分类号：V474　　单位代码：10335

密　级：公开　　学　　号：21860263

浙江大学

硕士专业学位论文

中文论文题目：基于操作系统的微小卫星综合电子系统容错技术研究

英文论文题目：**Research on Fault Tolerance Technology of Microsatellite Integrated Electronic System Based on Operating System**

申请人姓名：陈涤昕

指导教师：金小军

合作导师：张顾洪

专业学位类别：工程硕士

专业学位领域：集成电路工程

所在学院：信息与电子工程学院

论文提交日期　2021 年 04 月 28 日

基于操作系统的微小卫星综合电子系统容错技术研究

论文作者签名：陈涤昕

指导教师签名：王小军

论文评阅人 1：盲审专家甲

评阅人 2：盲审专家乙

评阅人 3：盲审专家丙

评阅人 4：

评阅人 5：

答辩委员会主席：马慧莲 教授 浙江大学航空航天学院

委员 1：皇甫江涛 副教授 浙江大学信息与电子工程学院

委员 2：韦滨 高级工程师 西安卫星测控中心

委员 3：

委员 4：

委员 5：

答辩日期：2021 年 6 月 10 日

致谢

时间如白驹过隙转瞬即逝，硕士研究生的求学生涯也即将结束，回首这近三年的学习生活，一路上有良师益友陪伴。值此论文成文之际，谨向给予我帮助与支持的各位致以真诚的感谢。

首先感谢的是我的导师金小军副教授。在浙江大学微小卫星研究中心的学习、工作使我学习了丰富的专业知识，锻炼了我的团队协作能力。在金老师的悉心指导下，我收获了不曾有过的学习成果。

感谢课题组的张顾洪老师、王春晖老师、张朝杰老师、楼海君老师、赵凡宇老师，在项目工作中给予我指导。感谢庄恒佳工程师、许榕城工程师、张兄利工程师、郁武龙工程师、张国瑞工程师对我学习与工作的帮助。

感谢同课题组的于卓群博士、赵昌昌硕士、沈建谅硕士、张文君硕士、钱晨硕士、孟广凯硕士，在日常生活及学习工作给予我的帮助，很高兴曾与你们一起学习，一起战斗。感谢朱耀伟博士、许笑一博士、莫杭斌硕士、余鑫硕士、杜扬钦硕士，很高兴能与你们成为朋友。

感谢同班的吴斌硕士、李可硕士、薛聪硕士、唐志勇硕士、陈鹏硕士、吴靖康硕士、陈思恒硕士、沈阳硕士、张锦博硕士、黄文廷硕士、求天楠硕士，和你们一起生活的日子将会成为我最美好的回忆。

最后，感谢我的父母，感谢对我的养育之恩，将我培养成才。

祝各位在未来的日子里，身体健康，工作顺利，学业有成。

陈涤昕

于浙大智泉大楼

2021 年 6 月 14 日

摘 要

随着卫星任务趋于多样化、复杂化，星载综合电子系统开始向多任务调度、高速计算、大数据高速传输与大容量存储等方向发展。传统的航天级处理器已经难以满足当前需求，采用工业级元器件搭载操作系统成为发展的必然趋势。但是工业级元器件不具备抗辐射能力，在复杂的太空环境中易受高能辐射粒子影响发生故障，因此采用容错技术提升基于工业级元器件星载综合电子系统的可靠性成为当前研究的热门方向之一。本文结合国外内微小卫星综合电子系统容错技术发展趋势，针对 ZHX 卫星项目需求从系统容错架构设计、基于操作系统的星载可靠性加载设计以及系统级冗余备份管理方案三方面开展了星载综合电子系统容错技术研究。本文主要贡献如下：

一、针对 ZHX 项目支持操作系统、在轨智能计算、在轨可更换升级等功能需求，设计了综合电子系统双机异构备份冗余架构，通过高低速数据总线分离提升数据传输有效性，并针对重要数据进行点对点通信备份，提升了整系统的数据传输容错能力。

二、针对基于操作系统的星载可靠性加载需求，提出了三种基于三模冗余可靠性的加载方式，从加载效率、资源消耗、可移植性以及可靠性等方面对加载过程进行分析与研究，并完成了基于操作系统的星载可靠性加载模块设计与 FPGA 代码实现。

三、针对双机异构备份冗余架构下多处理器竞争式机制，提出了父级仲裁式与同级自主式两种冗余管理方式，解决了温备份下管理权冲突问题，实现平稳可靠交接，并进一步设计实现了主从机切换流程与主从系统间系统信息同步机制，提升了双机异构备份冗余架构设计下的系统可靠性。

综上所述，本文所设计的综合电子系统架构采用搭载操作系统的 ARM 处理器与智能计算单元的硬件架构，并采用可靠性加载设计与冗余管理方案加固，因此具有高性能、高可靠性等优点，满足 ZHX 卫星综合电子系统设计需求。所提出的系统架构、星载可靠性加载方式及系统级冗余备份管理方案的设计思路具有通用性，功能模块具备可移植性，可以在其他卫星系统中得到应用。

关键词： 微小卫星 综合电子系统 操作系统 容错设计

Abstract

As satellite missions become more diversified and more complex, on-board integrated electronic systems have begun to develop in the direction of multi-task scheduling, high-speed computing, high-speed transmission of mass data, and large-capacity storage. Traditional aerospace-grade processors have been unable to meet current needs, and the use of industrial-grade components and operating systems has become an inevitable trend of development. However, industrial-grade components do not have the ability to resist radiation and are susceptible to failures caused by high-energy radiation particles in a complex space environment. Therefore, the use of fault-tolerant technology to improve the reliability on-board integrated electronic systems based on industrial-grade components has become one of the hottset research direction. Combined with the development trend of fault-tolerant technology for integrated electronic systems of micro-satellites at home and abroad, this paper carried out the research on fault-tolerant technology of on-board integrated electronic systems for ZHX satellite project requirements from three aspects: system fault-tolerant architecture design,on-board reliability loading design based on operating system and system-level redundant backup management plan. The main contributions of this paper are as follows:

1. For the functional requirements of the ZHX project, such as supporting operating systems, on-orbit intelligent computing, and on-orbit replaceable upgrades, a integrated electronic system dual-computer heterogeneous backup redundancy architecture was designed, and the efficiency of data transmission was improved through the separation of high-speed and low-speed data buses. Significant data is backed up by point-to-point communication to improve the data transmission fault tolerance of the entire system.

2. For the requirements of on-board reliability loading based on operating system, three modes of reliability loading based on triple modular redundancy are proposed, and the loading process is analyzed and researched from the aspects of loading efficiency, resource consumption, portability and reliability. And completed the on-board reliability loading module design based on the operating system and FPGA code implementation.

