整个系统的架构提出了解决方案；接着从系统的通信需求出发，对主要的通信方式进行了介绍和设计方案的描述，描述了两种通信网络；最后对本系统的硬件平台进行了简述，对两个模块处理器板和交换板进行了结构和功能的介绍。

第四章对模拟仿真综合核心处理机系统的关键技术进行了详细的阐述，其中 BSP 开发和硬件驱动的设计，驱动方面主要着重介绍了 RapidIO 驱动的设计与实现。详细介绍了 RapidIO 承载 FC 消息模块的设计与实现，针对以 FC 交换机作为交换设备和以 RapidIO 交换机作为交换设备时数据流的对比，提出了增加适配层的方案，接着详细描述了适配层的实现，以及中转节点的实现。此章节是本文的核心内容。

第五章分三个阶段对模拟仿真综合核心处理机系统进行通信测试。第一阶段的 RapidIO 消息的测试，此阶段 RpiadIO 不承载 FC 消息；第二阶段进行 FC 消息的测试；第三阶段进行 RapidIO 承载 FC 消息的通信测试。对测试结果进行分析，确认通过达标。最后将仿真系统可作为核心处理设备接入实际机载网络中投入应用。

第六章对全文进行总结与展望。

第二章 关键器件与相关协议的分析

2.1 操作系统支持---VXWORKS

大多数的嵌入式实时操作系统是单地址多任务环境，内核和应用处于同一特权级。由于地址空间的单一，因此某些应用可以访问到内核的地址空间上去[6]。介于这种情况的可能性，某些应用可能由于访问了内核的地址空间而改变了内核的运行情况最终影响到其他应用甚至内核，影响其他应用的正常工作，或者威胁内核运行状态，更而严重的会使得整个系统崩溃。

总所周知，航电系统领域通常对实时性、安全性以及应用的丰富性的要求越来越高，而本文中的嵌入式仿真系统为了符合航空电子应用软件标准接口（ARINC653）标准，选用了风河公司 VxWorks653 商用操作系统。

2.1.1 VxWorks 简介

ARINC653 主要描述了模块化综合航空电子设备 IMA(Integrated Modular Avionics)使用的应用软件的基本环境。它描述了航空机载应用与操作环境之间的接口、服务操作、以及两者间数据交互的行为，并描述了嵌入式航空电子的软件运行环境。

VxWorks653 是一个专用实时操作系统，面向关键安全级别 ARINC 653 集成模块化航空电子系统(IMA)，针对航电机载设备所需繁多的控制功能以及高效性能的要求，做出了与传统嵌入式实时操作系统所不同的特殊改动[7]。实现了分布式区间管理、各模块区间通信以及系统的安全监测等增强功能，以满足航空电子系统设备所面临的高要求。

VxWorks653 在系统架构上将系统进行分区管理，确保每个区间上的应用调度处于各自区间级别上，这些应用共享同一区间资源；区间管理则阐述了区间调度中主框架的原理规则，并描述了各区间的通信方式；并对关于安全监测中所可能出现错误等级区分和各等级错误的处理方式等问题作出解释。

2.1.2 VxWorks653 内核 OS

VxWorks653 操作系统由内核 OS 实现管理，此种操作系统管理方式使得各应用在时间和空间进行了有机的隔离。对比传统的嵌入式实时操作系统，这种管理隔离模式使得每个分区不会互相制约互相影响，即使某个区间出现了问题，也不会导致其他区间应用受到影响，从而提高整个嵌入式系统的安全性，可靠性[8]。

VxWorks653 的平台使用高性能的两级调度构架，其实现了在分区间上下文切换的低开销。VxWorks653 也支持标准的 ARINC 时间调度抢占算法。

VxWorks653 平台也支持一种基本的模式调度，即一组分区可以事先静态配置调度策略，也可以由内核 OS 动态决定其调度策略。

此外，VxWorks 的 653 平台提供混合模式的调用策略，称为 APPS（ARINC plus priority-preemptive schedul-ing）。在 APPS 调度策略中，当发生以下几种情形时采用基于优先级的调度策略[9]。比如，在一个分区的时间片内检测到空间时间（idle time），或者一个应用程序放弃它的时间片，再或者，存在空闲的分区时间（时间未分配）。基于优先级抢占式调度时，具有最高优先级，但没有空闲时间的分区的任务将会被调度到空闲时间进行执行。APPS 确保 VxWorks653 平台系统空闲时间的有效利用，无形中降低了高优先级任务的响应延迟。

2.2 PowerPC 的选取---PPC8640

Freescale（飞思科尔）的 MPC8640 处理器已经应用比较广泛了，它在嵌入式系统、存储领域、电信科技、军事、工业制造和普适计算等对性能、集成要求特别高的行业或领域都带来了突破性的应用成果。

高集成度是 MPC8640 的最大优势所在，它是基于 Power Architecture®技术的高性能 e600 内核与 PowerQUICC® 片上系统(SoC)平台相结合的产物。它集成了的南北桥功能，它作为单芯片完成的工作，等同于在其他设计方案中所需要的 3 个芯片的功能和性能需求。而且它的电路板规格相比其他普通芯片要更小，更加说明它的高集成度特点[10]。不仅如此，电路板的设计，布局方便也相比其他芯片更为优势，因为它内置了所有内核至外设的连接。图 2-1 为 MPC8640 的内部结构图。

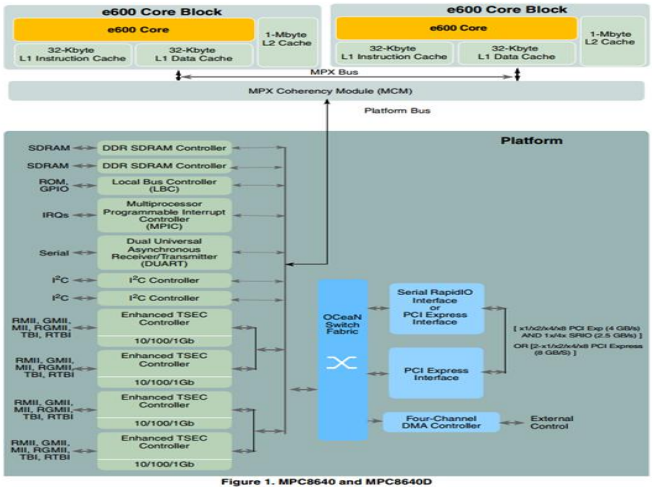


图 2-1 MPC8640 内部结构图

2.3 RapidIO 交换芯片---CPS-1616

2.3.1 CPS-1616 简介

CPS-1616（型号 80HCPS1616）是一个符合 RapidIO 规范（2.1 版）的中心包交换机，其功能是为在 DPS、处理器、FPGA、其他交换机或者任何其他基于 RapidIO 的设备之间分发的包提供中心路由选择。设备的低延时、可靠的包传输以及高吞吐量使其成为理想的嵌入式应用，包括通信、图像或工业控制[10]。交换的 S-RIO 结构允许有真正对等通信的拓扑结构。支持四种标准 RapidIO 级别的优先级，且可以将包单点传送、多点传送或广播到目标端口。这使其成为理想的用于穿过背板或线缆通信的设备。图 2-2 为 CPS-1616 芯片的结构示意图。

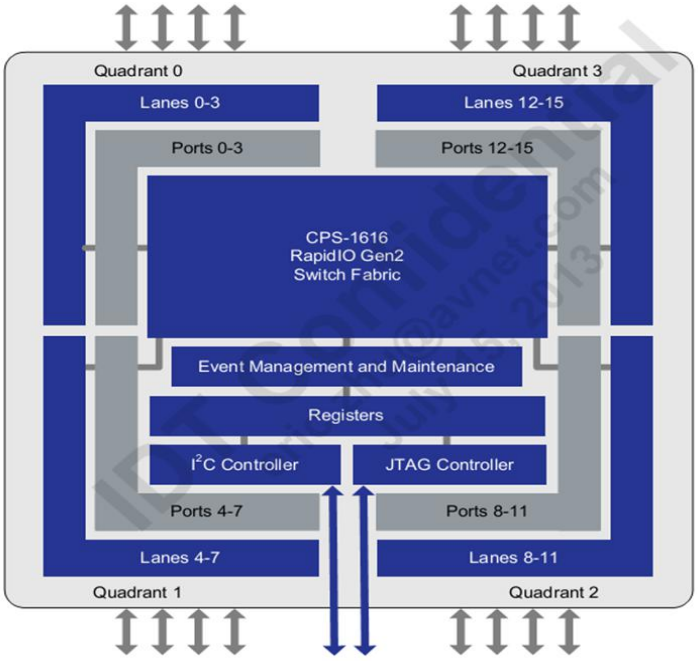


图 2-2 CPS-1616 结构示意图

CPS-1616 包括以下几点特性。

1. RapidIO 端口：16 双向 S-RIO 通道；1X、2X 和 4X 端口宽度允许每个端口最大为 20Gbps；端口速度可选：6.25、5、3.125、2.5 或 1.25Gbaoud；就各物理层速度而言，支持定义为级别 I 的端或长到达，和定义为级别 II 的短、中或长期到达；支持错误管理扩展；软件辅助错误恢复，支持热拔插。
2. I2C 接口：提供用于维修和错误报告的 I2C 端口；支持主从操作；主设备允许来自外部 ROM 上电配置；
3. 交换：80Gbps 峰值吞吐量；非阻塞数据流结构；贯通或存储和转发数据流

可配置；各种长度包的低延时；内部队列缓存和转发缓存；发送和接受的流量控制；全局路偶或本地端口路由容量；最大支持 16 个同时多点传送掩码，带广播；性能监控技术器用于性能和诊断分析，每个输入端口和输出端口计数器。

4. 串行解串器：发射机预加重和驱动长度+接收机均衡提供可能的最佳信号集成；嵌入式伪随机位序列生成器和检测编程多项式支持误比特率测试。
5. 附加信息：包追踪/镜像，每个输入端口可将所有符合用户定义标准的进入包复制到“追踪”输出端口；包过滤器，每个输入端口可过滤所有符合用户定义标准的进入包；可通过 S-RIO 端口、I2C 中进行设备配置[11]。

2.3.2 CPS-1616 的交换和加强功能

交换——所有包按照 RapidIO 规范（2.1 版）进行交换，包目的 ID（destID）决定包的路由，四个主要交换选项存在：

1. 单点传送：按照 RapidIO 规范（2.1 版）,根据包的 destID 将包发送到单个目的端口。
2. 多点传送：destID 指向多点传送掩码的包将多点传送至多点传送掩码提供的所有目的端口。遵照 RapidIO 规范（2.1 版）进行多点传送。
3. 广播：可通过配置各多点传送掩码使所有输出端口（包括源端口）包含在目的端口中，用于前述多点传送操作。

CPS-1616 支持 240G 的峰值吞吐量，此为 8 端口 4X、6 端口 2X 和 4 端口 1X 配置的线路速率（每个均为 5.0Gbaud = 6.25Gbaud-S-RIO 定义的 8b/10b 编码），且按照包头和优先级进行动态交换。

加强功能——提供加强功能是为了支持系统调试。用户可选的功能包括下列功能[12]：

1. 包追踪：包追踪功能根据用户定义的比较寄存器值，对每个进入包比较进行全速校验（包头加一部分净荷）。所有 S-RIO 端口都具备追踪功能，各端口彼此独立运行。如果启用某个端口的追踪功能，则最多会根据四个比较寄存器来校验各进入包的匹配性。如果有一个匹配，则两个可能的用户定义行为中的任一行为将会发送：
   i. 不仅包路由正常通过交换机到其恰当的目的端口，而且还会复制相同的包到“调试端口”或“追踪端口”。追踪端口自身可以是任何标准 S-RIO 端口。用作追踪端口的端口由用户通过简单的寄存器配置进行定义；

ii. 包被丢弃。如果没有匹配，则包路由正常通过交换机，不发生任何行为。可在系统初启和样机研究期间使用包追踪功能，确定用户感兴趣的具体包类型。可用于安全应用中，此时必须校验包的包头或净荷中的标签是否正确。之后，经确认（匹配）的包经路由到达追踪端口，被主处理器接收，可在软件层进行干预；

2. 端口环回：CPS-1616 为每个可用于高速 S-RIO 端口系统调试的端口提供内部环回。通过启用端口上的还回，发送至端口接收机的包立即在物理层经环回至发射机的旁路逻辑层或传输层。
3. 广播：设备交换操作支持广播流量（任何输入端口到所有输出端口）；
4. 安全功能：前述包追踪/过滤器能力允许在输入端口阻塞符合追踪标准的包。例如，该功能可允许过滤不信任的（未知的源或目的）包，过滤恶意或错误的维护包，或识别启动包传递到从设备。

2.4 FC 协议简介

光纤通道协议（Fibre Channel Protocol）是由 ANSI（American National Standarde Institute）提出的串行通信标准。光纤通道支持的高带宽，带来的低延迟、低位错率，支持热拔插，可连接的设备众多，以及它灵活的拓扑结构，这些特点使得它成为了未来航空电子系统互联的首选标准[13]。该标准是一些列协议构成的协议族。经过数次修订，和原有的光纤通道物理和信号接口 PH、PH2、PH3 标准对比，该协议族采用光纤通道帧和信号 FC-FS 以及光纤通道物理接口 FC-PI 标准代替。目前 FC 光纤通道系列标准包括以下 8 项独立的标准规范：光纤通道航空电子环境 FC-AE、光纤通道帧和信号 FC-FS、光纤通道交换网络 FC-SW、光纤通道仲裁环 FC-AL、光纤通道通用服务 FC-GS、光纤通道交换网络通用要求 FC-FG、光纤通道物理接口 FC-PI、光纤通道音频视频 FC-AV[14]。

2.4.1 FC 协议层次划分

FC 协议在功能层次上由下至上划分为 5 层，分别为 FC-0 物理层、FC-1 编码层、FC-2 协议层、FC-3 通用服务层、FC-4 映射层。如图 2-3 所示。

图 2-3 FC 层次结构

FC-0—物理层：有发送端、接收端、传输媒介组成。传输媒介可以使用单模或者多模光纤，不仅如此，屏蔽双绞线或同轴电缆等物理层常用的传输介质也可以作为其传输媒介。传输介质的多样性使其能够在各种应用环境下使用。且每条链路均支持 1Gb/s、2Gb/s、4Gb/s 和 8Gb/s 等多种传输速率，采用同步串行传输方式。

FC-1—编码层：定义了串行编码、解码和错误控制的传递协议，定义了 8B/10B 编码方案：光纤发送的信息应把一组 8 为代码转化为 10 为传输字符按位串行发送、光纤接收到的信息应分成 10 位一组，成为数据字符，然后解码成正确的 8 位码。并使用有序集来辨认数据帧的分界、传送下层状态信息和命令信息。

FC-2—协议层：定义了数据传输的通信原理和运行机制。包括若干类型服务、数据格式定义、序列摘除和重构、交换管理、分配地址、多重地址定义、多点广播控制和堆栈连接请求。并规定 FC 中数据传输的基本单元式最大长度为 21498 字节的可变数据帧，其中起始结束标示符、帧头、检验位共 36 字节，帧载荷最大为 2112 字节[15]，这也是下一小节会详细描述的 FC 帧结构。

FC-3—通用服务层：定义了一些列在一个节点上多通道端口的通用传输机制。如多条传送、多点传送、搜索组等服务。其中多点传输包括了针对所有节点的广播和部分节点的组播两种。

FC-4—映射层：负责在 FC 上映射一些其他的上层网络接口协议，使得它们能够在底层的 FC 网络上正常传输。

2.4.2 FC 消息帧结构

由于 FC 协议中所有的操作基础都是针对一个帧而言的，因此 FC 数据帧具有一个通用的形式。一个完整的 FC 数据帧包括了五个部分：SOF 帧起始标示符、Frame Header 帧头、PayLoad 载荷、CRC 循环冗余校验码和 EOF 帧结束标示符[16]。

图 2-4 为 FC 帧的完整结构。

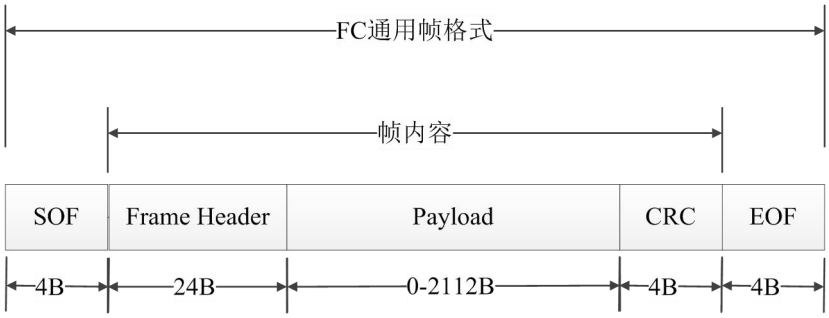


图 2-4 FC 数据帧完整结构

SOF：帧起始标示符，大小 4Byte，用于定义或识别各种类型的数据帧。

Frame Header：帧头，用于控制链路操作，转换设备协议，并且能对帧丢失情况以及传输次序进行检测。其帧头字段结构如表 2-1。

表 2-1 FC 帧头字段定义

字/bit	31~24	23~16	15~08	07~00
0	R-CTL 路由控制	D-ID 目的地址		
1	CS_CTL 类专用控制	S_ID 源地址		
2	TYPE 数据结构类型	F_CTL 帧控制		
3	SEQ_ID 序列数	DF_CTL 数据域控制	SEQ_CNT 序列计数	
4	OX_ID 发送方交换包标识符		RC_ID 接收方交换包标识符	
5	PARAMETER 参数			

PayLoad：数据字段按字大小进行排序，内容长度等于四字节的整数倍。

CRC：循环冗余度校验用来检验 FrameHeader 和 PayLoad 的完整性。CRC 字段是在传输编码前和接受解码后通过计算 FrameHeader 和 PayLoad 获取。需按字对齐。

EOF：帧结束标示符大小 4Byte，表明帧内容结束。

2.5 RAPIDIO 协议简介

RapidIO 技术最初由飞思科尔和水星网络共同研发的一项总线互联技术，最初研目的是为了作为用于作为总线系统连接处理器前端，RapidIO 不仅可以用于处理

器之间的互联， 还可以作为前端总线而使用[17]。

嵌入式系统所需求的是一套标准化的互连总线设计，低成本、高效率、通信方式多元化、支持 DMA 操作，支持以消息的形式进行数据交互、可以进行分布式处理和多 CPU 系统、支持多种拓扑结构，另外高稳定性和服务质量都是选择嵌入式系统总线的基本原则。RapidIO 在其制定之初即确定了几个基本原则：一是传输协议轻便简洁，尽量简单明了；二是层次分明，结构清晰，对于软件的开发制约越少越好；三是注重各芯片，各模块间的互联通信。

图 2-5 给出了 RapidIO 基本数据包的结构。

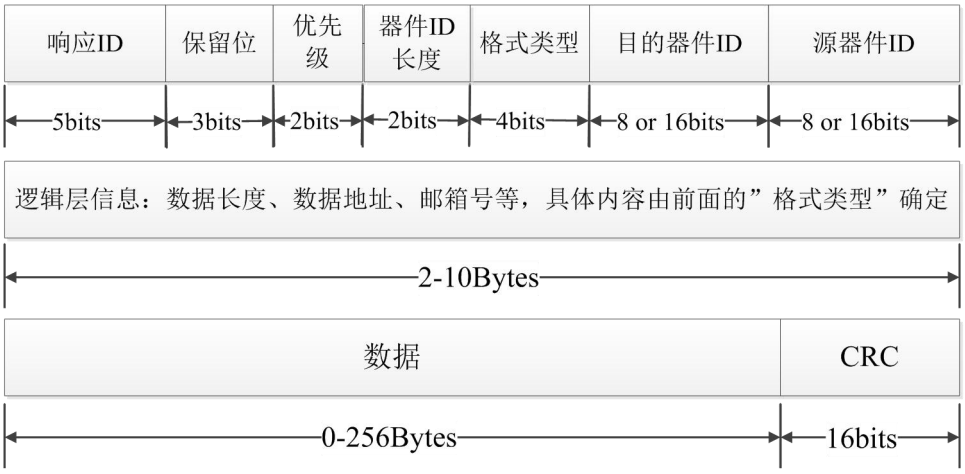


图 2-5 RapidIO 基本包格式

2.5.1 RapidIO 协议的体系结构

RapidIO 协议是依次为逻辑层、传输层和物理层，三层分级结构。

逻辑层定义了全部的协议操作和对应的数据包格式。逻辑层的主要作用是直接 IO/DMA（Direct IO/Direct Memory Access）和消息传递（Message Passing）。直接 IO/DMA 模式则是最简单实用的传输模式，直接 IO/DMA 使得主设备可以直接对从设备的存储器进行读写操作，直接 IO 系统包含了 6 种操作：NREAD—读操作；NWRITE—写操作；NWRITE_R—写操作，但需要等待响应；SWRITE—流写，针对对大数据量的 DMA 传送；Atomic—原子操作、Maintenance—维护包[19]。而消息传递模式与 IO/DMA 操作的区别在于它不需要知道从设备的存储器映射，它与以太网的传输方式类似，数据在设备中的地址由类似于 socket 的一个名为邮箱号的来确定[18]。从设备根据接受数据包的邮箱号，将数据保存到对应邮箱号的缓存区内。RapidIO 定义了两种不同消息事务包的数据包：第 10 类包（door bell）和第 11 类包，doorbell 适合传输长度为 8bit 或 16bit 的短消息，例如可以用于处理器的中断等，而第 11 类包消息数据由 16 个消息事务组成，每个消息事务的最大载

荷是256字节，因此支持传输载荷高达4096字节，适合高容量数据传输。而RapidIO协议中指出可以支持4个讯息信箱（mailbox），每个信箱可以装最多4个信件，这样发送方可以同时发送4个信件到一个目标信箱，从而完成对11类包的发送与接收[19]。

传输层完成了RapidIO数据包的路由转发、传输等功能。RapidIO采用了单一的公用传输层规范来适应不同的逻辑层和物理层，这也是RapidIO的一大优势，适应性强。RapidIO的路由和交换式通过每个终端设备的ID号，和交换器在它的每一个端口上的交换路由表实现，这和以太网交换类似，但区别在于每个端口的路由表需要在系统初始化时进行配置，这样的方法反而使得系统的路由去除了冗余，使得路由表容量不至于过大。终端对交换器的初始化和路由配置是由传输层包头中的一个字节HOP_COUNT实现，当交换器收到数据包时，首先判断HOP_COUNT的值，如果为0则说明数据包已经到达指定位置，交换器对此数据包的数据进行进一步操作，如果不是0，则交换器将HOP_COUNT值减1，然后按目的地址查询路由表转发[20]。

物理层定义了串行和并行两个模式，但是目前为止只有串行方式得到广泛使用，串行物理层定义了器件间的全双工串行链路。RapidIO的传输操作时基于请求和响应机制的。首先由发送端发出请求包，请求包被传送到交换器件，通过交换器件查找目的地址这个完整的请求包被转发到接收端。接收端完查看请求后发送响应包，经过交换器件传回发送端，相当于一个建立链接的过程。不仅如此，RapidIO协议传输过程中可靠性十分高，每一个数据包的传送过程中，对应端口的物理层上都会产生一个大小为4字节的控制符号包，用于描述数据包的传送状态，如未收到、被接收、需重传等[21]。通信双方均可以通过这个控制符号包实时的监控对方状态，以达到通信状态可控。

2.6 本章小结

本章首先介绍了本仿真系统的操作系统VxWorks，介绍了VxWorks的优越性，这些优越性正是本仿真系统选取它的重要理由；其次为了满足系统的功能和性能要求，介绍了本系统选取的两个最为关键的器件CPU—MPC8640和交换机（芯片）CPS-1616；最后介绍了本系统消息部分所用到的两个重要协议FC协议和RapidIO协议。

第三章 嵌入式仿真系统的总体设计

3.1 系统需求分析

该系统用于对航电机载集成处理系统进行仿真模拟，仿真系统主要由处理器板、交换板、背板、后走线板和电源这五部分组成。

仿真系统应具备模拟真实机载集成处理系统的全部能力，该系统主要用于实验室环境，可对系统的前期验证提供较完整的仿真环境，该系统也可以作为一个仿真平台，通过部署上层的应用软件可以模拟出所有的机载功能或者其他用户需求的功能，从而可以对机载系统的联调进行仿真验证以及模拟使用。因此设计主要集中在各硬件互联，模块内、模块间通信模式的设计。

3.2 系统架构设计

图 3-1 是系统的总体构架图：

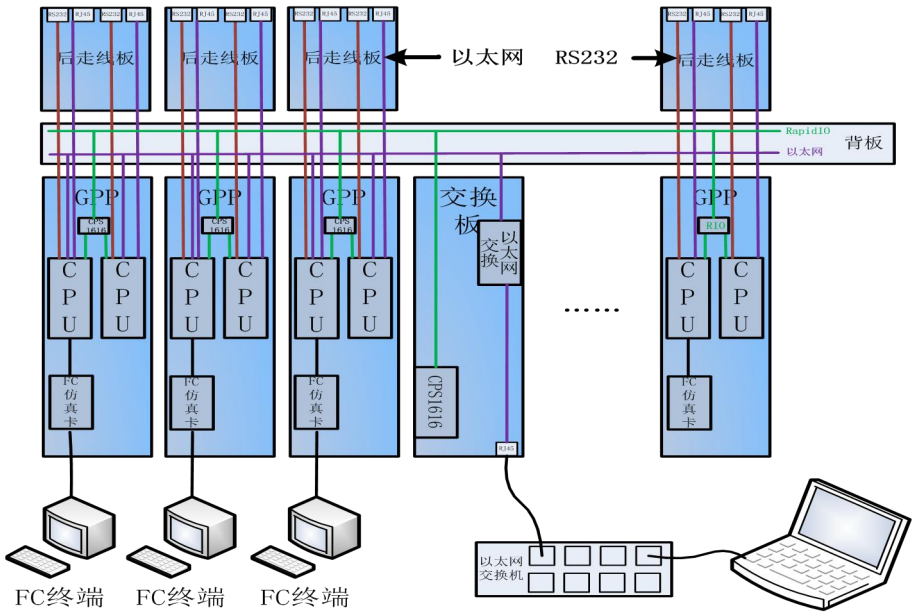


图 3-1 系统总体框图

处理器板（GPP）主要用于模拟实现通用处理模块的功能，整个系统可根据实际需求以提供多套处理器模块，每套处理器模块针对一个外部应用，而每套处理器模块的 1 号 CPU 都有一个 XMC 接口，可用于扩展 FC 仿真卡（项目中使用的是成电光信科技股份有限公司所研发的 FC 仿真卡），以实现与外部 FC 终端进行

数据交互处理。

交换板上的主要器件有两个交换器，一个以太网交换芯片，一个 RapidIO 交换芯片，主要用于各处理器板上 CPU 之间的以太网数据交换和 RapidIO 数据交换。

背板主要作为模块与模块之间的线路连接和交互平台。

后走线板主要用与将对应处理器板的接口向后引出，引出的网口和串口与计算机相连，可以进行对单板的调试。

处理器模块和交换模块作为本系统最重要的两个模块，系统所需得到的功能主要集中反映在处理能力需求和通信需求。

处理能力多数取决于硬件，如 CPU、内存等，本文主要针对软件部分，因此暂不赘述。而在通信需求上，合理高效的设计也在很大程度上利于处理能力的提升，通信需求又存在模块间和模块内，以及系统与外部设备的通信互联上。

模块间的互联通信方式主要有两种：以太网互联和 RapidIO 网络互联，每个处理器模块间，以及处理器模块和交换模块间均可以通过以太网通信和 RapidIO 通信。而处理器板模块内部两个 CPU 间的互联通信也同样是有以太网互联和 RapidIO 互联。整体系统与外部设备的通信方式则以串口或者网口进行通信。

3.3 系统模块互联

本系统模块互联从通信方式上分为两部分：千兆以太网互连和 RapidIO 网络互连。

3.3.1 千兆以太网互联

每个处理器板上的两个 CPU 配备了一个以太网 PHY（双通道），也就是两路千兆以太网信号。本系统的以太网互连关系如图 3-2 所示。

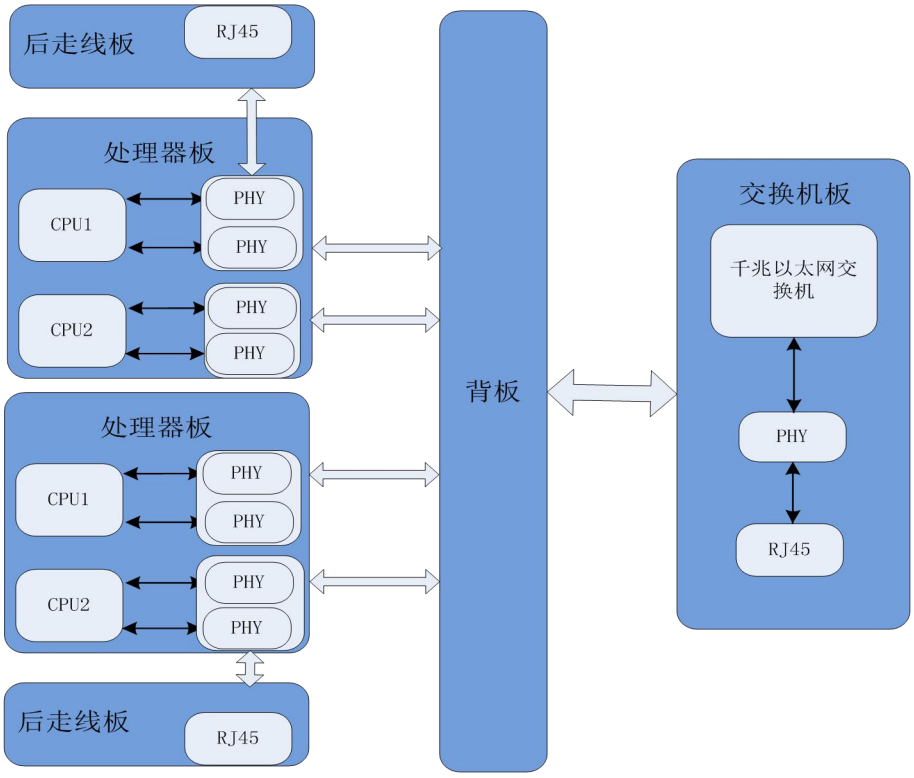


图 3-2 以太网互连关系图

由图 3-2 所示，每个 CPU 分别有两路以太网连线，其中一路接入背板，再由背板引出到交换板与交换板上的千兆以太网交换芯片相连，而交换板内以太网交换机连接至交换板上的一个网口；另外一路直接由后走线板引出到一个网口上。

这样的设计有效的实现了整个系统各模块间，各模块内部，以及外部设备与系统内部某个特定 CPU 的以太网互联通信。

模块内部的两个 CPU 的以太网通信通过背板，再经由交换板上的以太网交换芯片路由；而模块间的两个或者多个 CPU 的以太网通信也是同样的线路经过交换板的交换芯片路由完成；外部设备与系统内部通信方式则有两种，可以通过交换板上的网口，对整个系统或者某个模块注入激励，对于单个 CPU 的通信也可经由此种方式，这样的设计可减少了外部线缆的连接数量，也实现了外部对所有的处理器板的整体可控，方便进行整体调试。另外就是外部设备可以通过后走线板上的网口对特定的单个 CPU 进行通信，此通信链路主要是针对某个特定应用对单个 CPU 进行调试使用。

3.3.2 RapidIO 网络互连

本系统的 RapidIO 网络互连关系如图 3-3 所示：

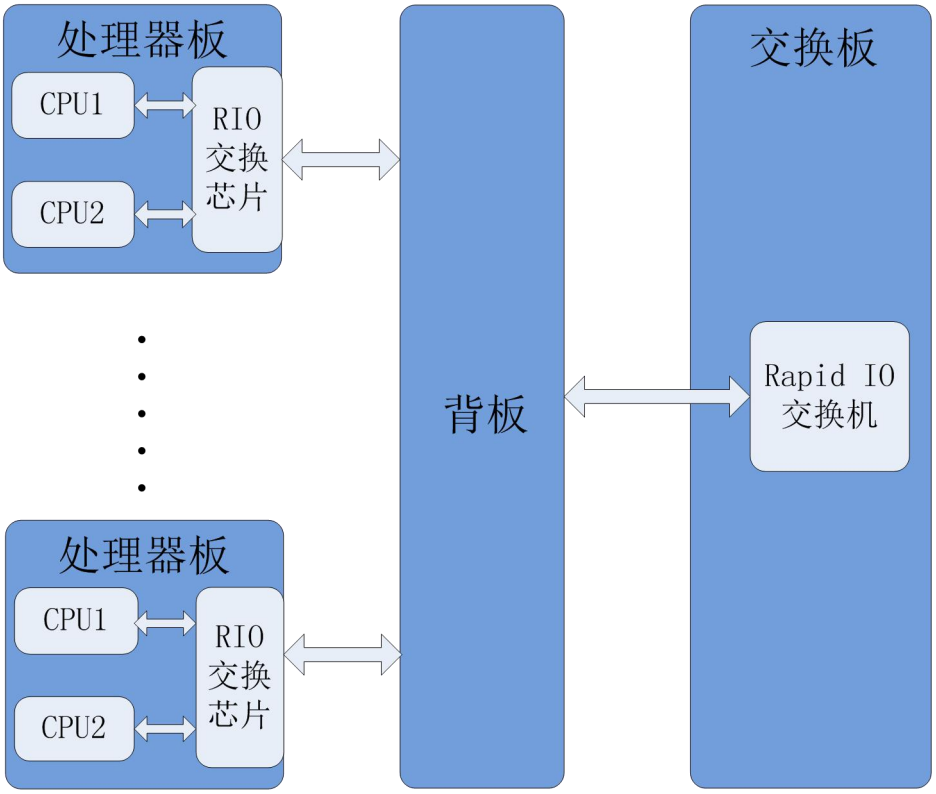


图 3-3 RapidIO 网络互连关系图

如图 3-3 所示，系统中所有的处理板上都有一个 RapidIO 交换芯片，交换板上也有同样的一个 RapidIO 交换芯片，本系统中把 CPS-1616 作为 RapidIO 交换机。