3. For the multi-processor competition mechanism under the dual-computer heterogeneous backup redundancy architecture, two redundancy management methods, parent-level arbitration and same-level autonomous system backup management methods, are proposed, which solves the problem of management rights conflicts under warm backup and realizes smooth and reliable transfer. And further designed and realized the master-slave switching process and the system information synchronization mechanism between the master-slave system, and improved the system reliability under the dual-computer heterogeneous backup redundant architecture design.

In summary, the integrated electronic system architecture designed in this paper adopts the hardware architecture of ARM processor equipped with operating system and intelligent computing unit, and adopts reliability loading design and redundancy management scheme reinforcement, so it has high performance and high reliability. And other advantages, to meet the design requirements of the ZHX satellite integrated electronic system. The design ideas of the proposed system architecture, on-board reliability loading method and system-level redundant backup management scheme are universal, and the functional modules are portable so it can be used in other satellite systems.

Keywords: Micro-satellites, Integrated electronic systems, Operation system, Fault-tolerant technology

目 录

图目录

表目录

1 绪论

随着小型化电子设备集成度的提升以及空间卫星技术的发展，卫星的小型化逐渐成为航天领域“卫星竞赛”的趋势[1-2]。相比于传统大卫星研制成本高、研发周期长、系统复杂度高，微小卫星具有功能密度高、研发周期短、技术迭代快等特点[3]。

综合电子系统是当前微小卫星设计中重要的分系统之一，除了完成传统的遥控、遥测、星务管理、等平台任务外[4-5]，还实现了姿轨控、GNNS、智能处理等有效载荷的多任务调度、高速计算、大数据高速传输与大容量存储等功能。所以对综合电子系统处理能力的需求较之传统星务也需大幅提升，传统的航天级处理器已无法满足相应需求[6-7]，采用工业级元器件并针对其开展可靠性容错设计成为重要的发展趋势之一[8]。

1.1 微小卫星综合电子系统研究现状

综合电子系统概念源自航空工业，自航空工业综合电子系统技术发展成熟后，被引入航天领域成为卫星综合电子系统。卫星综合电子系统是采用通信总线网络技术连接各星载电子设备，实现数据共享、功能综合与资源优化合理组合[5]。综合电子系统对功能的综合不仅可以减少卫星平台电子设备使用的种类，还可减少元器件的使用数量。系统部件功能的实现以及系统对外接口的规范化和标准化，减小综合电子系统设计与验证工作，从而降低研发成本、减短研发周期。

早期微小卫星综合电子系统主要采用分布式架构，卫星数据与资源只在系统区域内共享，通过通信总线实现各系统之间信息交互[9-10]。如 2010 年发射的“天绘一号”立体测绘卫星，其综合电子系统采用分布式架构。通信总线采用 CAN 总线，并设计 2 级 CAN 总线网络，平台设备挂载在一级 CAN 总线上，载荷设备挂载在 2 级 CAN 总线上，实现了平台与设备之间的总线隔离。但卫星星务计算机、姿轨控单元、遥控单元独立设计，资源分散未合理配置[11]。

随着微小卫星朝着小体积、低功耗、高功能密度、低成本等的方向发展，分布式架构难以满足小卫星综合电子系统的需求。近年来国外的微小卫星综合电子系统都朝着集中式架构的方向发展。英国的萨瑞卫星技术公司的综合电子系统采用“数据集中，功能分布”的设计理念，星上计算机通过 CAN 总线获取星敏、飞轮、陀螺、电源控制、数传等模块的工程遥测

数据，实现数据的综合。

由美国研制发射的深空探测卫星空间技术-5（ST-5）总重 25kg、功耗约为 24W，并采用了集中式架构的综合电子系统[12]，其综合电子系统由两块总重约 1.5kg 的板子组成，完成了包括数据通信、遥控遥测、姿轨控、探测数据处理等卫星任务。

法国 Thalse Alenia Space 宇航技术研发公司研制的 Space bus4000 平台采用了集中式架构的综合电子系统[13]。该系统内性能强大的核心控制单元通过卫星通信总线网络技术实现与星上载荷单元的资源共享与数据交互，最终实现卫星任务一体化整合。

国内微小卫星综合电子系统研究起步较晚，与国外发展存在一定差距，但是随着近几年太空资源竞争以及商业航天发展，国内综合电子系统研究有了一定发展。哈尔滨工业大学自主研制的 TS-1 立体测绘太阳同步轨道卫星，综合电子系统通过系统总线收集其他分系统的遥测数据、工程数据以及系统状态信息，实现信息的集中管理，并通过系统总线对各分系统进行指令控制，实现综合电子系统与姿轨控、通信系统、电源系统、有效载荷系统协同工作[14]。

2011 年 11 月 9 日发射的“萤火一号”火星探测器的综合电子系统采用了集中式架构。“萤火一号”综合电子系统通过丰富的接口与其他分系统进行通信，并集中处理数据，实现功能的综合，完成了包括姿轨控数据管理、整卫星时统、热控、电源管理等任务[15]，综合电子系统功能图如图 1. 1 所示。

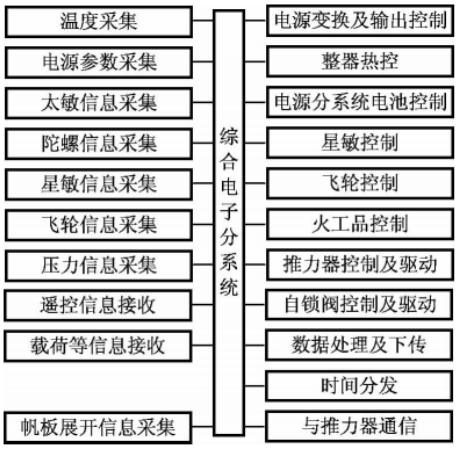

图 1. 1 “萤火一号”综合电子系统功能

南京航空航天大学发射的对地成像科学实验卫星“天巡一号”采用了数据集中化管理的方式完成星务管理、姿轨控制、电源控制单元管理任务[16]。

北京控制工程研究所的陈建新等人，在嫦娥三号巡视器上设计了集中式架构的综合电子系统，采用统一导航控制和数管计算机，软硬件协同设计、硬件资源复用，合并遥测和数传

等方法对系统功能与部件集成，以实现复杂功能[17]。

浙江大学自主研制并于 2015 年 9 月 25 日在太原发射中心成功发射的 ZDSP-2 皮星 2 号卫星的综合电子系统也采用集中式的架构[18-19]。综合电子系统对星上硬件资源进行统一分配管理，充分发挥软件功能，完成星务处理、姿轨控、电源管理、遥控遥测数据处理等卫星任务。

北京宇航系统工程研究所的徐喆垚等人设计的 LH-2 卫星采用了一体化综合电子系统思路。综合电子系统通过多种数据交互接口与数传模块、GPS 导航接收模、电源控制模块德等部分连接，实现各模块的信息共享[20]。