每块处理器板上的两个 CPU 都通过 RapidIO 总线与本处理器板上的 RapidIO 交换芯片相连，再由交换芯片通过 RapidIO 总线引出到背板，背板再连接交换板上的 RapidIO 交换机。

对于每块处理器板上的 RapidIO 交换芯片存在三条路由，假设为 A，B，C 三条，其中目的地址为 CPU1 的 RapidIO 数据走路由线路 A，目的地址为 CPU2 的 RapidIO 数据走路由线路 B，其余的目的地址均通过路由线路 C 将 RapidIO 数据路由至交换板，再通过交换板上的路由表查询转发。也就是说，当某个处理器板内部两个 CPU 通信的话，RapidIO 数据只经过该处理器板内部的 RapidIO 交换芯片一次路由，实现了板内通信。当多个处理器板间的 CPU 要进行通信时，源地址 CPU 会将数据包发至所在处理器板的 RapidIO 交换芯片，查询路由表后通过 C 线路，发送至交换板的 RapidIO 交换芯片，此时再通过交换板的 RapidIO 交换芯片查询路由转发只目的 CPU 所在的处理器板的 RapidIO 交换芯片，最后经过三次路由转发至目的 CPU，以实现板间通信。

3.4 硬件平台概述

3.4.1 处理器板概述

处理器板是本系统的核心模块，每一块处理器板配备了两片 Freescale 的 PowerPC 架构处理器 MPC8640，用于模拟实现通用处理模块的全部功能。图 3-4 给出了处理器板的结构图。

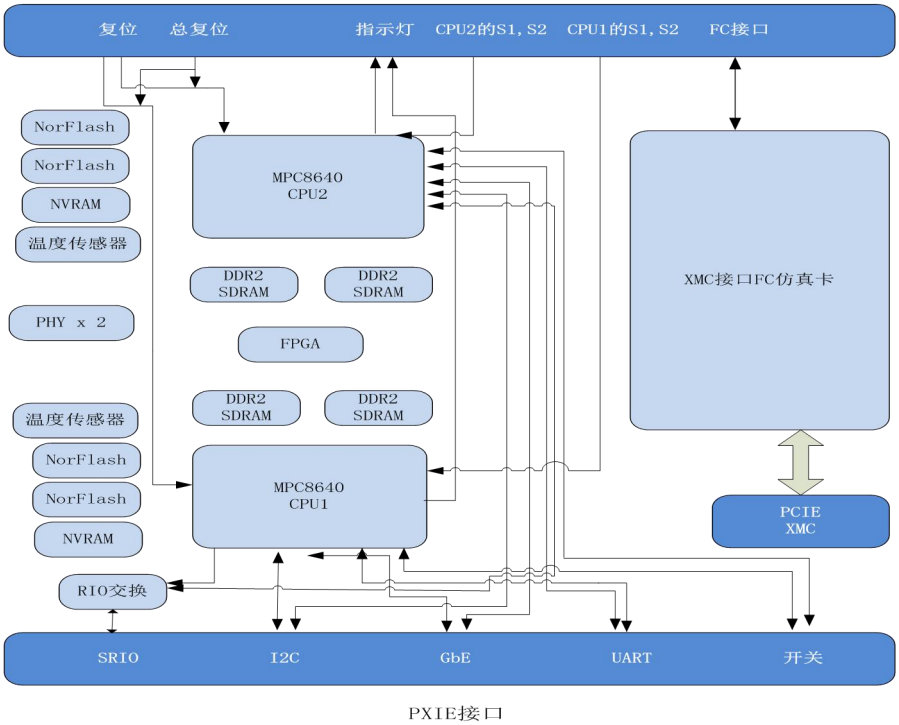


图 3-4 处理器板结构图

如图 3-4 所示，都片处理器都配备了以下外部资源：DDR2 SDRAM—256MB x 2、FLASH—128MB x 2、NVRAM—128KB x 2、以太网 PHY（双通道）x 1、温度传感器 x 1。

温度传感器可通过 I2C 实时读取温度值；处理器板的前面板安装有两个复位开关，分别用于两片 CPU 的复位。同时还有一个总复位开关，可以对此处理器办板的两个 CPU 同时复位；处理器板的 1 号 CPU 配备了一块 XMC 接口的 FC 仿真卡，用于收发 FC6 网络消息；前面板共有两组开关量信号 S1，S2 分别连接两个 CPU；每片 CPU 各有一路串行 RapidIO 信号接入到处理器板上的 CPS-1616 交换芯片上，此芯片再通过背板接入到交换板上的 RapidIO 交换芯片；并且两片 CPU 各有两路前兆以太网信号接入背板，其中一路接入到交换板的以太网交换芯片，另外一路直接从后走线板引出；为了处理器的调试使用，每片 CPU 还各有一路有 UART 接入背板，由后走线板引出。

3.4.2 交换板概述

交换板是各处理器板之间桥梁，处理器板之前的网络数据交换、RapidIO 数据交换都要经过处理器板，外部网络与各处理器板之间的通信也可通过交换板实现。

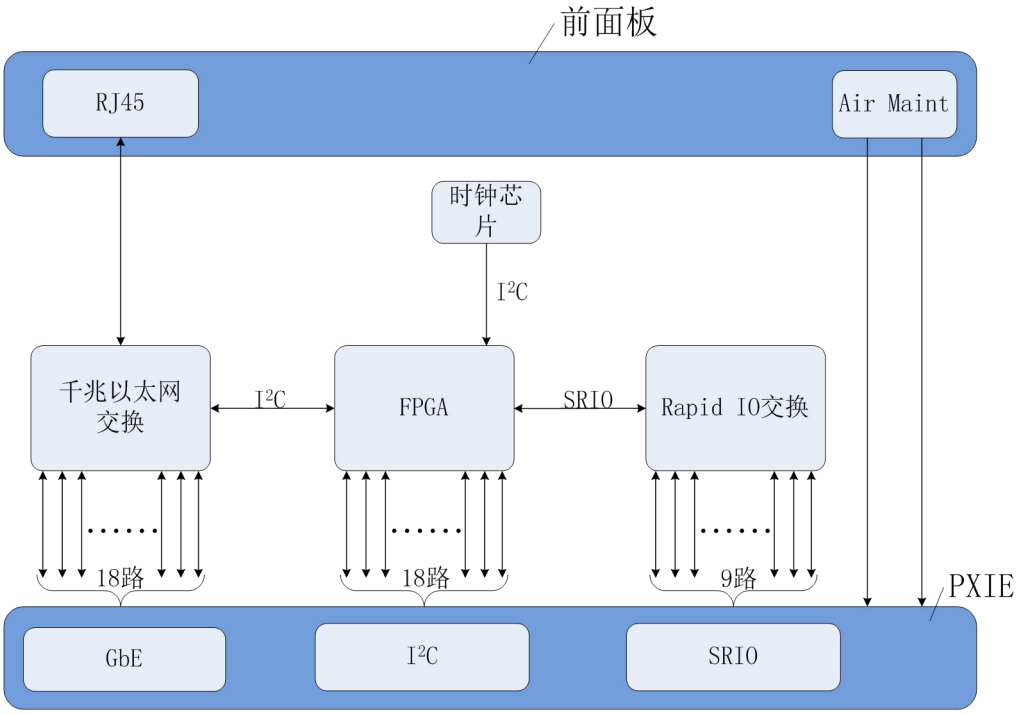


图 3-2 交换板结构图

如图 3-5 所示，交换板中的千兆以太网（GbE）交换机选用的是 BCM5396，该交换机无需外部配置即可完成以太网的二层交换，且同时支持 16 端口的交换，完全满足调试需求，以太网交换机借助与背板与各处理器板相连接；交换板的前面板有一个以太网接口，其内部与以太网交换机相连，用户通过此接口可以同任一处理器进行通信与调试；作为 RapidIO 交换机的 CPS-1616 芯片，只需简单配置即可使用，可同时支持 16 个端口，最高可达 80Gbps 的峰值吞吐量，同样借助与背板与各处理器板相连接；前面板有两路 Air 和 Maint 输入开光量由所有 CPU 共同使用；针对交换板上的时钟芯片，提供日历时钟（年/月/日/时/分/秒），该日历时钟信息通过独立的 I2C 总线分布到各处理器板的每个 CPU 上，时钟芯片的初始时钟在设备出厂时进行配置的，可通过 PC 人为设定；该系统还提供另外一种时钟—RTC 时钟（42 位无符号数），该时钟由充当时钟服务器的 CPU 设置，发送给交换板的 FPGA，交换班的 FPGA 再分发给各个 CPU 的本地 RTC，从而实现同步。

3.5 本章小结

本章针对航电机载集成处理系统需要解决的问题，对嵌入式仿真系统进行了总体上的设计。首先叙述系统整体功能及性能需求，对系统的架构提出了解决方案；接着从整个系统的通信需求出发，对主要的通信方式进行了介绍和设计方案的描述；最后对本系统的硬件平台进行了简述，两个模块处理器板和交换板进行的结构和功能。

第四章 嵌入式仿真系统的关键技术与实现

仿真系统的关键技术主要有以下几点： RapidIO 驱动程序、利用 RapidIO 高速通信承载 FC 消息以实现 FC 消息的交换。本文主要负责除硬件逻辑外的其他部分，而对于各个模块、各 CPU 间的通信机制在前文中有描述，所以下面分别对 RapidIO 驱动程序以及利用 RapidIO 承载 FC 消息以及 FC 消息的交换的设计思路和具体实现方法进行详细阐述。

4.1 RapidIO 驱动开发

在嵌入式系统中，有一个最小系统的概念，所谓最小系统，即最精简，能满足最基本需求，对于每个项目，每个工程，最小系统的定义和功能都有不同，由发开人员自行定义的。编写板级支持包 BSP（Board Support Package）的第一个目的就是使目前的嵌入式系统成为最小系统。因此在做硬件驱动开发之前，首先编写 BSP。

板级支持包通常是针对具体的硬件平台，用户所编写的启动代码和部分设备的硬件驱动程序的集合，它所实现的功能包括初始化、驱动部分设备。BPS 的概念只针对嵌入式操作系统，意义上类似于 DOS、Windows、UNIX 等依赖的 BIOS 引导[22]。

4.1.1 BSP—板级支持包

在目标系统上电后，首先执行的代码就是 BPS，它主要是用于加电后初始化目标机硬件、初始化操作系统以及提供部分硬件的驱动程序。

1. 初始化：所谓初始化就是指上电复位开始直到 VxWorks 开始初始化用户应用时（即系统执行到 userAppInit 函数处）的一段时间内系统执行的过程。这个过程主要包括了三个部分的工作[23]。
   i. CPU 初始化：初始化 CPU 的内部寄存器。
   ii. 目标机初始化：初始化控制芯片的寄存器、I\O 设备寄存器，为整个软件系统提供底层硬件环境支持。
   iii. 系统资源初始化：为操作系统以及系统的正常运行做准备，进行资源初始化。
2. 使 VxWorks 能够访问硬件驱动程序，这主要是 BPS 包含部分必要的设备驱动程序和相关设备的初始化操作。

3. 在 VxWorks 系统中，集成了与硬件相关的软件和部分硬件无关的软件。

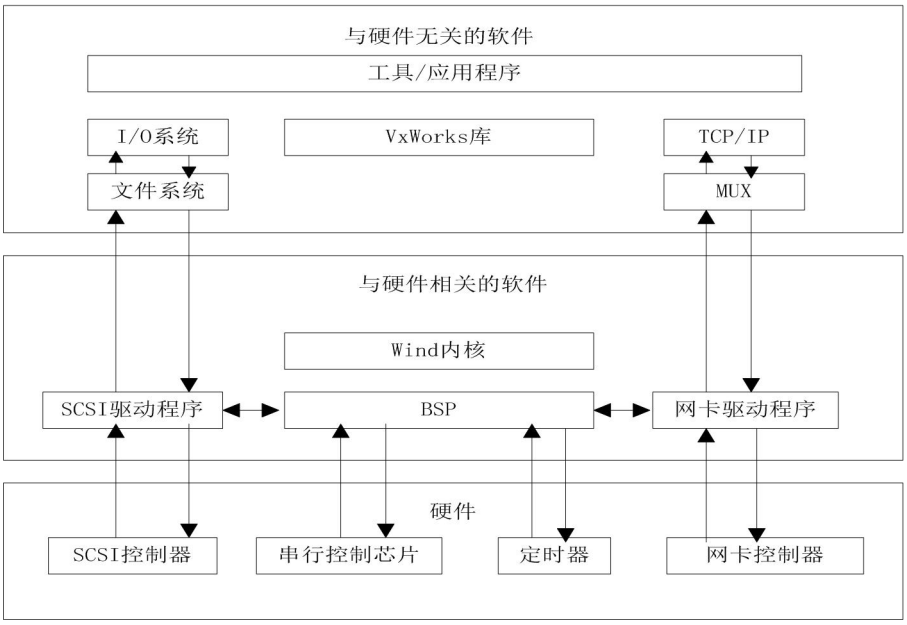


图 4-3 VxWorks 系统层次结构

从图 4-1VxWorks 系统层次结构中可以看出，BSP 不是一个设备驱动程序，它并不能直接访问硬件设备，BSP 只能运行在指定设备的硬件环境中。

在本系统中最初设计的最小系统主要是针对网口，串口功能进行了添加。

针对 VxWorks 系统，最小系统的设计由 Boot Loader/Bsp Prioject 完成，在目标工程中，在现有的 BSP 包内，根据本系统需求，进行修改编码。

在 VxWorks6.7 下，根据现有的代码进行修改，并在硬件描述设备表中，添加或删除设备等，最后编译形成一个二进制文件—bootrom，我们将此文件烧录到 FLASH 中，每个处理器板的每个 CPU 都具备了最小系统。

之所以添加网口，是为了通过网口加载程序，并且通过网口通信进行调试。而串口对于 VxWork 操作系统来说，起到显示输出的作用，只有串口通信功能存在时，才能对系统的各状态进行查看，调试。

因此，此最小系统的设计是为了后续系统开发所做的准备工作。

4.1.2 RapidIO 驱动

嵌入式系统中的驱动程序，是直接控制设备操作的那部分程序，并且也是上层的一系列软件接口，驱动程序的几个基本功能有：对设备进行初始化、打开设备操作、关闭设备操作、从设备上接收数据并提交给系统、把数据从主机上发送

给设备和对设备进行控制操作。

VxWorks 设备驱动程序是直接控制下层硬件设备的一个上层软件接口，同时对上层应用程序提供接口函数，这样编写出来的应用程序部依赖于硬件，增强了应用程序的可为维护性和可移植性。

本系统需要设计的驱动众多，由于本文重点描述的是系统中 RapidIO 传输 FC 模块，所以主要介绍 RapidIO 驱动的设计。图 4-2 为 RapidIO 驱动流程图。

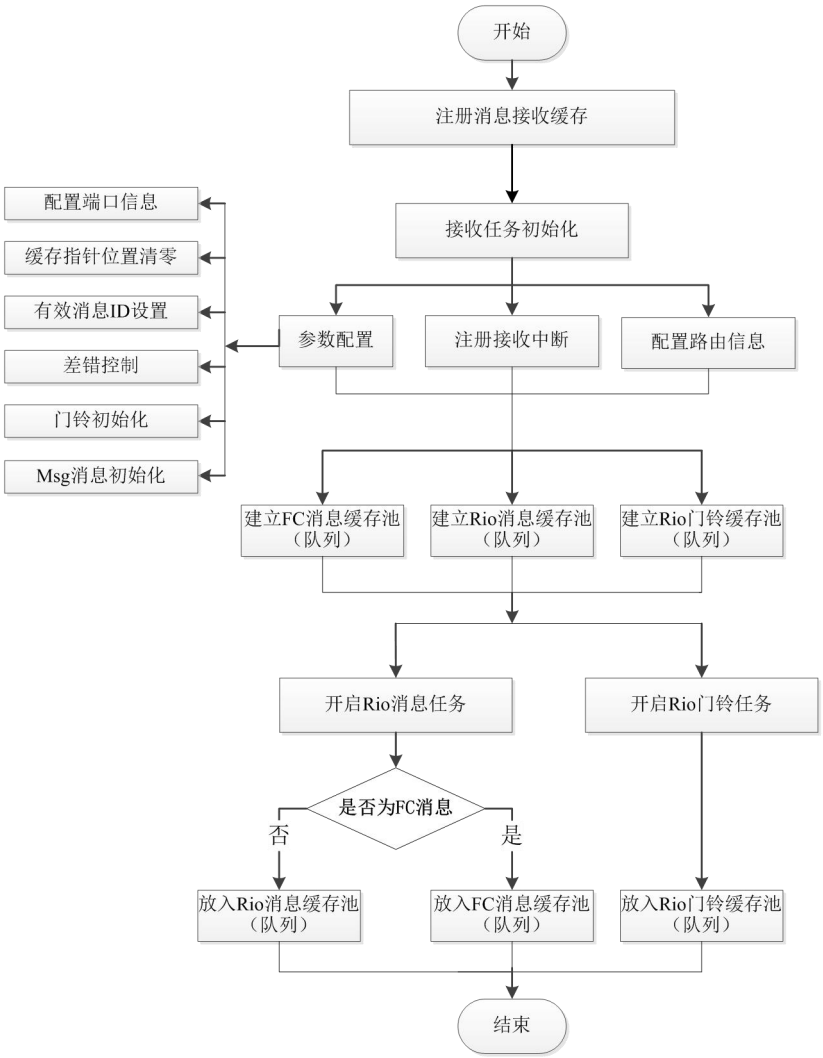


图 4-2 RapidIO 驱动流程图

RapidIO 驱动做的是初始化 RapidIO 模块，初始化的工作主要有三方面，一是注册接收缓存，二是对接收任务的初始化，三是开启接收任务。

首先注册了 RapidIO 驱动模块结构体的缓存。

RapidIO 驱动模块结构体如下所示：

```
typedef  struct  RioDriver
{
    INT32  hRio;
    INT32  state;
    LIST_MANAGE_S*  fcMsgList;
    LIST_MANAGE_S*  fcMsgListPool;
    LIST_MANAGE_S*  rioMsgList;
    LIST_MANAGE_S*  rioMsgListPool;
    LIST_MANAGE_S*  rioDoorbellList;
    LIST_MANAGE_S*  rioDoorbellListPool;
    RECV_RIO_MSG_STR*  pRecvMsg;
    OSUI_EVENT_ID  msgRecvEvent;
    OSUI_COUNT_SEM_ID  doorbellRecvSem;
    OSUI_COUNT_SEM_ID  msgRecvSem;
    UINT8  recvMsgTaskStatus;
    UINT8  recvDoorbellTaskStatus;
    int  recvMsgTaskID;
    int  recvDoorbellTaskID;
}FC_RIO_DRIVER_S;
```

此结构体主要用于存放 Rio 状态信息、FC 消息缓存池信息、Rio 消息缓存池信息、Rio 门铃缓存池信息、数据接收缓存、各类消息接收计数信号量、以及接收线程的状态等，并注册数据接收缓存，根据最大的消息类型，缓存大小为 4096 字节。

接收任务初始化中做了注册中断和创建事件的工作，保证每次接收到消息时，也就是发生事件，任务处于堵塞状态，这时由于快速响应的中断，在处理完消息后，释放了事件，任务继续运行。这样的机制保证了接收缓存不会堵塞，也不会对系统造成额外的负重。还对 RapidIO 进行了一系列的参数配置，包括有配置端口信息，其中端口信息又含有寄存器基地址值、大小值、输入输出窗口大小等；将缓存读、写指针位置均清零；设置能够接受的有效 ID；差错控制主要是开启错误信息中断；对门铃和 Msg 消息进行初始化；配置路由信息，通过对 CPS-1616 芯片寄存器的读写进行路由信息的配置。

在初始化过程中，建立了三个缓存池，分别用于存储 FC 消息，RapidIO 消息，RapidIO 门铃消息。每个节点的大小均不一样，FC 消息—2048 字节，RapidIO 消

息—4096 字节，RapidIO 门铃—512 字节。每种消息都是以队列的形式存储，在接收的时候，会弹出队列头，将新接收的消息放在队列尾部。

最后开启 RapidIO 消息接收任务和 RapidIO 门铃接收任务，这里由于 FC 消息是承载于 RapidIO 消息之内的，因此没有开 FC 消息的接收任务，只是在 RapidIO 消息接收任务中，会判断消息的类型，判断方法在下文 4.2.2.2 节中描述，然后放入各自的缓存池。每种消息都是以队列的形式存储，所以在接收的时候，会弹出队列头，将新接收的消息放在队列尾部。

4.2 Rapidio 传输 FC 模块

此模块的功能主要是利用 RapidIO 承载 FC 消息并利用 RapidIO 交换芯片实现 FC 消息的交换，与以往的类似设备不同的是，FC 消息的交换通常是通过 FC 总线网络交换机实现的，本系统利用 CPS-1616 交换芯片代替了 FC 交换机。

本仿真系统与采用 FC 交换机的航电机载集成处理系统相比，通用的航电机载集成处理系统的任意模块之间，或者与外部设备的通信都通过 FC 交换机进行数据转发，而本系统如果是在系统内部各模块的通信可以直接通过 RapidIO 交换芯片实现，如果系统的内部模块需要与外部设备进行通信，则必须先把数据通过 RapidIO 交换芯片传递给 CPU 板上的 FC 仿真卡，然后通过仿真卡再把数据传送出去。

4.2.1 模块数据流

由于此系统避免使用了昂贵的 FC 交换机，取而代之的是 RapidIO 交换，因此首先分析带有 FC 交换机的航电机载集成处理系统中的系统数据流，再对比利用 RapidIO 承载 FC 数据的系统数据流。

采用 FC 交换机的航电机载集成处理系统中，任何连个模块之间的消息通信都必须通过 FC 交换机进行数据转发。两模块系统通信如图 4-3 所示。

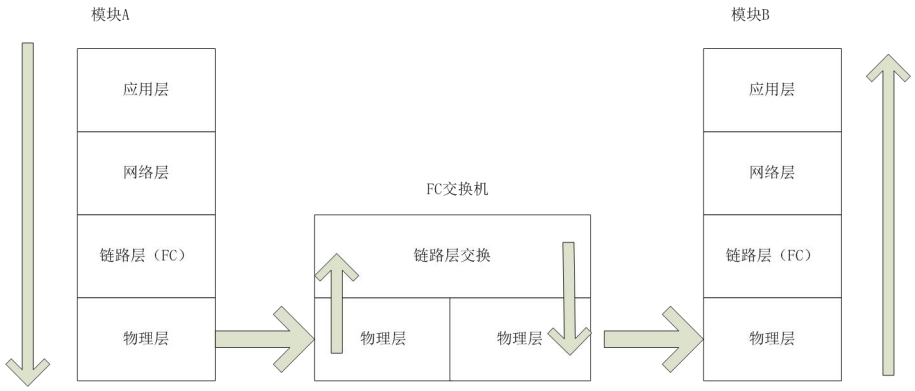


图 4-3 带 FC 交换机的系统数据流

在图 4-3 中，如果模块 A 要与模块 B 进行消息通信，会产生这样的数据流：模块 A 的应用层产生应用消息，然后调用网络层提供的服务，把消息传递给网络层，网络层接收到应用层的消息后，对消息进行封装，添加网络头和尾，网络层将封装好的消息成为网络包，将网络包递交给链路层（链路层是 FC 协议）。链路层把网络包再进行封装，添加 SOF、FC 帧头、CRC 和 EOF 之后，把数据传递给物理层。

FC 交换机实现的功能是进行 FC 帧的交换。交换的最小单位为 FC 帧，FC 交换机值识别 FC 帧头，并不对净荷进行解析。

在模块 B 的接收端，进行相反的操作。对 FC 帧一次进行解封装，最后把应用数据传递给模块 B 的应用层进行使用。

链路层和物理层与 FC 协议体系的对应关系如图 4-4 所示

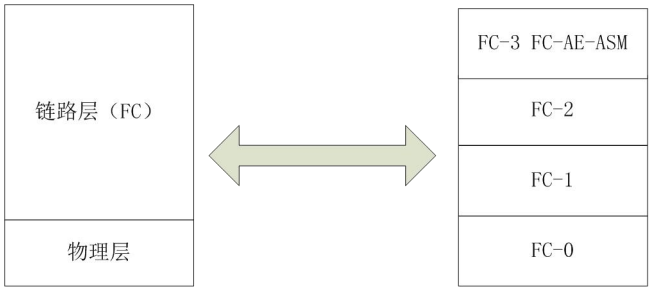


图 4-4 FC 协议体系对照图

如图 4-4 所描述，FC 机载网络中模块的结构层次中，链路层和物理层对应 FC 协议体系的 FC-0 至 FC-3 层。

而在本系统中，模块间的数据流和带有 FC 交换机的航电机载集成处理系统的数据流有所不同。

本系统内部功能节点之间的数据通信如图 4-5 所示。

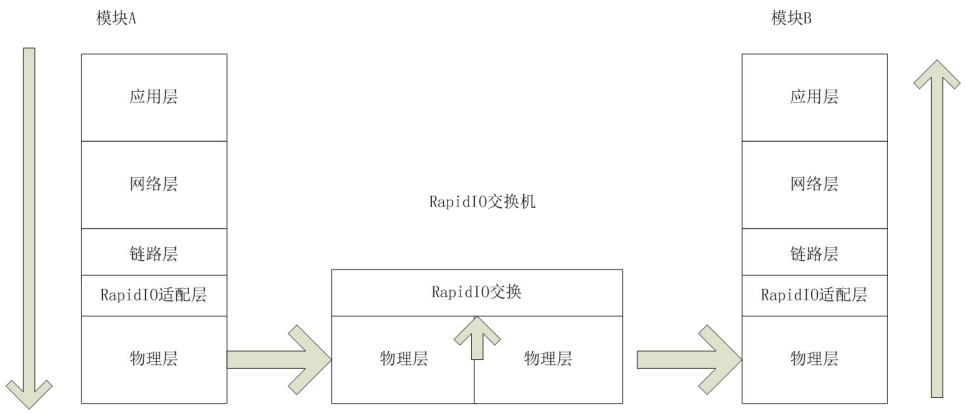


图 4-5 本系统内部模块消息通信

对比图 4-3 和图 4-5，在图 4-5 中，增加了 RapidIO 适配层。这是因为在本仿真系统内部，模块之间进行消息通信时，并没有 FC 交换机，取而代之的是 RapidIO 交换芯片。为了通过 RapidIO 芯片实现 FC 数据的交换，就必须增加一个 RapidIO 适配层，把 FC 数据封装在 RapidIO 包中。

链路层和 RapidIO 适配层的详细组成如图 4-6 所示：

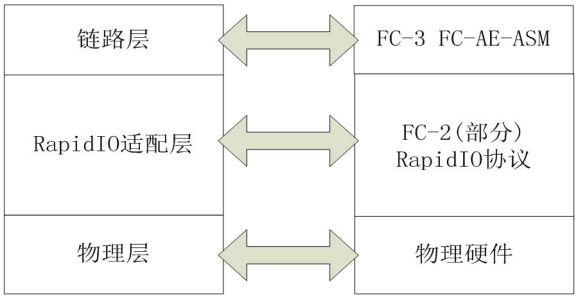


图 4-6 链路层与 RapidIO 适配层组成

由图 4-6 中可以看出，链路层实现了 FC-3 层的功能，RapidIO 适配层实现了部分 FC-2 层的功能和 RapidIO 协议。

本系统内部功能模块与外部设备模块的数据通信如图 4-7 所示。

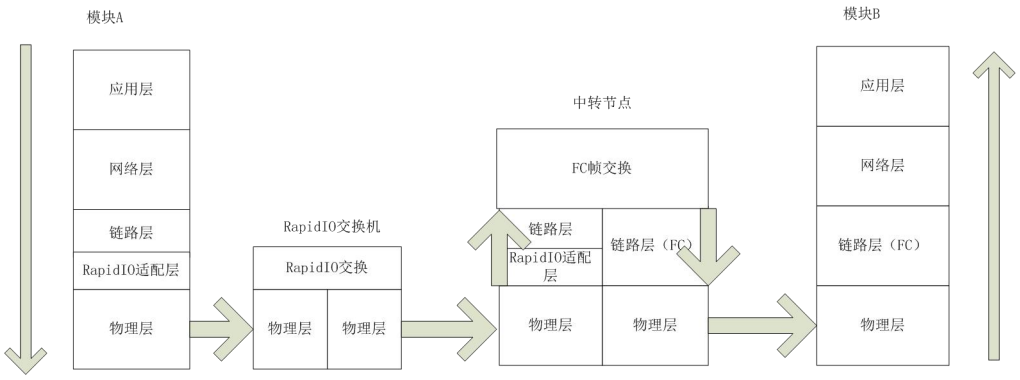


图 4-7 本系统内部模块和外部模块通信

模块 A 在系统内部，而模块 B 为外部设备的通信模块。若 A 要与 B 进行通信，那么首先模块 A 产生的数据通过 RapidIO 交换机发到中转接点，然后中转接点再把数据发到外部模块 B。

根据此设计思路，我们需要解决的问题细化到 RapidIO 适配层的实现，中转接点的实现，下面给出各部分的具体实现方案。

4.2.2 RapidIO 适配层实现

首先，RapidIO 实现数据传输的传输原理如图 4-8 所示：

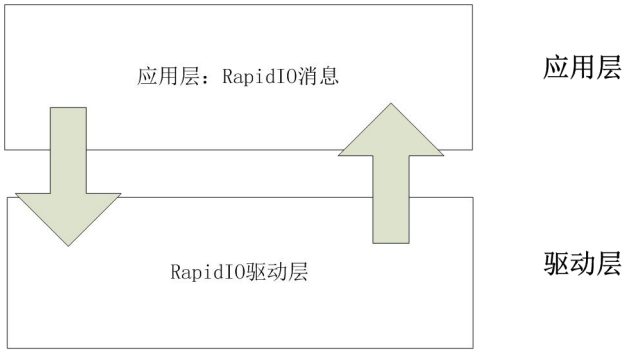


图 4-8 RapidIO 数据传输模型

在 RapidIO 模型中，分为两层：应用层和驱动层。应用层负责产生和处理 RapidIO 消息。驱动层负责与硬件交互。发送时，应用层产生的 RapidIO 消息递交给驱动层。接收时，驱动负责从硬件接收数据，然后上传给应用层。

而要通过 RapidIO 传递 FC 帧，那么在图 4-5 的模型中，应用层数据不仅仅有 RapidIO 数据，同时还有 FC 帧。因此，必须通过一个中间层来适配应用层的需求。所以新的层次模型如图 4-9 所示。

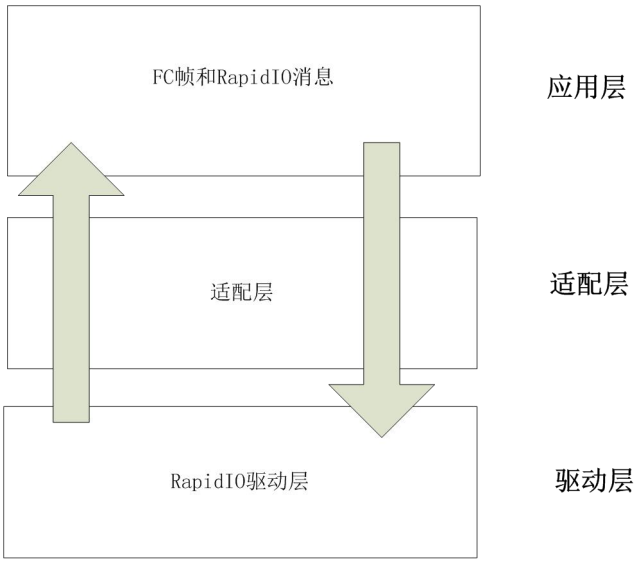


图 4-9 带适配层的 RapidIO 数据传输模型

对比图 4-8 和图 4-9 可以发现，在图 4-9 中，也就是本仿真系统中的应用层增加了 FC 帧，而图 4-8 中只有 RapidIO 消息。整体层次上，在应用层和驱动层之间增加了适配层。

适配层主要功能是为了适配底层驱动和上层应用，实现 FC 帧和 RapidIO 消息的统一，实现消息的封装，将 FC 消息承载到 RapidIO 消息中去；在发送时，必须解决的问题是目的设备号，因为在 FC 网络层有一个地址空间，实现时必须实现 FC 地址空间到 RapidIO 地址空间的转化；而从整体性能和层次上分析，RapidIO 层和 FC 层并不是完全独立的，RapidIO 协议设计时也没有考虑到映射其他协议的问题，所以，我们必须在 FC 数据中做一些特殊标志来区别 FC 数据和 RapidIO 数据。