国内外微小卫星综合电子系统都朝着集中式架构方向发展。微小卫星硬件资源有限，采用集中式架构的综合电子系统可实现资源共享、统一分配、功能综合。

卫星综合电子系统采用集中式架构，数据集中在综合电子系统处理[21-22]，这对星载处理器的处理计算、处理能力提出了更高的要求[23]。卫星星载处理器主要分为 Intel 80X86 系列处理器、MIL-STD-1750A、ARM 处理器、PowerPC 体系结构处理器、SPRAC 架构处理器等[24]。其中 ARM 处理器的性能表现突出，具有低功耗、低成本、高性能的特点并且处理器的接口资源丰富，硬件可扩展性强。尤其是搭载了操作系统的 ARM 处理器在星务管理、数据处理方面性能强大，非常适合微小卫星的应用场景。

星载处理器采用 ARM 处理器并搭载操作系统的设计方案目前已经成为国内外研究热点之一。美国海军学院于 2007 年发射的 MidSTAR-1 在有效载荷上采用了搭载 Linux 操作系统的 ARM 处理器[25]。新加坡南洋理工学院于 2011 年发射的 X-Sat 卫星采用了搭载 linux 操作系统的多 ARM 处理器冗余设计[26]。美国加州理工大学于 2013 年发射的 IPEX 卫星，星载计算机采用了 ARM9 处理器并搭载了 Linux 操作系统[27]。国内方面，国防科学技术大学的黄影在星载 COTS 星载计算机设计中采用 ARM9 处理器作为星载计算机并移植了经过裁剪的 Linux 操作系统[28]。

1.2 综合电子系统容错技术研究现状

1.2.1 太空辐射环境及危害

卫星在太空中会受到三种种类不同、来源不同的高能粒子冲击，按照高能粒子的来源不

同可分为太阳高能粒子、地球辐射带高能粒子和银河宇宙射线高能粒子[29]。

太阳粒子是指脱离太阳引力发射到太空中的高能粒子，这些高能粒子对太空中的航天器与宇航员构成严重的威胁。地球辐射带又称为范艾伦辐射带，由被地球磁场俘获后的太阳粒子构成，其中包含了高能质子与电子。银河宇宙射线是来自太阳系外的外太空带电高能粒子组成，这些粒子的能量超过1000MeV，因此常规的防护材料及手段无法满足卫星综合电子系统抗高能粒子辐射需求[30-31]。

当卫星处于太空辐射环境中，综合电子系统中电子设备受到多种高能粒子冲击，会产生一定的辐射效应影响。按照辐射效应引起的机理不同，辐射效应主要分为辐射总剂量(Total Ionizing Dose,TID)和单粒子效应(Single-event effect,SEE)，其中单粒子效应主要表现为单粒子闩锁(Single-event-latch-up,SEL)和单粒子翻转(Single-event upset,SEU)两种危害[32-33]。

辐射总剂量主要用于描述元器件在发生不可逆故障前所能够吸收辐射总能量的级别。随着辐射的累计，辐射总剂量会引起元器件的物理特性下降，结构上采用屏蔽、加固的方法抗辐射[34]。

单粒子闩锁主要作用于CMOS器件[35]，由于电路的输入端或输出端引入了外来的噪声电流、电压，导致寄生的双极性晶体管形成闩锁导通，产生过大的瞬时电流。这种高电流轻则导致电路或元器件无法正常工作，重则导致电子设备或元器件烧毁，形成永久性损坏。通过对关键电路设置限流电阻，设计断电重上电模块实现单粒子闩锁防护[36]。

单粒子翻转是卫星在轨故障的主要原因之一，相关文献中提及的对1971-1986年国外39颗同步卫星的故障统计[37]如表1.1所示。

表1.1 同步卫星故障统计表

故障类型	出现次数	故障率
电子诱发电磁脉冲	293	18.5%
静电放电	215	13.5%
单粒子翻转	621	39.0%
其他	1589	29.0%

可以看出单粒子翻转导致的故障概率占近40%。单粒子翻转主要是由于单个高能辐射粒子冲击到元器件内部引起器件内部的逻辑翻转、存储数据出错等功能暂时性缺失。随着元器件集成度的提升，单粒子翻转概率也随之增加[38-40]。单粒子翻转导致的故障属于可修复性故

障，可通过硬件冗余和数据编码等容错技术纠错[34]。

1.2.2 卫星容错技术

容错技术是指当卫星出现错误故障时，通过一定方法忽略错误状态或纠正错误结果，系统能以正常运行状态继续执行星载任务。容错技术主要分为硬件冗余、信息冗余、软件冗余等[41]，硬件冗余容错技术主要分为被动冗余、主动冗余两类[42]。

被动冗余是指多个相同的功能模块执行相同的任务，产生的操作结果经过仲裁器判决后输出结果。按照仲裁器逻辑的不同，可分为二模冗余和多余度冗余。二模冗余主要实现错误检测功能，两个相同的模块执行相同的操作，若仲裁器输出的结果不同，说明系统中有错误发生，此时系统进入错误自检与错误纠正流程。多余度冗余实现输出正确结果功能，多个相同的模块执行相同的操作并将操作结果输入仲裁器，仲裁器按照“多数优先”的表决机制判决输出结果[43]。由于资源限制，三模冗余是微小卫星综合电子系统常用的多余度冗余类型。

主动冗余是指多个相同的模块功能运行与结果输出由检测切换单元控制。检测切换单元通过故障管理逻辑选择多个功能相同模块中的一个输出结果。主动冗余的纠错能力较强。且在某一模块进入工作状态运行并输出结果时，其他模块处于备份状态[44]。

信息冗余是指在数据信息中加入冗余校验信息实现错误数据检测、纠正，常用的编码方式有汉明码、RS 编码以及循环冗余校验码(Cyclic Redundancy Check,CRC)[45]。冗余校验信息的编码方式不同，检错、纠错能力不同，冗余校验信息码元越多，检错、纠错能力越强。

软件冗余方式与硬件冗余类似，软件程序运行检测到故障出现时，冗余的功能模块接替发生故障模块工作，常见的软件冗余容错方式有 N 版本技术[46]和恢复块。

1.2.3 综合电子系统数据存储容错技术

数据存储器作为星载处理器程序运行与数据存储的重要载体，搭载操作系统的星载处理器因大容量数据存储需求对数据存储器的可靠性提出了更高的要求。为提升数据存储器的可靠性，减小单粒子翻转造成的危害，国内外的卫星研发团队都对数据存储容错技术进行了大量的研究与实践，并取得较多成果。

美国发射的 Alsat-1 卫星，使用 Intel 80C386EX 作为星载处理器并搭载了由萨瑞卫星科技公司设计的操作系统。卫星数据存储单元采用 32KB 大小的 EPROM 与 4MB 大小的程序存储

器[47]。处理器的固件信息存放于 EPROM 中，依靠 EPROM 自身的品质保证数据的可靠性。系统运行所需数据存放在采用了错误检测与纠正(Error Detection And Correct,EDAC)且硬件三模冗余的程序存储器中。为保证 RAM 数据传输过程中的可靠性，采用 RS(256,256)编码纠错方式，每 256 字节数据最大可纠错 2 字节。为保证程序数据传输过程的可靠性，数据采用(12,8)汉明编码方式，每 1 字节可纠错 1 位数据。