4.2.2.1 地址转换和数据包封装

在发送，会涉及到 FC 端口 ID 到 RapidIO 的地址转换问题，首先适配层提供的发送 API 接口为：

INT32 ExternRioSendMsg(UINT8 destTarget, UINT8 mailbox, UINT8 priority,
UINT8 packetDataType, char* packetDataBuf, UINT16 packetDataLen);

其中第一个参数 destTarget 即表示为目的地址。为此，设计了一个配置方案表，如表 4-1。表中规定了 FC 的 portID 和 RapidIO 地址的映射关系，指定此关系的原则是简单明了。将此配置方案表文件存储成 XML 文件的格式，可以通过 PC 机进行编辑，通过以太网传输到系统内部的每个模块，每个模块把 XML 内容保存到本地的 FLASH 芯片中。

表 4- 1 RapidIO 地址和 FC 地址转换表

FC 地址	0x10001	0x10002	0x10003	0x10004	0x10005
Rio 地址	1	2	3	4	5
FC 地址	0x10006	0x10007	0x10008	0x10009	0x1000a
Rio 地址	6	7	8	9	10
FC 地址	0x1000b	0x1000c	0x1000d	0x1000e	0x1000f
Rio 地址	11	12	13	14	15
FC 地址	0x10010	0x10011	0x10012	其他	
Rio 地址	16	17	18	0xff	

表 4-1 中给出了 18 组 RapidIO 地址和 FC 地址的对应关系，分别代表了 9 块处理器板上的 18 块 CPU。而对于其他 FC 地址，或者表中没有的地址，也就是外部的 FC 地址，由于内部模块要发送至外部 FC 应用需要通过中转节点，因此外部的 FC 地址均映射到相应处理板的中转节点的 Rio 地址上。

此表可以通过编辑配置文件进行自行设计，并且可以同时支持多个配置文件，配置文件由一个配置文件结构体所控制，结构体如下所示：

```
typeded    struct
{
    unsigned    int    type;
    unsigned    int    length;
    char    version[16];
    TIME_S    time;
    unsigned    int    sum_check;
    unsigned    int    cfg_num;
    unsigned    int    cfg_current;
    unsigned    int    addr;
    unsigned    int    portID;
    CFG_INFO    *pCfg;
}CFG_PORTIDADDR_S;
```

上述结构体中有配置文件类型，有效配置文件长度，生成日期，部分字节相加校验和，配置方案个数，当前所选配置方案号，以及 RapidIO 广播地址和 FC 广

播端口 ID，最后是配置方案表的内容。在 RapidIO 地址和 FC 地址进行转换时，首先应该判断是否为广播地址，若不是再进行查表工作，查找方式直接利用循环查找，这是由于表内内容并不是很多，利用最简单明了的查找方式即可。

适配层需要对 FC 消息进行封装，也就是利用 RapidIO 消息承载 FC 帧。针对第二章，2.4 节的对 RapidIO 协议的描述，RapidIO 提供的第 11 类包可以用于多事务消息发送，最大可传输载荷为 4096 字节，它的包格式如图 4-10 所示。

图 4-10 RapidIO 第 11 类包格式

而在第二章，2.4.2 节中，通过对 FC 帧格式的描述所知一个完整的 FC 帧最大长度为 2148 字节，小于 RapidIO 所支持的第 11 类数据包，因此完全可以将 FC 帧作为 RapidIO 第 11 类包的载荷封装到包内，如图 4-11 所示：

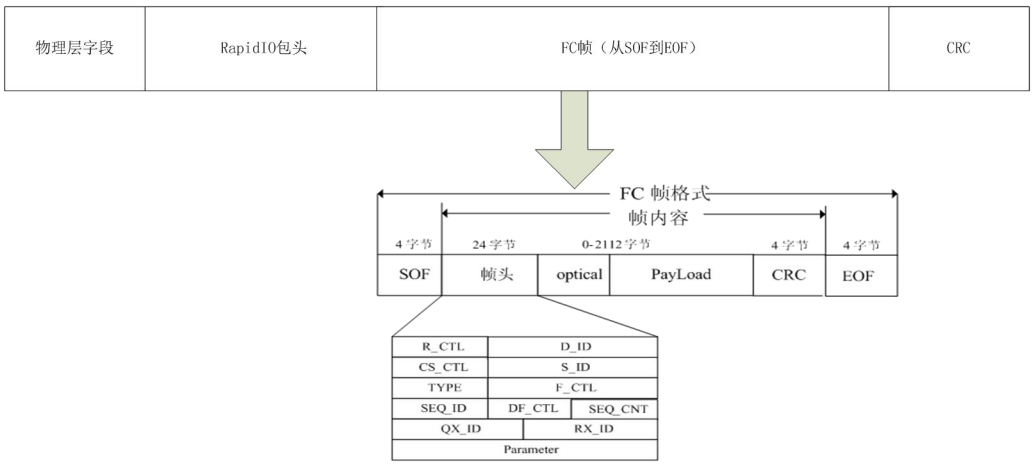


图 4-11 RapidIO 承载 FC 数据

4.2.2.2 数据类型的判断

在接收的过程中，由于上层数据包括了 FC 帧和 RapidIO 数据。所以在适配层必须提供两个缓存，分别用于存储 FC 帧和 RapidIO 数据。适配层缓存如下图 4-12

所示。

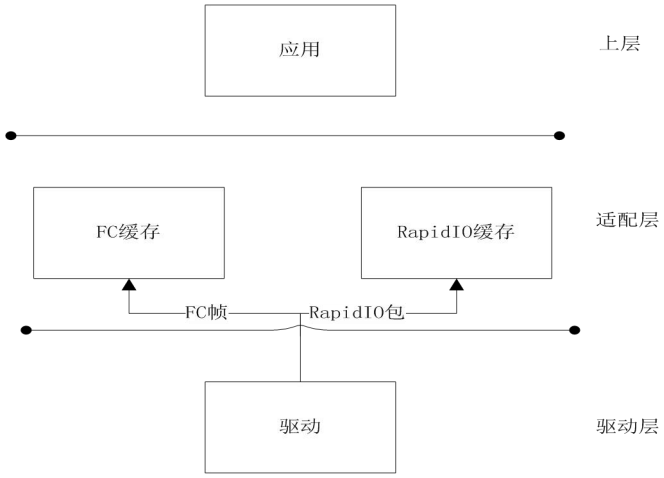


图 4-12 适配层缓存构成

在适配层中，接收消息流程图如图 4-13 所示。

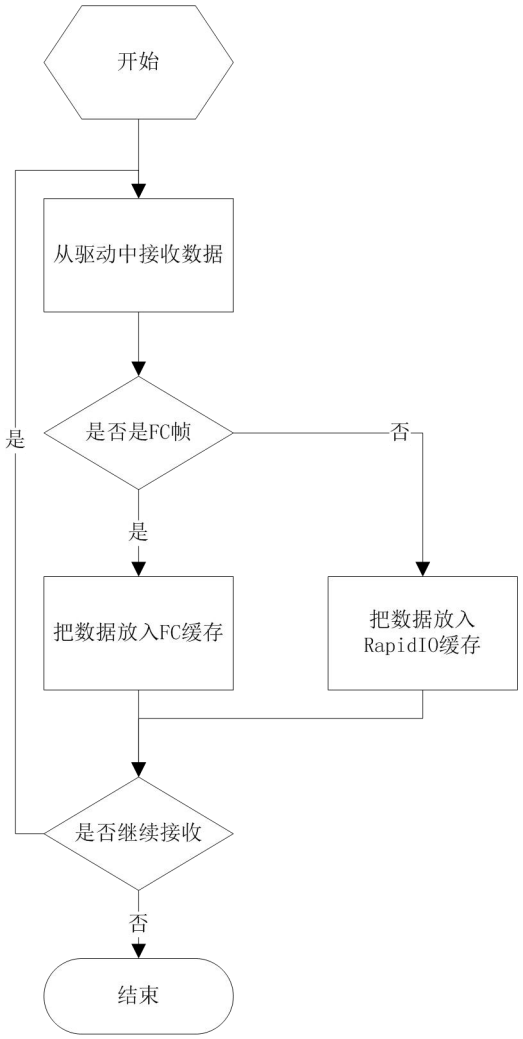


图 4-13 数据接收流程图

接收数据时，首先从驱动中接收数据，然后判断此数据是否为 FC 帧，如果是 FC 帧，则把数据放入 FC 缓存中，若是 RapidIO 数据，则把数据放入 RapidIO 缓存中，然后再进行下一包的接收。

在上述流程图中，需要判断数据是否为 FC 帧，由于 RapidIO 数据包头中并没有相关的字段用以区别数据的类型，不像 FC 帧的帧头中有 Type 字段，Type 为 1 的时候表示数据是 ELS 消息，Type 为 49 时表示是 ASM 消息，因此对于 RapidIO 消息，只能通过净荷中的特殊字符来区别数据的类型。

在此系统中，由于数据只有 FC 帧（承载与 RapidIO 数据）和 RapidIO 数据两种类型，并且 FC 帧具有 SOF、CRC、EOF 这三个特殊标识，因此我们可以通过判断数净荷据帧中的 SOF、CRC 和 EOF 来判断是否为 FC 帧。

由于采用 FC 第三类服务，因此有效的 SOF、CRC 和 EOF 为：

表 4-2 FC 第三类服务有效 SOF、CRC 和 EOF

SOFi3	0xBCB55656
SOFn3	0xBCB53636
EOFn	0xBC95D5D5/ 0xBCB5D5D5
EOFt	0xBC957575/ 0xBCB57575
CRC	0XA5A5A5A5

因此，适配层在接收到一个数据包时，进行以下判断：

1. 判断数据包载荷的第一个字是否为有效的 SOF，如果是则进行第二步，如果不是则判断当前数据包不是 FC 帧。
2. 判断数据包载荷的最后一个字是否为有效的 EOF，如果是则进行第三步，如果不是则判断当前数据包不是 FC 帧。
3. 判断数据包载荷的倒数第二个字是否是有效 CRC—0xA5A5A5A5，如果是则判断此数据包载荷为 FC 帧，否则不是 FC 帧。

在就是说，只有当上述三个步骤判断完毕并且成立的情况下，该消息才是 FC 帧，否则只该消息就为 RapidIO 消息。

图 4-14 为判断消息类型的流程图。

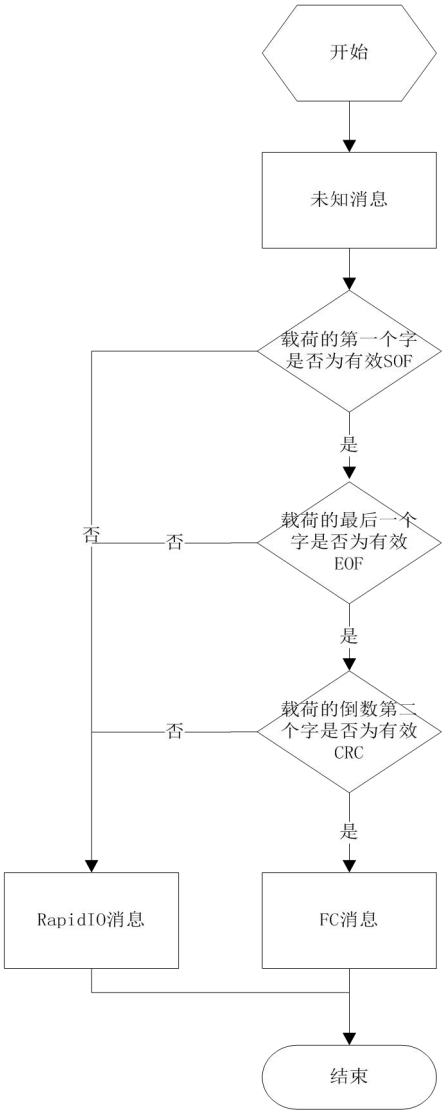


图 4-14 判断消息类型流程图

4.2.3 中转节点的实现

在第三章，3.2 节中曾提到每块处理器板的 1 号 CPU 都有一个 XMC 接口，可用于扩展连接一块 FC 仿真卡（项目中使用的是成电光信科技股份有限公司所研发的 FC 仿真卡）。图 4-7 中的中转节点就特指扩展了 FC 仿真卡的处理器板模块。

首先，中转节点包括一块 FC 仿真卡，同时也通过背板连接到了 RapidIO 交换芯片，而其他未扩展 FC 仿真卡的模块也都通过背板接到了 RapidIO 交换芯片。其次，从数据来源来说，中转节点的数据可能来自仿真卡，也可能来自 RapidIO 交换芯片，而未扩展 FC 仿真卡的处理器板数据只来自交换芯片。因此，在功能上来将，中转节点不止可以作为一个普通的处理器板，同时也可以是一个桥接设备，实现系统内部 RapidIO 协议和外部 FC 协议的桥接。

中转节点的层次模型如图 4-15 所示。

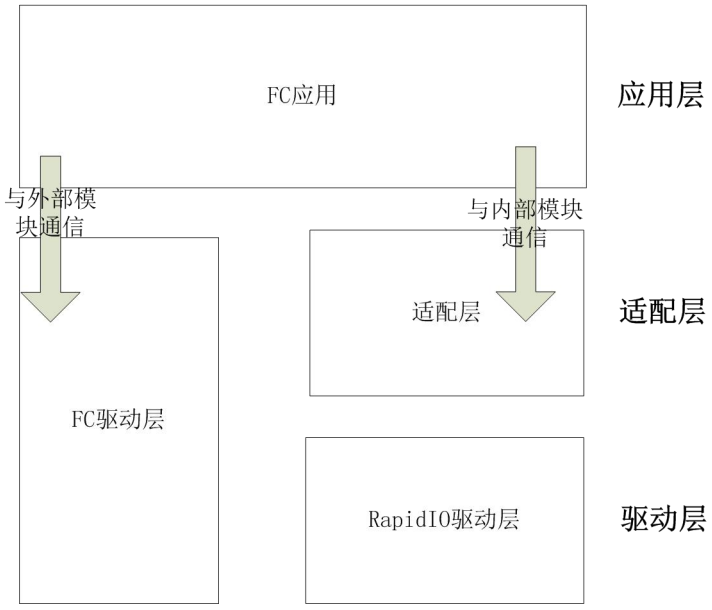


图 4-15 中转节点层次模型

中转节点的底层包括 FC 驱动层、RapidIO 驱动层、适配层。上图中的 FC 应用其实就代指了 FC 仿真卡。当中转节点要与外部设备通信时，数据通过 FC 驱动层发送出去；当与系统内部模块通信时，通过适配层和 RapidIO 驱动层发送。

当系统内部模块与外部设备通信时，中转节点还需要负责数据的转发，因此 FC 应用还必须实现转发功能。FC 应用的具体构成如图 4-16 所示。

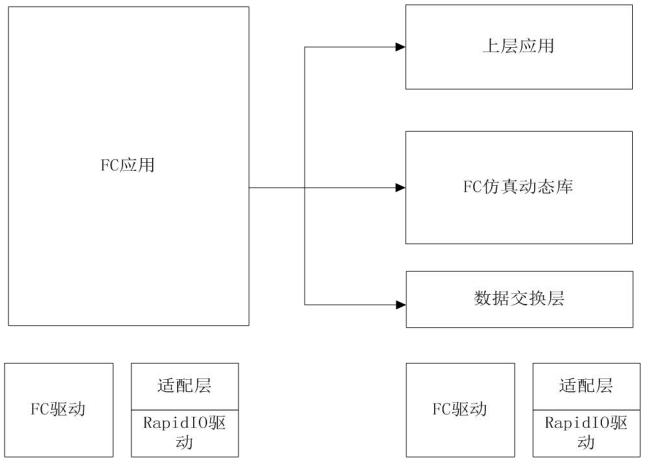
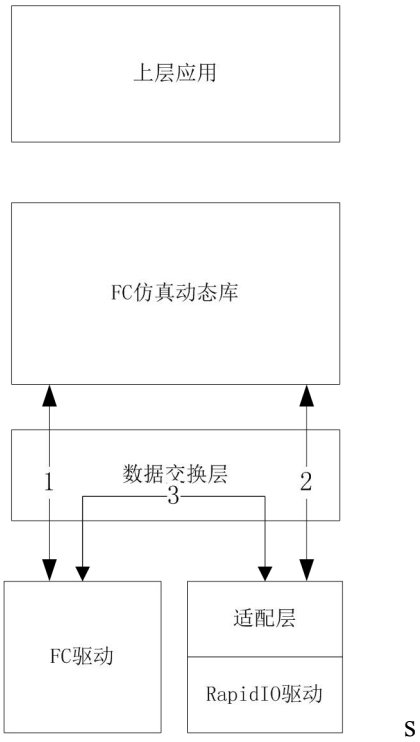


图 4-16 FC 应用构成

在图 4-16 中，FC 应用包括三层构成，自低向上分别是数据交换层、FC 仿真动态库层和上层应用。数据交换层负责从 FC 驱动和适配层中提取数据，然后进行消息转发。数据交换层的接收数据流有 3 种，如图 4-17 所示。



S

图 4-17 数据交换层数据流

图 4-17 中标注了三种交换层的数据流。

1. FC 仿真动态库通过数据交换层和 FC 驱动进行数据交互，也就是仿真卡与外部 FC 模块通信。
2. FC 仿真动态库通过数据交换层和适配层进行数据交互，也就是仿真卡与系统内部模块进行通信。
3. FC 驱动通过数据交换层和适配层进行数据交互，也就是外部 FC 模块与系统内部模块进行通信。

通过上面三种数据流，可以知道数据交换层的功能和难点就是实现数据目的地址的识别。具体点来说，就是判断以下几点。

1. 从 FC 仿真动态库接收消息后，判断此消息是发送给 FC 驱动还是适配层。
2. 从 FC 驱动接收消息后，判断此消息是发送给 FC 仿真动态库还是适配层。
3. 从适配层接收到消息后，判断此消息是发送给 FC 仿真动态库还是 FC 驱动。

从上面的 4.2.2.1 节可知，目的地址的识别，需要靠查找地址转换表实现。

当数据交换层收到 FC 仿真动态库数据时，处理流程如图 4-18 所示。

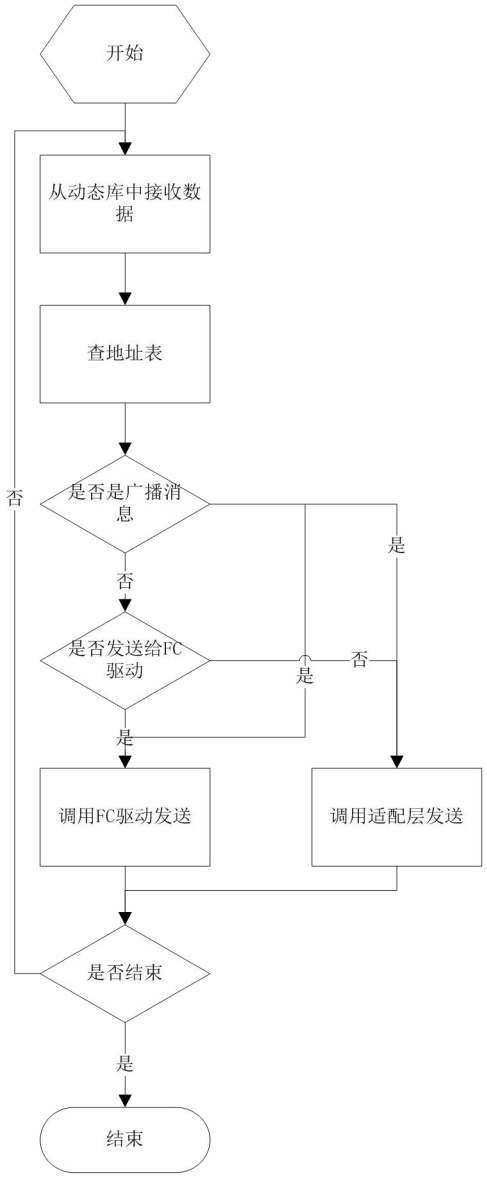


图 4-18 数据从仿真动态库发送流程图

当交换层从仿真动态库中接收到 FC 帧后，首先查询地址表，判断此消息是否是广播消息，如果是广播消息则把 FC 帧分别发送到 FC 驱动和适配层中。如果不是，则需判断此 FC 消息是发送至 FC 驱动还是适配层，判断依据仍参照地址表，如果地址表中有此目的地址的对应表项，则把消息发送给适配层，如果没有对应表项，则发送给 FC 驱动。

当数据交换层从 FC 驱动或者适配层接收数据时，处理流程如图 4-19。

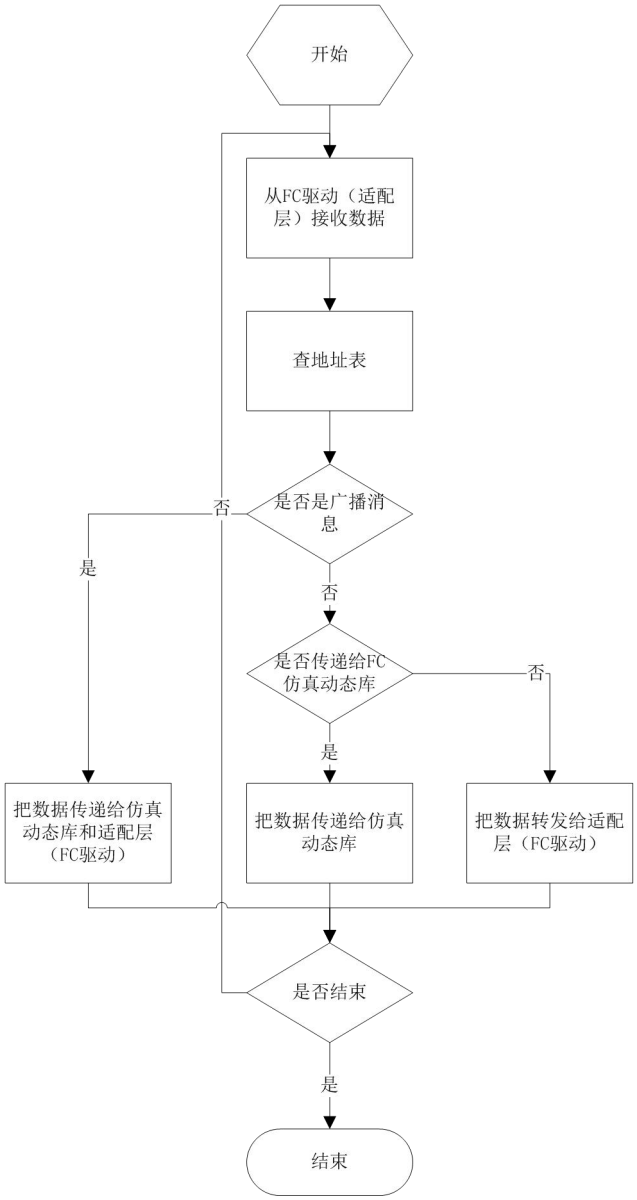


图 4-19 数据从 FC 驱动或适配层发送流程图

首先数据交换层从 FC 驱动或者适配层接收数据，然后通过查找地址表判断数据是否为广播消息，如果是，则把此消息传递给仿真动态库层同时转发适配层和 FC 驱动。如果不是广播消息，则判断此消息是否是发送给 FC 仿真动态库，判断依据为本地端口地址与目的 ID 是否匹配。

4.2.4 应用消息发送接收流程

综合上述解决方案，给出消息的发送和接收流程。

发送消息流程如图 4-20 所示。

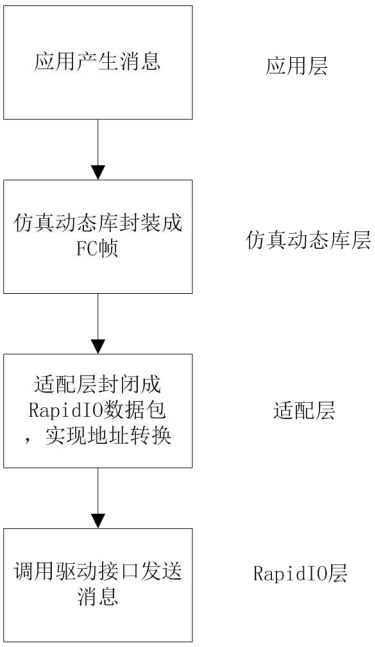


图 4-20 发送消息流程图

应用层产生应用消息，把消息传递给仿真动态库。仿真动态库接收到消息后，添加 SOF，FC 帧头，ASM 消息头（或者 ELS 消息头）、CRC、EOF，封装成 FC 真。仿真动态库调用适配层接口把 FC 帧传递给适配层，适配层再将 FC 帧进行进一步封装，将其整个 FC 帧放到 RapidIO 第 11 类包的净荷中然后调用标准 RapidIO 接口将 RapidIO 包发送出去。

图 4-21 为消息接收的流程。

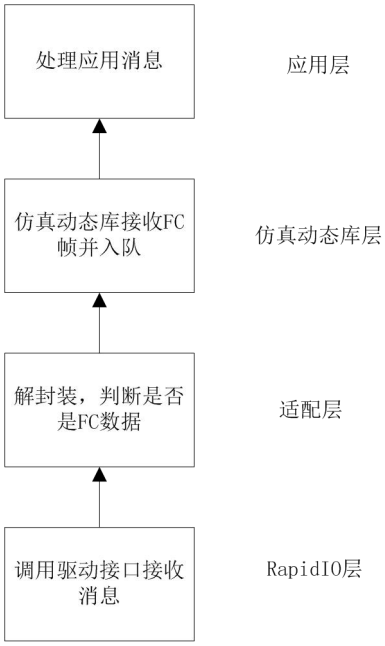


图 4-21 接收消息流程图

RapidIO 层从驱动接收数据，然后把数据包递交给适配层，适配层判断此数据是 FC 帧还是 RapidIO 数据包（如何区别参见 4.2.2.2 节）。如果是 FC 数据，则将数据递交给仿真动态库，最后再递交给应用层。

4.3 本章小结

本章对模拟仿真综合核心处理机系统的关键技术进行了详细的阐述。BSP 开发和硬件驱动的设计，驱动方面主要着重介绍了 RapidIO 驱动的设计与实现。详细介绍了 RapidIO 承载 FC 消息模块的设计与实现，针对以 FC 交换机作为交换设备和以 RapidIO 交换机作为交换设备时数据流的对比，提出了增加适配层的方案，详细描述了适配层的实现，以及中转节点的实现。

第五章 仿真系统的测试测试方法

本仿真系统由于系统的复杂性，测试任务涉及方面很广，而本文主要着重于RapidIO 和 FC 的通信测试。前后可以分为 RapidIO 测试、FC 测试、RapidIO 承载 FC 测试。每个阶段均进行反复的功能性测试和长时间的稳定性测试，功能测试会持续 5 分钟左右，而性能测试会根据测试情况持续半小时到 12 小时之间。通过后方能进入下一阶段，以保证仿真系统的可靠性。

通过自行设计开发的测试软件，对系统进行全面的测试，测试软件也可以作为本仿真系统的一种上层应用软件。图 5-1 为测试软件的界面图。

图 5-1 测试软件界面

本测试软件通过 UDP 协议与各处理器板进行通信，根据自定义的一套协议命令码，发送命令使各处理器板进行各种功能测试，测试软件的功能很全面，对系统整机的功能点测试都有覆盖，其中网络和始终控制都是基于 FC 卡的功能模块。在图 5-1 中圈的位置可以看出，根据设备端口的选择，可以设置为两种设备类型：FC 类型和 RapidIO 类型，分别对应 FC 测试和 RapidIO 承载 FC 消息的测试，而消息收发控制中，根据自定义的一套协议，不同的消息 ID 则可以控制消息发送的目的地址，以达到各测试线路的切换。而 RapidIO 测试则会在串口调试界面通过调

用测试函数进行。

5.1.1 RapidIO 测试

为使 RapidIO 承载 FC 消息的设计方案能够实现，首先必须保证 RapidIO 通信的稳定和高性能，这也是后续功能能够实现的基础，为此设计了几种场景进行测试，如图 5-2 所示。

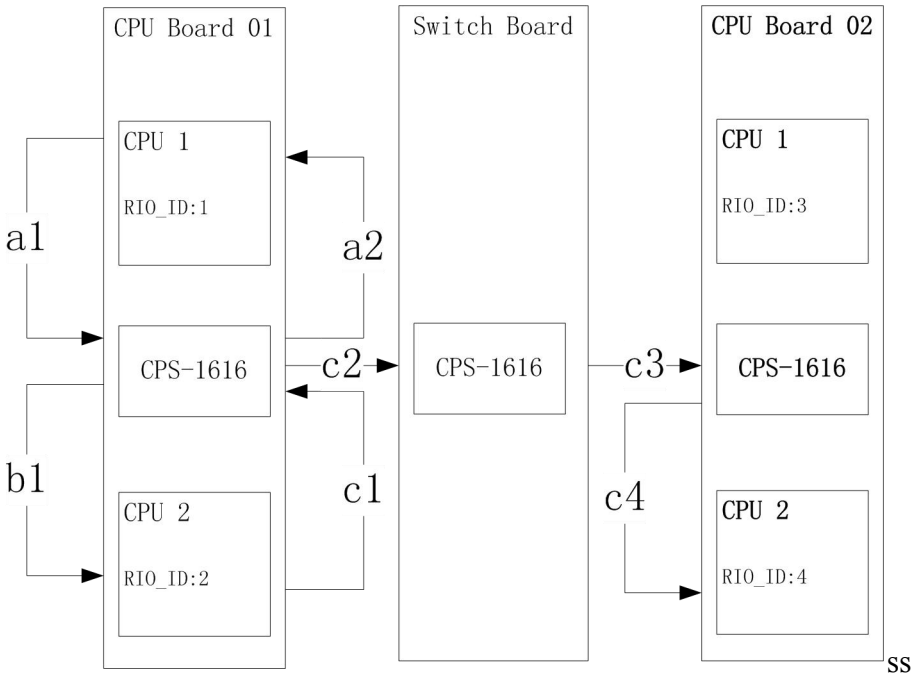


图 5-2 RapidIO 测试配置示意图

根据图 5-2 RapidIO 测试配置图可以看到，针对 RapidIO 通信设计出了 3 种测试方法。分别为单 CPU 的 RapidIO 自环测试，单处理器板内两个 CPU 间的 RapidIO 互联通信测试，两处理器板间的 CPU 间的 RapidIO 互联通信测试。

1. 单 CPU 的 RapidIO 自环测试：如图 5-2 中的“a1—a2”通信线路，01 号单板上的 CPU1 以自己的 RioID 作为目的地址发送 RapidIO 消息，经过所在处理器板上的交换芯片路由后，再转发给 CPU1。该测试的目的在于验证单块处理器板上每个 CPU 进行 RapidIO 数据交换处理时 CPU 的使用情况以及 RapidIO 速度、稳定性，这样的测试保证了 CPU 的正常工作，也间接验证了 RapidIO 基于 VxWorks 的底层驱动的情况。
2. 单板内两个 CPU 间的 RapidIO 互联通信测试：如图 5-2 中的“a1—b1”通信链路，01 号单板上的 CPU1 以 CPU2 的 RioID 作为目的地址发送 RapidIO 消息，经过所在板上的交换芯片路由后，发送给 CPU2。该测试的目的是为了验证单板上两个 CPU 间进行 RapidIO 数据交换处理的能力，得到 RapidIO 速率，对

RapidIO 的性能进行验证，而且也对两个 CPU 进行 RapidIO 互联通信时的 CPU 性能进行了测试。

3. 两单板间的 CPU 间的 RapidIO 互联通信测试：如图 5-2 中的“c1—c2—c3—c4”通信线路，01 号单板的 CPU2 以 02 号单板的 CPU2 的 RioID 作为目的地址发送 RapidIO 消息，首先消息会发送至 01 号单板的交换芯片查找路由，在 4.1.2 节 RapidIO 驱动中介绍到配置路由信息，这里配置的路由有一个默认端口，凡是目的地址在板外的，均通过这个默认端口发送至交换板的交换芯片，交换板再通过目的地址查找路由转发至 02 号单板的交换芯片，最后再由 02 号交换芯片转发给给 02 号单板的 CPU2，因此两单板间的 CPU 间的 RapidIO 互联通信一共经过了 3 层路由。此测试的目的在于验证了两个单板间的 CPU 进行 RapidIO 数据交换处理的能力，可以得知此类通信的 RapidIO 速率和性能，同时也对 CPU 的性能进行了验证。根据此测试项，又可以衍生出多单板间的多 CPU 间 RapidIO 互联通信的测试。

5.1.2 FC 测试

在保证 RapidIO 通信的稳定和高性能的前提下，还需要针对 FC 消息的通信性能进行验证。在每块处理器板的 1 号 CPU 都有一个 XMC 接口，配备了一块成电光信科技股份责任公司所生产的 FC 仿真卡。在 FC 通信功能完整，性能优越稳定的前提下，才能进一步验证 RapidIO 承载 FC 通信的性能。因此设计出了如图 5-3 FC 测试配置示意图上所指出了 3 种测试场景。

本文由于着重于 RapidIO 承载 FC 消息的实现，所以对 FC 的测试也着重针对消息收发的测试，而对 FC 仿真其他功能，如设备管理、时钟管理、网络管理等就不再赘述。