Maxwell 公司为满足航天领域高可靠性需求，研制的 SCS750 处理器单元，其核心处理器采用 PowerPC 750FX，数据存储器采用 256MB SDRAM 与 8MB EERPOM。SDRAM 数据由 FPGA 进行 RS 编码纠错，EEPROM 数据由 FPGA 采用汉明码实现 1 位纠错 2 位检错[48]。

美国发射的空间技术 5 卫星搭载了 Mongoose 5 抗辐射处理器，数据存储器包含 2MB EEPROM、40MB DRAM、32KB SRAM。DRAM 存储器采用 EDAC 技术纠错，EEPROM 分为不可擦写的 boot 区和可重写区，通过 EEPROM 器件自身的抗辐射性保护数据[12]。

中国科学院发射的“创新一号”卫星，是我国自行研制的第一颗重量低于 100kg 的微小卫星。“创新一号”数据存储器采用 PROM 与 RAM，其中 PROM 存放应用程序，RAM 用于存放程序运行时的数据，通过 EDAC 技术对 RAM 数据纠错[49]。

“天绘一号”卫星采用双冗余硬件备份的 128 MB Flash 存储器，可通过指令实现存储器切换。Flash 存储空间中前 96MB 作为数据存储区，后 32MB 为备份存储器。由于 Flash 在运行中会产生坏块，因此综合电子系统还设计了坏块管理与地址映射功能，并对重要的地址映射表进行三模冗余保护[11]。

“天巡一号”采用 EDAC 纠错、信息冗余校验（CRC 检验、奇偶校验等）实现数据存储器数据的错误纠正[16]。哈尔滨工业大学的李攀提出了一种基于 eMMC 星载大容量数据存储技术，其中提出了采用扩展汉明码加上硬件三模冗余方式实现卫星大容量数据的可靠存储[50]。

“萤火一号”火星探测器，在数据存储电路中采用了 EDAC 设计，数据存储时通过汉明码进行编码，实现数据一位可纠错，二位可检测功能。同时使用 Ada 语言编写的多任务并发星务软件，可在探测器任务空闲状态时对存储单元依次循环做出“读出-写入”对比，实现翻转错误数据纠错[15]。

嫦娥二号卫星为实现大数据量图像信息的存储，存储器使用 16 GB NAND Flash 并采用 4 级流水线方式提升数据写入速度。图像数据写入 NAND Flash 时由独立的 FPGA 进行 RS(256,256)编码[51]，并在卫星入站回放数据时使用 RS(256,256)解码纠错，消除单粒子翻转造

成的数据错误[52]。

可以看出国内外卫星存储数据容错设计主要采用了高可靠性器件、信息冗余、硬件冗余容错技术。高可靠性器件是采用强抗辐射性的存储器件，以器件自身的品质保证数据的可靠性。信息冗余主要采用汉明码、RS 编码等 EDAC 技术实现数据传输、存储时单粒子翻转纠错功能。硬件冗余主要采用双机备份、三模冗余设计，通过指令切换与数据仲裁消除单粒子翻转所引发的数据错误。

1.2.4 综合电子系统系统容错技术

为提升综合电子系统的可靠性，研究人员探索如何通过容错技术实现综合电子系统对故障结果的屏蔽或纠错[5]，从而延长综合电子系统的生命周期。在综合电子系统容错技术方面，国内国外都做了大量的研究并取得了较好的成果。

Pignol 于 2005 年发表的一篇文章中针对系统单粒子翻转问题提出了多种容错技术，主要包括双机备份、三模冗余及其他容错方法等[53]。

Maxwell 公司设计的 SCS750 处理器单元采用对 PowerPC 处理器进行三模冗余仲裁的方式得出正确的处理器运算结果，提升整系统的可靠性[48]，冗余结构如图 1.2 所示。

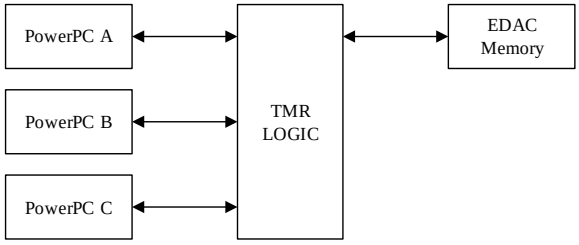

图 1.2 SCS750 处理器冗余结构

美国于 2007 年成功发射的 CFEsat 小卫星采用了基于 SRAM 型 FPGA 作为可重构星载计算机处理器。在硬件结构设计上采用了三模冗余方式，通过一块反熔丝 FPGA 不间断地读出三片 SRAM 型 FPAG 的状态信息进行仲裁判断，并通过 CRC 校验的方式检测是否有错误发生，若发生错误则立即重置 SRAM 型 FPGA[54]。

列日大学于 2010 年 10 月发射的 OUTFI1 卫星，该卫星的两块星载计算机互为热备份，并通过 I2C 总线进行数据通信。星载计算机通过总线采用交换“生命”信号的仲裁方式确定卫星的控制权[55]。

韩国航天研究所从 2012 年起连续发射了 3 颗低地球轨道卫星。卫星采用多级备份容错设

计，其中处理单元 PM32 采用双机冷备份容错设计，结构如图 1.3 所示。采用该容错设计，卫星 5 年任务周期结束后，预估整系统可靠性为 0.96[56]。

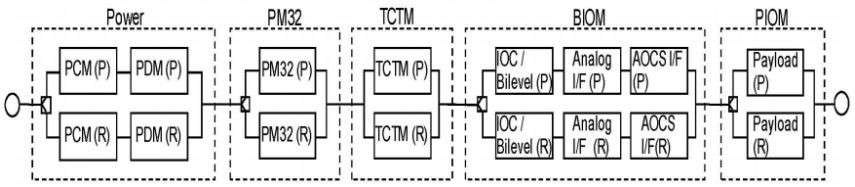

图 1.3 LEO 卫星双机备份容错结构

国内在系统容错的研究上也取得较多成果。“创新一号”采用双机热备份模式，两个星载计算机同时挂载在内部总线上通过双机通信 FIFO 实现星载计算机间的通信，并由独立的判决逻辑控制器实现双机工作状态的切换[49]，结构如图 1.4 所示。