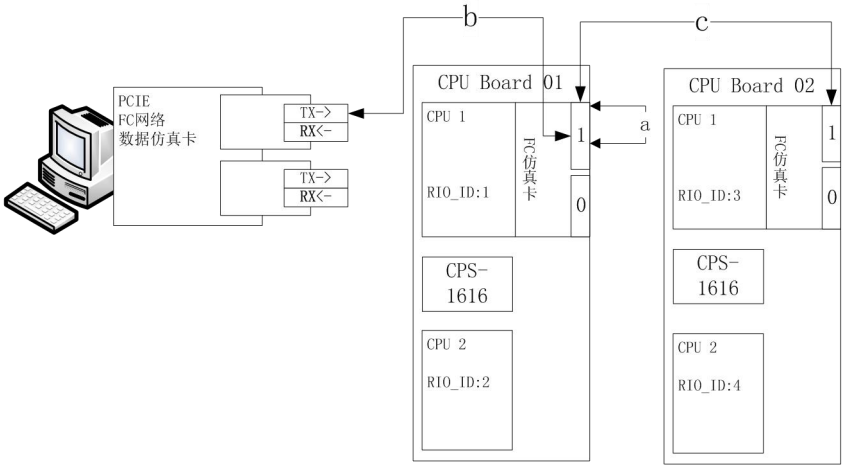


图 5-3 FC 测试配置示意图

由图 5-3 FC 测试配置示意图可以得知，这里的 FC 消息并不是承载与 RapidIO 的，而是单纯的 FC 帧消息的收发，如图 5-2 中所标记出的 a、b、c 这 4 条通信线路均未通过 RapidIO 交换芯片，也未通过交换板（因此图 5-3 中未画出交换板），a、b 和 c 三条线路均为两条光纤线，但都只用单向线路表明，并未画出两条光纤线，真实设备连线中为两个 FC 仿真卡的输入端和输出端对连。图 5-3 中的 3 条通信链路分别对应 4 种测试场景：FC 单板单端口自环测试、FC 单板双端口通信测试、FC 单板端口与 PC 端的 FC 仿真卡的通信测试、不同处理器板间的 FC 端口互联通信测试。

1. FC 单板单端口自环测试：如图 5-3 中的通信线路 a，直接将 CPU Board01 上的 FC 仿真卡 1 端口的输入输出用光纤线对接，进行 FC 数据通信。该测试的目的在于验证单板上的 CPU 进行 FC 数据交换处理时，CPU 的性能使用情况以及 FC 的速率，并且验证了 FC 基于 VxWorks 的底层驱动情况。
2. FC 单板端口与 PC 端的 FC 仿真卡的通信测试：如图 5-3 中的通信线路 b，将 CPU Board01 上的 FC 仿真卡的 1 端口的输入输出分别与 PC 机上的 FC 仿真卡的输出输入利用光纤线相连，收发 FC 数据。此测试的目的在于验证 CPU 板对不同的 FC 数据源的收发的处理能力，以及此情景下的 FC 速率。
3. 不同处理器板间的 FC 端口互联通信测试：如图 5-3 中的通信线路 c，将 CPU Board01 上的 FC 仿真卡的 1 端口的输入输出分别与 CPU Board02 上的 FC 仿真卡的 1 端口的输出输入利用光纤线相连，进行 FC 数据收发。此测试的目的在于验证多 CPU 板卡之间 FC 点到点的数据处理能力，以及此情景下 FC 传输速率。

5.1.3 RapidIO 承载 FC 测试

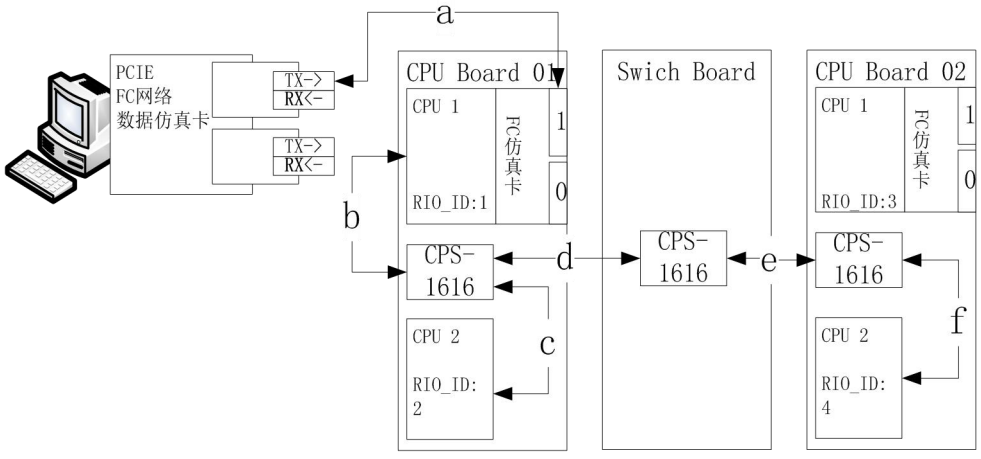


图 5-4 RapidIO 承载 FC 测试配置示意图

由图 5-4 RapidIO 承载 FC 测试配示意图可知，图中标记了 a，b，c，d，e 和 f 这 5 种通信线路，其中 a 线路是采用光纤线，也 FC 总线协议传输，a 线路是纯粹的 FC 帧消息，并没有以 RapidIO 承载的方式进行传输，a 线路只是测试中为了与外界通信搭建，用以验证通信结果。b，c，d，e 和 f 均是以 RapidIO 总线进行数据传输通信，其中 RapidIO 消息的净荷为 FC 帧消息，图 5-4 中给出了两种测试场景：单板内两个 CPU 的 FC 数据转化为 RapidIO 数据通信测试，跨单板的 CPU 之间的 FC 数据转化为 RapidIO 的数据通信测试。

1. 单板内两个 CPU 的 RapidIO 承载 FC 数据通信测试：如图 5-4 中的通信线路“b—c”，CPU Board01 的 CPU1 以此单板的 CPU2 的 FC 地址作为目的地址，适配层将 FC 消息封装在 RapidIO 消息内，将 FC 地址转化为 RapidIO 地址，以 RapidIO 消息的形式发送给此单板的 RapidIO 交换机 CPS-1616，通过 RapidIO 交换机查找目的地址，路由转发到 CPU2，CPU2 再对 RapidIO 消息的净荷进行解析，得到 FC 帧消息，整个过程进行了一次路由。此测试的目的在于验证同一个单板上 CPU1 以 RapidIO 承载 FC 消息的形式传送给 CPU2 后，速率的损耗，以及转化后的数据处理能力，CPU 的的性能使用情况等。
2. 跨单板的 CPU 之间的 RapidIO 承载 FC 数据通信测试：如图 5-4 中通信线路“b—d—e—f”，CPU Board01 的 CPU1 以 CPU Board02 上的 CPU2 的 FC 地址作为目的地址，适配层将 FC 消息封装在 RapidIO 消息内，并将 FC 地址转化为 RapidIO 地址，并以 RapidIO 消息的形式发送给此单板的 RapidIO 交换机 CPS-1616，此时与 5.1.1 节中第 2 点相似，由于目的地址在板外，所以直接通过默认路由，将此消息转发给 Switch Board 上的 RapidIO 交换机，此交换机又通过查找目的地址找到 CPU Board02，将此消息转发给 CPU Board02 上的 RapidIO 交换机，最后再由 CPU Board02 上的 RapidIO 交换机转发给最终的目的地址 CPU2，整个过程经过了三次路由。此测试的目的在于验证跨单板的 CPU 之间将 FC 消息转化为 RapidIO 后速率的损耗，以及转化后数据的处理能力，CPU 的性能使用情况等。

5.2 测试结果

通过自行设计开发的测试软件，经过三个阶段的充分测试，仿真系统均运行正确且稳定。以下分别为本仿真系统上述上中测试结果的详细数据指标。所有设备状态信息和消息收发情况均为实际运行过程中应用测试软件界面的真实截图，

而所有功能性能状态数据均是通过对应用软件界面真实统计数据整理所得。

5.2.1 RapidIO 测试结果

通过 5.1.1 节对 RapidIO 测试方案的描述，在对应的线路正确连接，以及测试函数正确无误的前提下，测试将会在串口调试界面进行，通过命令行调用测试函数，得到相应的测试结果。

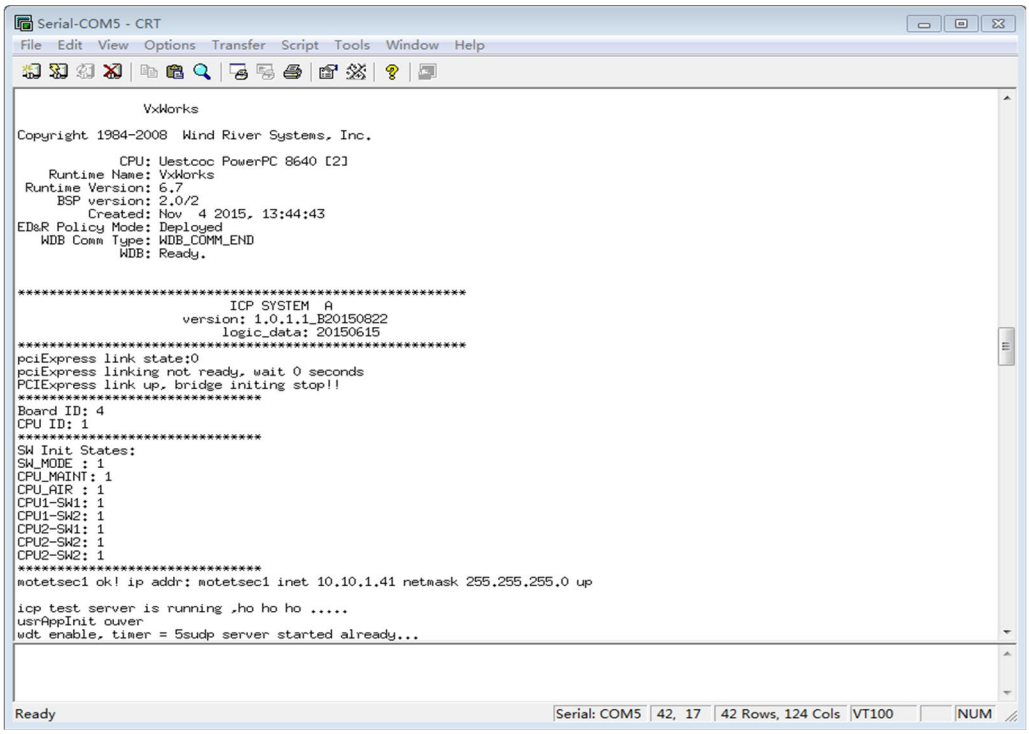


图 5-5 串口界面图

图 5-5 为串口调试界面图，可以看到 BootRom 加载完成的界面，界面上显示了一系列状态信息，接下来就是在命令行输入测试函数。对 RapidIO 测试的函数大体上包括 3 个线程：接受线程，发送线程和速率计算线程。其中发送的目的地址作为函数参数，在调用发送函数的时候使目的地址可控从而达到不同测试线路的切换。

在串口界面输入测试命令后 test_rio_recv，进行 Rio 初始化、接收线程和速率计算线程的初始化，然后输入测试命令 test_rio_send，在编译前此命令的一个重要参数—目的地址，决定了通信线路的不同，以实现在 5.1.1 节中所提到的三种测试线路。

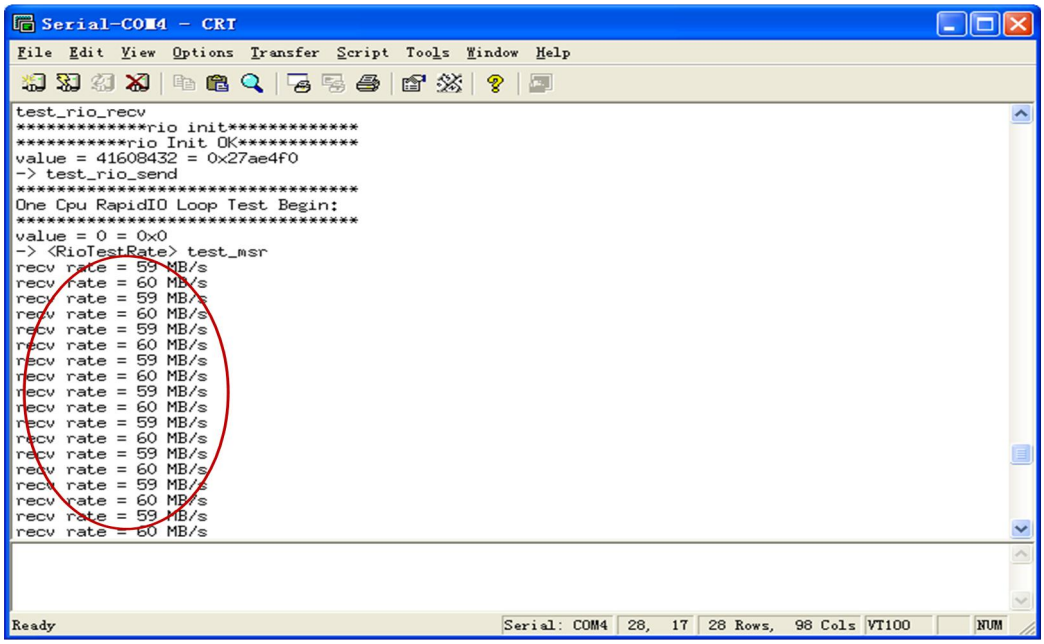


图 5-6 RapidIO 自环测试结果

图 5-6 为 RapidIO 自环测试，也就是对应 5.1.1 节中的第一种测试线路，同一个 CPU 经过 CPS-1616 与自己进行通信，红圈部分可以看出速率在 60MB/S 左右。

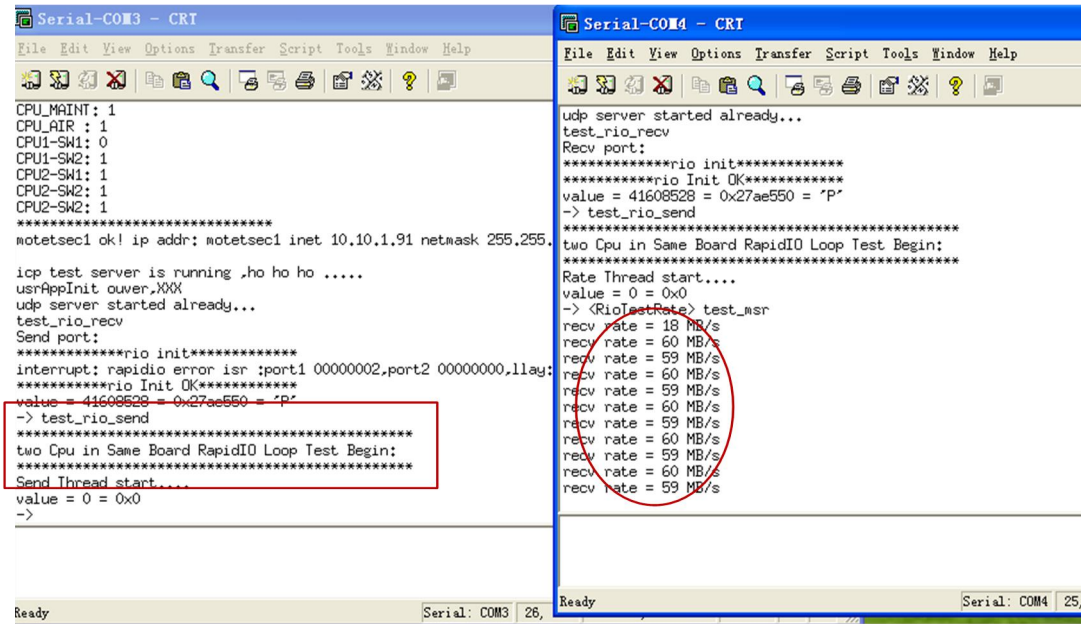


图 5-7 RapidIO 同板两 CPU 通信测试结果

图 5-7 为 RapidIO 同板两 CPU 通信，也就是对应 5.1.1 节中的第二种测试线路，同一块处理器板上两个 CPU 经过本板的 CPS-1616 进行通信，图 5-7 左边红色长方形框形中为发送命令，右边的红圈部分为接受速率，可以看出速率在 60MB/S 左右。