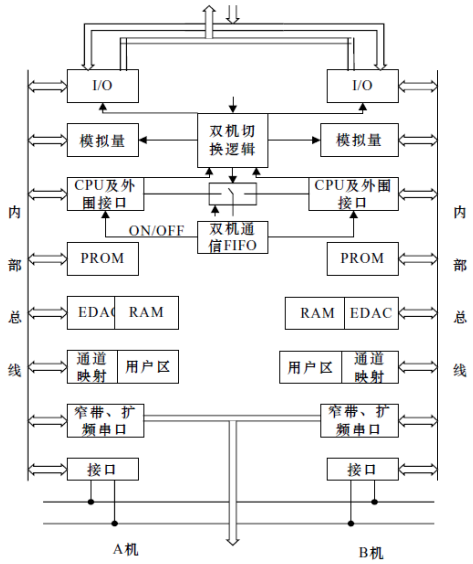

图 1.4 “创新一号”双机备份结构

“萤火一号”卫星采用星载计算机双机冷备方式，通过特殊方法将两套 CPU 总线耦合成一套 CPU 总线，所有功能模块均挂载在公共总线上[15]。仲裁器通过控制星载处理器电源实现双机切换，结构如图 1.5 所示

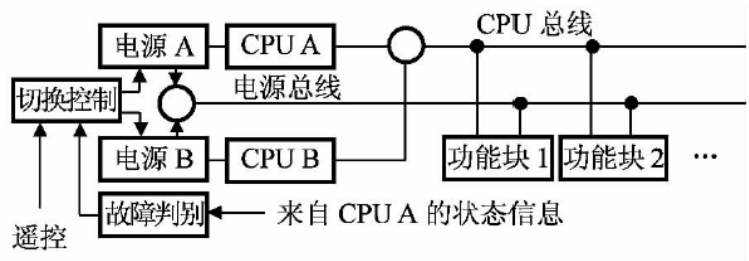

图 1.5 “萤火一号”仲裁结构

浙江大学于 2010 年成功发射自主研制的“皮星一号 A”卫星采用双机热备份结构，由

CPLD(Complex-Programmable-Logic-Device)执行故障检测与双机切换[57]。哈尔滨工业大学于2015年发射的“紫丁香二号”卫星，采用ARM处理器与FPGA双模冗余设计。ARM处理器默认为主机并采用了检测到主机故障后，仲裁模块切换备份工作系统的管理方式[58]。

2016年南京理工大学自主研制的“立方星”，采用双机冷备份模式，两个处理器共享一个数据存储器，通过仲裁器实现双机切换管理[59]，结构如图1.6所示。2019年哈尔滨工业大学的吴生龙设计了一种基于商用元器件的双机温备份星载计算机容错设计。设计采用看门狗检测的仲裁方式实现星载处理器的冗余备份[60]。

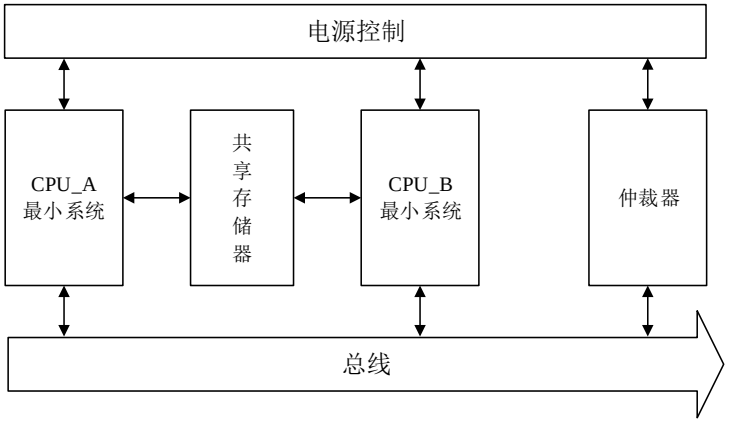

图1.6 “立方星”双机备份结构

国内卫星在双机备份容错方式的原理与实践方面都取得了不错的成果。在三模冗余设计方面，哈尔滨工业大学研制的三模冗余计算机，使用1553B总线与其他分系统计算连接，组成局域系统。2010年，中国科学院软件研究所设计了一种基于国产的BM3803处理器的星载计算机架构，采用了寄存器三模冗余的容错设计[61]。

从上述现状调研中可以看出，卫星系统容错技术主要采用双机备份容错或三模冗余容错，冗余备份管理方式主要采用电源控制及工作状态选择。其中双机备份容错技术使用较多，三模冗余容错技术使用较少，主要因为三模冗余硬件结构复杂、功耗高、辐射积累量会影响器件寿命且仲裁过程需要数据与时间高度同步，不适用于资源有限、低功耗需求的微小卫星。

1.3 研究内容及章节安排

本论文依托ZHX项目，根据ZHX项目支持操作系统、在轨智能计算、在轨可更换升级等功能需求，主要研究内容从系统容错架构设计、基于操作系统的星载可靠性加载设计以及系统级冗余备份管理方案三方面展开，针对星载综合电子系统容错能力提升进行研究。

本文在结构上共分 6 章，各章的内容具体如下：

第一章为绪论，阐述了微小卫星综合电子系统架构及星载处理器的发展方向，调研卫星容错技术的研究背景、国内外研究现状，并根据调研内容分析总结卫星数据存储及综合电子系统冗余备份使用容错技术，最后介绍本文研究内容及本文、章节安排。

第二章为综合电子系统总体方案，本章从 ZHX 综合电子系统的功能需求及性能指标出发，设计综合电子系统双机异构冗余备份，并介绍了综合电子系统硬件架构设计及功能单元的划分。针对容错总线需求，设计可靠总线、高速总线及备份总线相结合的总线设计并根据接口特殊功能需求设计接口转换与扩展模块。

第三章为操作系统的星载可靠性加载，本章介绍了可靠性模型分析，并根据模型分析确定存储数据的容错技术路线。结合 RT-Linux 操作系统的启动过程及特点设计了三种基于不同接口的大容量操作系统数据的靠性加载设计。

第四章为双系统冗余备份管理方案研究，本章介绍了双机异构备份冗余架构设计下多处理器竞争所引发的管理权冲突问题，并针对此问题设计了主从状态管理。结合主从管理提出了两种双系统冗余备份管理方案，实现故障下系统管理权的平稳过渡，并从实时性、可靠性及工作状态几个方面分析评估两种冗余备份管理方案。最后介绍了双系统重要信息同步机制，实现主从系统公有信息的实时同步。

第五章为综合电子系统容错技术测试与分析，本章结合 ZHX 综合电子系统原理样机及双系统冗余备份管理方案研究原理样机进行容错技术与管理方案的功能与性能测试，包括一、操作系统星载可靠性加载设计的完备性测试、速度性能测试、资源消耗测试，并从翻转纠错耗时、资源消耗、硬件可靠性、可移植性几个维度分析评估三种可靠性加载设计。二、双系统冗余备份管理方案的故障状态下主从状态切换功能测试，。

第六章为总结与展望，首先对本文的工作和成果进行总结，并对操作系统星载可靠性加载设计的功能拓展与双系统冗余备份管理方案的逻辑完善进行展望。

2 综合电子系统总体方案

本章从 ZHX 综合电子系统主要性能指标出发，针对 ZHX 卫星支持操作系统、在轨智能计算、大容量数据传输、在轨可更换升级等功能需求，设计综合电子系统的冗余备份架构，并介绍综合电子系统硬件架构及架构内主要电子设备的器件选型、主要功能模块划分与容错总线设计。