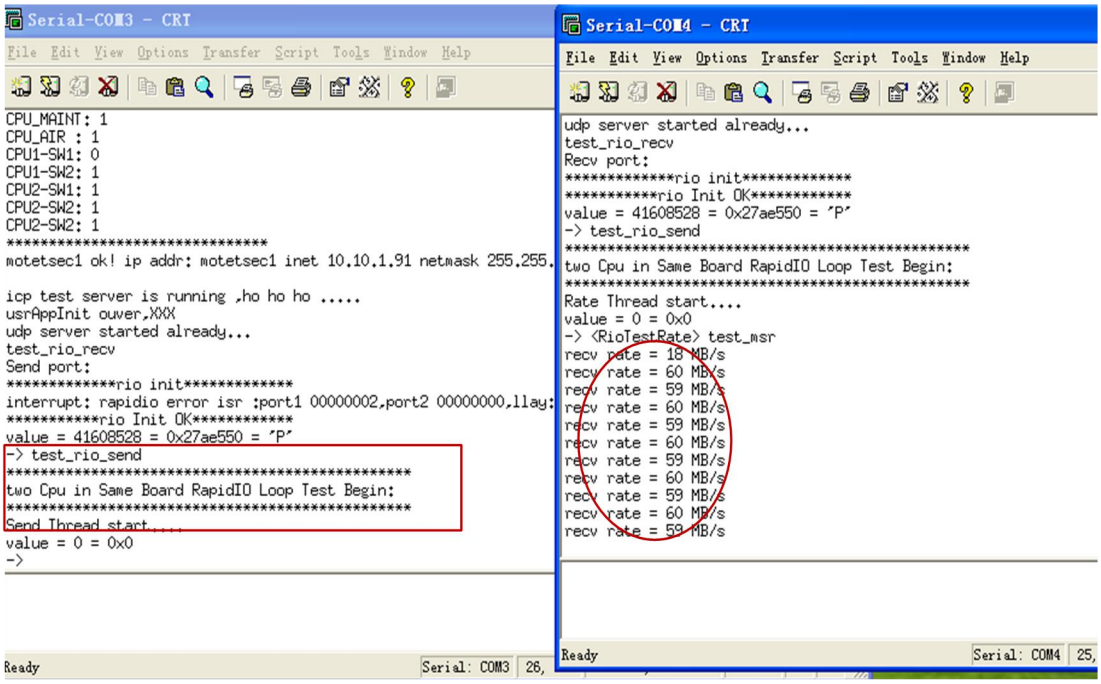


图 5-8 RapidIO 异板两 CPU 通信测试结果

图 5-8 中为 RapidIO 同板两 CPU 通信，也就是对应 5.1.1 节中的第三种测试线路，不同处理器板上两个 CPU 经过交换板的 CPS-1616 进行通信，可以看出速率在 60MB/S 左右。

表 5-1 RapidIO 测试结果

通信线路		发送速率	接收速率	性能达标
RapidIO	自环测试	60MB/S	60MB/S	是
	同板两 CPU	60MB/S	60MB/S	是
	异板两 CPU	60MB/S	60MB/S	是

表 5-1 为 RapidIO 测试结果，性能是否达标是根据用户所需求的速率进行对于得到。

5.2.2 FC 测试结果

通过 5.1.1 节对 FC 测试方案的描述，在对应的线路正确连接，连线均为光纤线连接，并未通过 RapidIO 总线承载通信，以及测试软件正确无误的前提下，测试将会通过自行开发的测试软件进行，此处不再赘述测试软件的用法，着重关注测试结果。

图 5-9 FC 自环测试结果

图 5-9 中为 FC 自环通信测试结果，对应 5.1.2 节中的第一种测试线路，FC 卡的自环测试，红色长方形框中可以看出接收和发送速率均在 50MB/S 左右，以及在图 5-9 中发送消息的个数和接收个数存在微量差异，是由于测试软件界面刷新时间所影响的，并非丢帧所导致。

图 5-10 FC 卡与 PC 端 FC 仿真卡通信测试结果

图 5-10 中为处理器板上的 FC 卡与 PC 端的 FC 仿真卡的通信测试结果，对应 5.1.2 节中的第二种测试线路，图中左边为测试软件，右边为 FC 仿真软件，图 5-10 中红色长方形框中显示的该系统与 FC 仿真软件互发消息时的发送和接受速率。可以看出本系统与 PC 端 FC 仿真卡通信时，本系统的接受速率在 100MB/S 左右，发送速率在 45MB/S 左右。在图 5-10 中发送消息的个数和接收个数存在微量差异，是由于测试软件界面刷新时间所影响的，并非丢帧所导致。

图 5-11 异板两 FC 卡通信测试结果

图 5-11 中为不同处理器板上的两个 FC 卡的通信测试结果，对应 5.1.2 节中的第三种测试线路，图 5-11 中的红色长方形框中为两个 FC 卡通信的发送和接受速率。可以看出异板两 FC 仿真卡通信时，接收和发送速率均在 50MB/S 左右。在图 5-11 中左边和右边的测试软件中，发送消息的个数和接收个数存在微量差异，是由于测试软件界面刷新时间所影响的，并非丢帧所导致。

表 5-2 FC 测试结果

通信线路		发送速率	接收速率	性能达标
FC	自环测试	50MB/S	50MB/S	是
	与 PC 端 FC 仿真卡	45MB/S	100MB/S	是
	异板两 FC 仿真卡	50MB/S	50MB/S	是

表 5-2 为 FC 通信测试结果，性能是否达标是根据用户所需求的速率进行对于得到。

5.2.3 RapidIO 承载 FC 测试

通过 5.1.1 节对 RapidIO 承载 FC 测试方案的描述，在对应的线路正确连接，此时连线均为 RapidIO 总线，以及测试软件正确无误的前提下，测试将会通过自行开发的测试软件进行，此处不再赘述测试软件的用法，着重关注测试结果。

图 5-12 FC Over RapidIO 自环测试结果

图 5-12 中为 RapidIO 承载 FC 自环通信测试结果，对应 5.1.3 节中的第一种测试线路，图 5-12 的红色长方形框中显示出接收和发送速率均在 28MB/S 左右，在图 5-12 中中发送消息的个数和接收个数存在微量差异，是由于测试软件界面刷新时间所影响的，并非丢帧所导致。

图 5-13 FC Over RapidIO 同板两 CPU 通信测试结果

图 5-13 中为相同处理器板上两个 CPU 的 RapidIO 承载 FC 通信测试结果，对应 5.1.3 节中的第二种测试线路，图 5-13 中红色长方形框中显示出接收和发送速率均在 28MB/S 左右，在图 5-13 中中发送消息的个数和接收个数存在微量差异，是由于测试软件界面刷新时间所影响的，并非丢帧所导致。

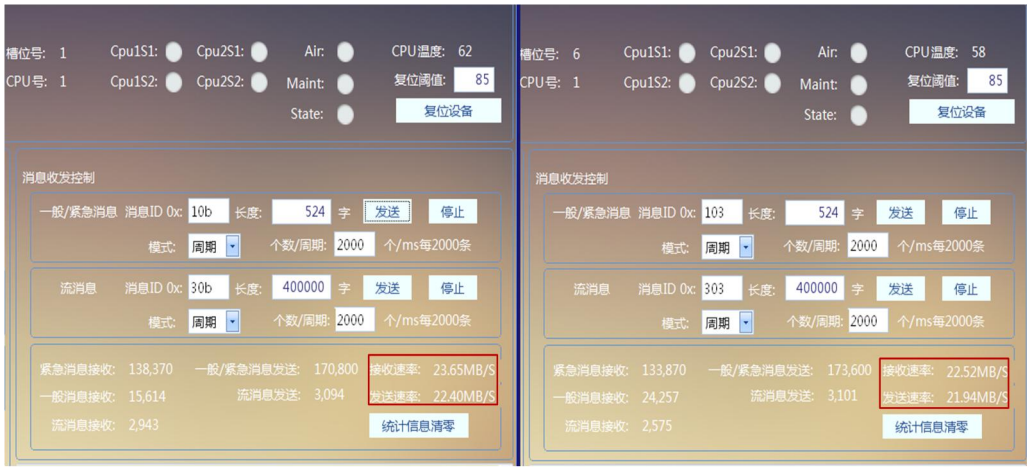


图 5- 14 FC Over RapidIO 异板两 CPU 通信测试结果

图 5-14 中为不同处理器板上两个 CPU 的 RapidIO 承载 FC 通信测试结果，对应 5.1.3 节中的第三种测试线路，图 5-14 中红色长方形框中显示出接收和发送速率均在 23MB/S 左右，在图 5-13 中中发送消息的个数和接收个数存在微量差异，是由于测试软件界面刷新时间所影响的，并非丢帧所导致。

表 5- 3 FC 测试结果

通信线路		发送速率	接收速率	性能达标
FC Over RapidIO	自环测试	28MB/S	28MB/S	是
	同板两 CPU	28MB/S	28MB/S	是
	异板两 CPU	23MB/S	23MB/S	是

表 5-3 为 FC 通信测试结果，性能是否达标是根据用户所需求的速率进行对于得到。

5.3 本章小结

本章分三个阶段对模拟仿真综合核心处理机系统进行测试。首先叙述每个阶

段的测试方案和测试目的，第一阶段为 RapidIO 消息的测试，此时的 RapidIO 消息并没有承载 FC 消息；第二阶段是 FC 消息通信的测试；第三阶段是在 RapidIO 承载 FC 消息的测试。对测试结果进行了分析，确认仿真系统通过达标。本仿真系统经全面测试合格后，已投入实际机载网络中应用。

第六章 全文总结与展望

6.1 全文总结

本文首先从航电系统的发展现状和发展态势和嵌入式系统的研究意义两个方面，引出本课题研究工作的背景及意义。之后介绍了综合核心处理机的国内外研究历史与现状，以及本文的主要贡献与创新。接着介绍了本系统的操作系统支持—VxWorks，以及系统中最为重要的两个器件，PowerPC8640 和 CPS-1616 交换芯片的特征以及为何选用这些器件。其后，介绍了本仿真系统的两个重要理论基础 FC 协议和 RapidIO 协议。从层次划分和帧结构这两方便详细介绍了 FC 协议，对 RapidIO 协议的体系结构进行了介绍。

针对模拟仿真综合核心处理机系统需要解决的问题，对仿真系统进行了总体结构设计。首先阐述了系统整体功能及性能需求，在总体设计的基础上从通信需求出发分别对其中最为重要的两种通信互联模式进行了描述，以太网互联结构和 RpiadIO 互联结构，介绍了系统的主要硬件平台，处理器板和交换板。

本文的核心内容是模拟仿真综合核心处理机系统的关键技术。在 BSP 开发和硬件驱动的设计中，驱动方面主要着重介绍了 RapidIO 驱动的设计与实现，详细介绍了 RapidIO 承载 FC 消息模块的设计与实现，针对以 FC 交换机作为交换设备和以 RapidIO 交换机作为交换设备时数据流的对比，提出了增加适配层的方案，详细描述了适配层的实现，以及中转节点的实现。根据对整体需求提到的多项技术难点的设计，实现了一套完整的仿真系统。

分三个阶段对模拟仿真综合核心处理机系统进行了充分测试。首先确定了测试的流程，第一阶段为 RapidIO 消息的测试，此阶段 RpiadIO 不承载 FC 消息；第二阶段进行 FC 消息的测试；第三阶段进行 RapidIO 承载 FC 消息的通信测试。论文给出了详细的测试方案，对测试结果进行了分析，确认通过达标。最后将仿真系统可作为核心处理设备接入实际机载网络中投入应用。

6.2 后续工作展望

在后期的联调工作中，由于用户所承载的上层软件或设备的要求不同，上层设备不断更新和功能的增加，以及数据量的大小和复杂性的不断提升，可能导致本仿真系统出现性能和功能上的瓶颈。所以仍需要配合上层软件或设备进行调试和修改。目前可能导致瓶颈的原因存在于软件层面，仍可能有更好的方案实现RapidIO 承载 FC 消息交换，在现有方案中，由于此模块中为了进行消息识别，FC 消息在 RapidIO 消息净荷中的输入输出等导致内存拷贝的次数过多，而内存拷贝的次数过多容易导致性能损耗。

另外一个方面，在测试中发现本系统的 FC 通信效率主要决定于软件的性能，软件运行系统越复杂，FC 的性能就会越低，而在内存性能提升的情况下也能相应的提高 FC 的性能。所以提升内存性能和减少软件复杂度，减少软件中数据拷贝次数，可以有效的提高 FC 的使用效率。

致　谢

三年的硕士研究生生活对于我来说是幸运的，在这三年中不仅学到了很多专业知识，工程应用技术，更学习了与人相处之道。这些都是主要受益与自己身边的老师、同学、同事等人的教导、帮助。在这里对他们表示由衷的感谢！

首先，我想真挚地感谢我的导师，邱昆教授。在这三年里，邱老师在学术上渊博的知识、敏锐的眼光、严肃认真的治学态度使深深的感染和鼓励着我。这将是我一生都值得学习的宝贵财富。自从踏入研究生生活后，邱老师便成为了我心目中的偶像和领航人。在此，我想对邱老师说：感谢您给我的启发，这三年辛苦您了！

同时，我要感谢科研团队的凌云老师、许渤老师、胡钢老师在学习和科研上对我的耐心指导和帮助。在项目研究过程中，三位老师在整个系统的设计和论文的写作方面给了我很大的支持和帮助。在此，对三位老师也表示诚挚的谢意！

此外，我还非常感谢成都成电光信科技股份有限公司，给了我一个难忘的实习经历。这个经历对我至关重要，它让我懂得了如何将学校里面学习的理论知识转化为实际工程应用技术，也算是提前进入社会的一个预演。感谢公司的所有同事，他们对我在系统设计和研发过程中遇到的技术难点给出了很好的建议和指点，很荣幸能够在这个集体里工作。

另外，我要感谢教研室的各位师兄师姐、师弟师妹。你们让我感受到了团队大家庭温暖和谐的氛围，祝愿你们在以后的事业生活上越来越好！

感谢各位老师参与本论文的评审和答辩，您宝贵的指导意见让我收获颇丰！

参考文献

[1] 林强, 熊华钢, 张其善.光纤通道综述[J].计算机应用研究, 2006, 02(12): 32-34

[2] 徐亚军，熊华钢. 未来航电系统 FC 互连的拓扑结构研究[J]. 电光与控制，2004,11:9-12

[3] 王一帆. 基于 FC 技术的航电网络仿真平台设计及性能研究[D].四川：电子科技大学，2010.6: 6-10

[4] 王学龙. 嵌入式 VxWorks 系统开发应用[M]. 北京：人民邮电出版社，2003:90-91

[5] 何粹刚. 机载光纤网络系统设计与仿真软件开发.[D]四川：电子科技大学，2010.10:34-41

[6] 周建涛.基于光纤通道的通信系统的研究与实现[D]. 西安:西北工业大学, 2007, 30-35

[7] 曹佳平等. VxWorks 设备驱动开发详解[M]. 北京：电子工业出版社，2009:12-15

[8] 罗国庆. VxWorks 与嵌入式开发[M]. 机械工业出版社，2003:23-25

[9] 安军社，刘艳秋，孙辉先. VxWorks 操作系统板级支持包的设计与实现[M]. 中科院空间中心.北京 100080

[10] IDT sRIO Gen2 Switch(CPS1848/CPS1616) Getting Started Guide Version0[S]. February,10 2011

[11] CPS-1616 Datasheet and User Manual Central Packet Switch[S]. July 10,2013

[12] 刘娟.光纤通道的核心技术研究与实现[D]. 西安:西安石油大学，2010:2-5

[13] 林强, 熊华钢, 张其善.光纤通道综述[J].计算机应用研究, 2006, 02(12): 32-34

[14] Q. Lin, H. G. Xiong, Q. S. Zhang. Credit Determination of Fibre Channel in Avionics Environment[J]. Chinese Journal of Aeronautics, 2007, 20(3): 247-252

[15] LogiCORE IP Serial RapidIO v5.6[S],UG503 March 1,2011

[16] 邓豹,赵小东. 基于串行 RapidIO 的嵌入式互连研究[j]. 航空计算技术, 2008(3):123-126

[17] 富勒. RapidIO 嵌入式系统互连[M]. 北京:电子工业出版社,2006,06:

[18] WindRiver System Inc.VxWorks BSP Developer's Guide 5.5[S].2002:270-273

[19] WindRiver Corporation. VxWorks 6.7 Programmer's Guide(Edition 1)[S].1999-03

[20] George Paap, Ed Silha. PowerPCTM : A Performance Architecture[S], IEEE ,1993, 1063, 1063-6390/03,104-108

[21] Keith Diefendorff， Rich Oehler， Ron Hochsprung. Evolution of the PowerPC[S]，IEEE,Micro,1994,0272-1732/94,34-49D

[22] Andrew J.Kornecki，Janusz Zalewski，Daniel Eyassu，Daniel Eyassy. Learning Real-Time Programming Concepts through VxWorks Lab[J]， IEEE Communications Magazine, 2007, 20(3): 247-252

[23] Rajkumar R. The real-time publisher/subscriber inter-process communication model for distributed real-time systems: design and implementation[C]. Chicago, 1995, 66-7