2.1 功能及性能指标

2.1.1 综合电子系统主要功能

卫星综合电子系统功能包括星务程序管理，遥控指令接收、执行，实时遥测与延时遥测下发，整星实时时钟广播与校正，工程遥测数据收集，健康信息检测，间接指令输出，卫星在轨智能计算等。

(1) 卫星星地测控综合管理。各分系统数据信息的处理，遥控数据接收、执行与转发，整星遥测数据的预处理及组帧下发。

(2) 卫星功能综合管理。星上实时时间的地面授时校正与整星实时时间同步，功能载荷的任务调度，电源、温控管理等。

(3) 卫星在轨智能计算功能。星载智能加速平台实现人工智能算法，执行卫星在轨智能任务。

(4) 卫星运行状态自主管理。根据卫星运行过程中的工程信息调整配置状态，如通信故障时的总线切换、关键数据发生单粒子翻转故障时 FPGA 可重构纠错等。

(5) 姿轨控组件数据处理与指令控制。姿轨控分系统的数据集中在综合电子系统处理，并根据处理结果返回指令控制姿轨控组件状态。

2.1.2 综合电子系统主要性能指标

综合分析综合电子系统所需实现的主要功能并结合已有的卫星研制经验，制定综合电子系统的主要性能指标，主要性能指标内容如表 2. 1 所示。

表 2.1 综合电子系统主要性能指标

名称	性能指标
星载处理器	处理器主频高于 1.3GHz，可搭载操作系统。 内存：≥ 512MB，数据存储器 ：≥ 1GB。 支持 SPI、UART、I2C、MMC/SD、GPMC 等接口。
FPGA	具有强抗辐射能力。 等效系统门≥800000 个。 寄存器≥5000 个，组合逻辑单元≥10000 个 可用 RAM 资源≥150000bit。
智能加速平台	支持通用人工智能算法框架。 浮点计算能力≥0.8TeraFLOPS。
卫星实时时钟管理	独立计时单元，系统掉电可继续工作。 时间精度不小于 10ms，校时误差不大于 3ms。
CAN 总线	双路冗余备份，通信速率 500Kbps。
UART	支持 115200、921600 等多种波特率。
SPI	最大支持 48Mbps 传输速率，支持 RS422 差分接口。
I2C	支持最大 400KHz 时钟速率。
高速 Ethenet 总线	双路冗余备份，通信速率 100Mbps。
健康信息检测	6 路电流采集、2 路温度采集、处理器状态、硬件状态信息。
关键数据纠错	三模冗余修复故障，可靠性加载纠错用时≤120s
工作模式	系统可更换升级，系统级冗余备份。
功耗	工作模式下≤14W。

2.2 综合电子系统架构

针对 ZHX 卫星综合电子系统在轨可更换升级的功能需求，结合卫星的资源与容错技术研究，综合电子系统采用双机异构温备份冗余架构[62-63]，实现系统级冗余备份，提升综合电子系统可靠性。ZHX 卫星搭载两块异构的综合电子系统板卡 A 板卡、B 板卡，两块板卡上的综合电子系统功能相同。两块板卡以接插件的形式安装在指令执行单元上，结构如图 2.1 所示。

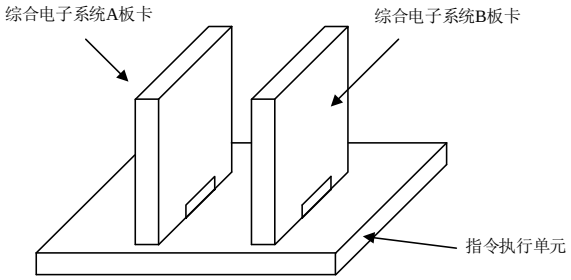

图 2.1 双系统冗余备份结构图

2.2.1 硬件总体设计方案

ZHX 综合电子系统硬件设计主要分为星务处理单元、FPGA 单元、智能计算单元、综合电子系统存储器组四个部分，综合电子系统结构如图 2.2 所示。

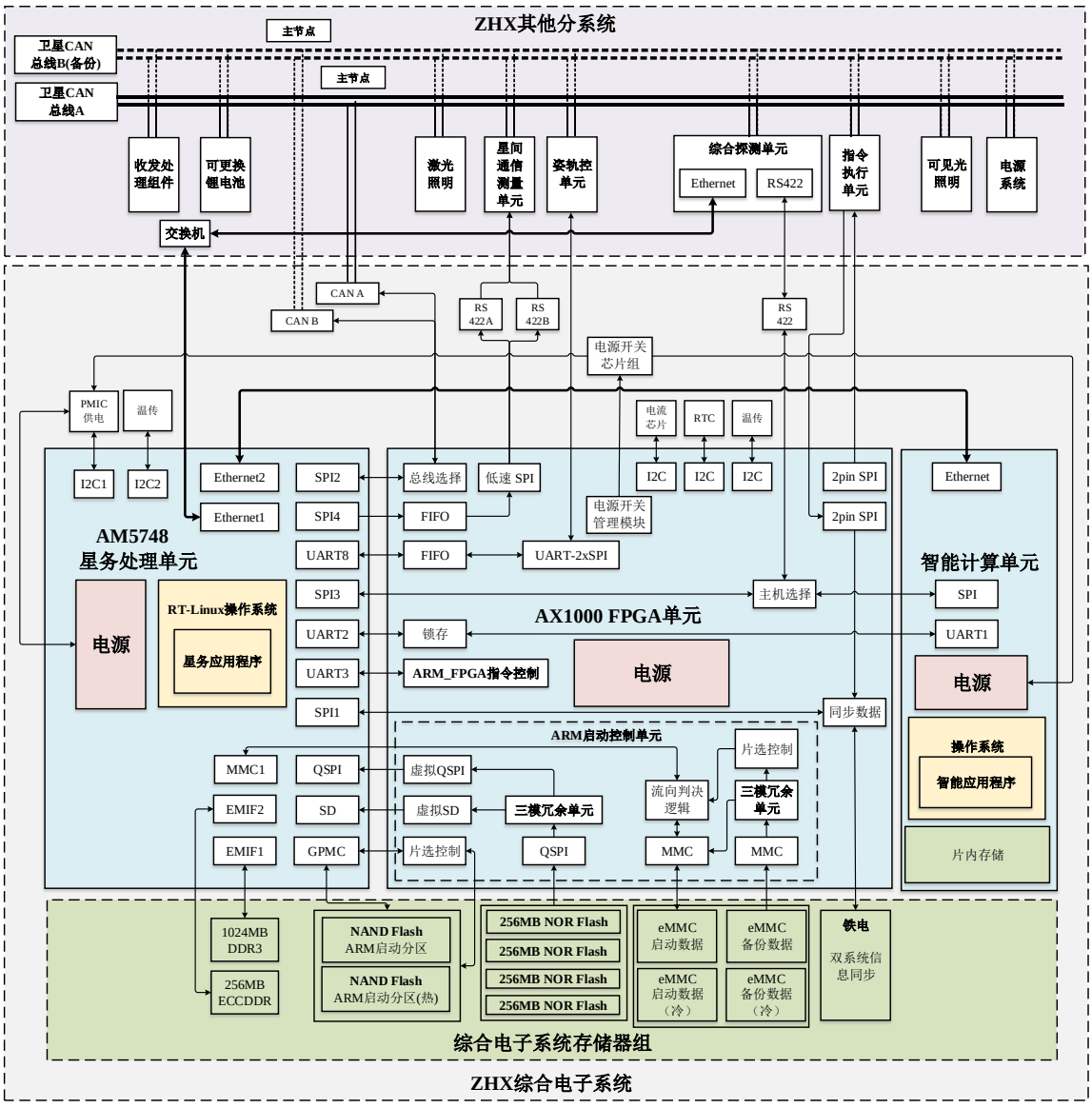

图 2.2 ZHX 综合电子系统结构

(1) 星务处理单元

星务处理单元是综合电子系统中最重要的功能模块，负责综合电子系统与卫星其他载荷系统的信息交互，姿轨控分系统的数据处理、各分系统工程遥测信息的采集，下行遥测的组帧传输等卫星任务。星务处理单元采用搭载 RT-Linux 操作系统的 ARM 处理器。结合综合电子系统对处理器的性能需求，星务处理单元 ARM 处理器采用 TI 公司的 AM5748 处理器。AM5748 处理器接口丰富、功耗低，支持 ECC DDR，主要性能指标如表 2.2 所示：

表 2.2 AM5748 处理器主要性能指标

序号	名称	性能指标
1	核心处理器	ARM Cortex A15
2	协处理器	ARM Cortex M4C66x
2	主频	1.5GHz
3	功耗	2.4W

(2) FPGA 单元

FPGA 单元使用反熔丝架构的 FPGA[64]。反熔丝架构 FPGA 只支持一次编程烧写，程序烧写完成后不可更改。LUT 表的配置信息具有非易失性，掉电后不丢失。在太空高能粒子辐射环境中反熔丝 FPGA 的抗辐射性能强。反熔丝 FPGA 的 LUT 表配置逻辑一旦烧写不可改变，几乎不受单粒子翻转影响。

反熔丝 FPGA 的选型主要考虑器件性能与可购买性，由于反熔丝 FPGA 主要用于航天与军工领域，因此性能较好的反熔丝 FPGA 一直受禁售限制。根据综合电子系统对 FPGA 的性能指标要求，反熔丝 FPGA 采用 Microsemi 公司的 AX1000，主要性能指标如表 2.3 所示

表 2.3 AX1000 主要性能指标表

序号	名称	性能指标
1	等效系统门数量	1000000 个
2	标准门数量	612000 个
3	寄存器数量	6048 个
4	组合逻辑单元	12096 个
5	最大触发器数量	12096 个
6	RAM 资源总量	165888 bit

FPGA 的功能模块如图 2.3 所示。FPGA 主要实现了整星实时时钟广播与校时、冗余硬件管理、指令解析与执行、健康信息检测、接口转换与扩展、操作系统可靠性加载、系统冗余备份管理等功能。

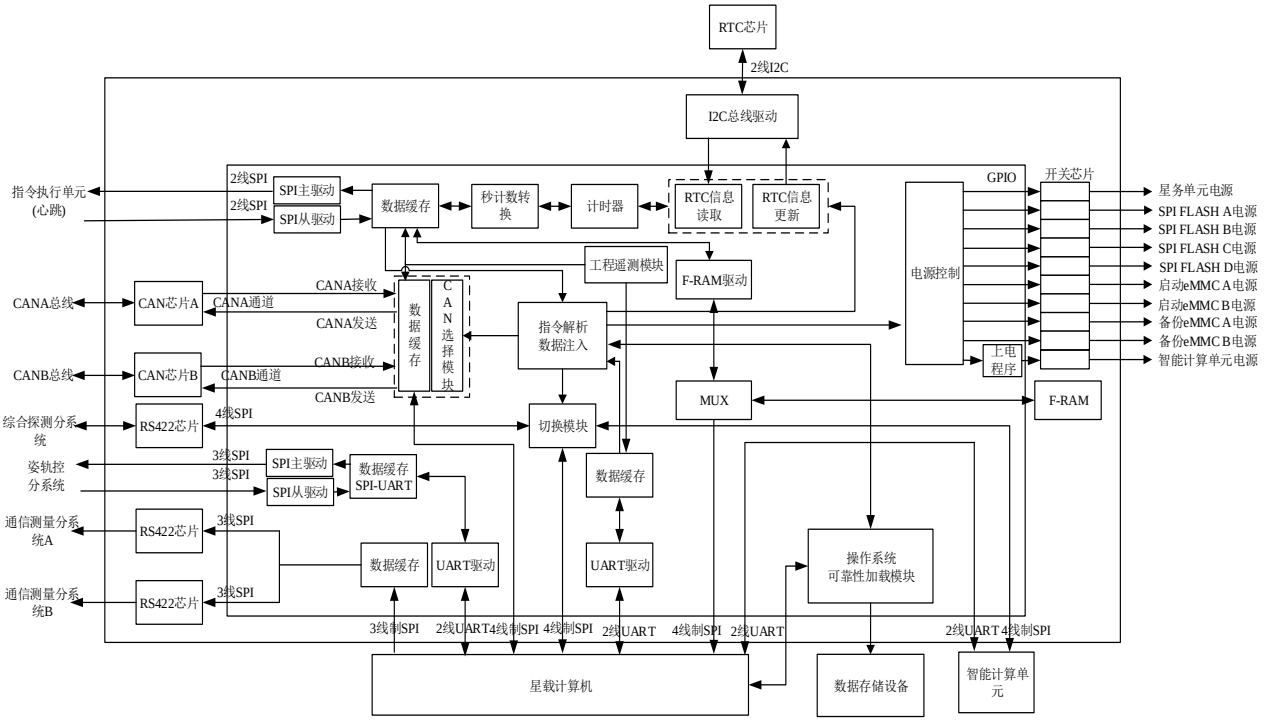

图 2.3 FPGA 功能模块图

1) 整星实时时钟广播与校时

整星实时时间采用基于星上实时时钟(Real Time Clock,RTC)实现。FPGA 读取 RTC 获取实时时间信息，并在 ARM 处理器的管理下，按节拍通过 CAN 总线广播实时时间信息，各分系统根据 CAN 总线上的时间信息同步系统时钟。地面授时，由 FPGA 校正 RTC 内时间信息。

2) 冗余硬件管理

综合电子系统采用硬件冗余备份的方式提升元器件的可靠性。部分器件采用冷备份方式，因此综合电子系统需具备冗余硬件管理功能。由于反熔丝 FPGA 可靠性高于 ARM 处理器，且反熔丝 FPGA 在功能上可控制 ARM 处理器电源，功能逻辑优先级更高，因此由反熔丝 FPGA 实现综合电子系统的冗余硬件管理。

3) 综合电子系统健康信息检测

FPGA 通过温度传感器获取综合电子系统板卡的温度信息；通过电流管理芯片获取 6 路硬件设备的电流信息。FPGA 与部分元器件按协议通信获取元器件的状态信息。ARM 处理器通过 UART 串口以节拍形式与 FPGA 通信，以此作为心跳信号。FPGA 通过上述方法获取综

合电子系统健康监测信息。

4) 指令解析与执行

FPGA 接收的指令来源有两个，一是指令执行单元发送的直接指令，二是星务处理单元 ARM 处理器发送的间接指令。为减少 FPGA 指令解析的工作量，直接指令与间接指令采用相同的帧协议，FPGA 使用同一模块解析指令。

指令的来源不同，优先级也不同，直接指令优先级高于间接指令，在执行直接指令时，执行对应的元器件具有独占性，此时间接指令对相应元器件的控制操作将被 FPGA 忽略不执行。

(3) 智能计算单元

智能计算单元是综合电子系统的重要功能载荷，搭载了算力强大的终端平台，使卫星具备在轨智能计算能力。针对 ZHX 卫星器件国产化、系统可更换升级等需求，智能计算单元采用双系统异构设计。经调研后，采用 Nvidia 公司的 Jetson TX2 核心板[65]作为 A 板卡智能计算单元，采用寒武纪公司 1H8 核心板作为 B 板卡智能升级单元，两种核心板的主要性能指标如表 2. 4 所示。

表 2. 4 智能计算单元性能指标表

参数名称	1H8	TX2
主 CPU	ARM Cortex A7	ARM A57 处理器
智能终端处理器	1H8 IP	Pascal 架构 GPU
内存	128MB DDR3	8GB LPDDR4
数据存储器	128MB NAND Flash	32GB eMMC
功耗	<1.2W	<7.5W
最高工作频率	1.0GHz	2.0GHz
计算能力	0.8 TeraFLOPS	1.5 TeraFLOPS

(4) 综合电子系统存储器组

搭载 RT-Linux 操作系统的 ARM 处理器所需的数据存储容量大，传统卫星采用 EERPOM 或小容量 Flash 数据存储器的设计方案无法满足需求。综合考量数据存储器的容量与读写速度，采用大容量 Flash 作为数据存储器介质并设计数据存储器组，为操作系统可靠性加载设计提供硬件支持。数据存储器信息如表 2. 5 所示。

表 2.5 综合电子系统数据存储器信息

数据存储器类型	容量	数量
SPI NOR Flash	256MB	4
NAND Flash	2GB	2
启动 eMMC	8GB	2
备份 eMMC	32GB	2
DDR3	512MB	2
ECC DDR3	256MB	1
F-RAM	512KB	1

为使 ARM 处理器能最大程度发挥性能，内存采用两片 512MB DDR3 组合使用，可提供 1GB 的 RAM 存储空间。RAM 介质抗单粒子翻转能力差易受单粒子翻转影响，因此采用一片 256MB 的 ECC DDR 对内存数据进行 ECC 纠错。

2.2.2 总线设计

ZHX 卫星的总线采用低速通信总线、高速通信总线及备份通信总线相结合的设计方案[66]。低速总线传输遥控、遥测、整星实时时钟等对可靠性要求较高的数据，采用 500K 的高速 CAN 总线作为卫星低速通信总线。针对大数据量文件及图像信息传输的需求，由于星务处理单元和智能计算单元都拥有搭载操作系统的 ARM 处理器，因此高速通信总线采用百兆带宽的 Ethernet。备份通信总线作为低速总线或高速总线出现故障时，综合电子系统与卫星重要分系统点对点通信的应急总线。

(1) 低速通信总线

控制器局域网络(Controller Area Network,CAN)总线数据传输采用差分电平及显性逻辑电平判决方式。基于此电气特性，CAN 总线支持总线仲裁，报文标志符低的节点获得总线控制权。CAN 总线不同节点在总线传输都能获得相同的数据，数据传输实时性高，并容易形成冗余结构，大大的提升了总线的可靠性、鲁棒性与灵活性。低速通信 CAN 总线由 CAN_A、CAN_B 两条 CAN 总线组成，采用双总线冗余热备份结构，提升低速通信总线可靠性[67]。

综合电子系统负责遥控、遥测信息等数据的收集与处理，因此在卫星低速通信总线协议规定综合电子系统的 CAN 节点报文标志符最低，仲裁优先级最高，所有数据传输事务都由综

合电子系统发起，其他分系统仅作为从机应答。

综合电子系统处于正常工作模式时，ARM 处理器通过 CAN 总线以时间片形式按节拍完成以下工作：

1) 发起数据传输事务，按顺序依次请求卫星各分系统返回工程遥测信息。

2) 完成遥测数据组帧、打包并发送给通信测量分系统。

3) 检测 CAN 总线通信是否正常工作。

4) 星上实时时钟发送控制

硬件上选用 MCP2515 做为 CAN 总线控制器芯片，减少 ARM 处理器与 FPGA 对 CAN 总线的管理工作。

(2) 高速通信总线

Ethernet 是一种应用广泛的局域网通讯方式，硬件上通过交换机实现综合电子系统内的星务处理单元、智能计算单元与综合探测分系统、姿轨控分系统之间的大数据量信息交互。硬件上采用双总线冗余冷备份方式提升高速总线的可靠性。

为保障通信数据的正确性，降低电路中的干扰对通信数据信号的影响，星务处理单元与智能计算单元的数据通路都经过反熔丝 FPGA。但由于百兆 Ethernet 通信速度快，通信协议内容复杂，反熔丝 FPGA 性能有限无法准确保证通信数据信号的正确性与完整性，因此 Ethernet 的数据通路不经过 FPGA。

(3) 备份通信总线

当 CAN 总线或 Ethernet 高速总线出现故障无法正常传输数据时，综合电子系统通过备份通信总线与卫星主要分系统点对点的传输数据。综合电子系统与主要分系统备份总线接口类型如表 2.6 所示。备份通信总线为低速、高速总线的备份，因此备份总线自身不再冗余备份。

表 2.6 备份总线接口类型

	指令执行单元	综合探测分系统	姿轨控分系统	通信测量分系统
星务处理单元	两路 SPI	标准 SPI	UART-两路 SPI	两路 SPI
智能计算单元	/	标准 SPI	/	/

2.2.3 接口转换与扩展

ARM 处理器硬件接口资源有限，卫星分系统接口数量众多且部分接口功能 ARM 处理器

难以实现，因此由 FPGA 实现接口转换与扩展，减轻 ARM 处理器资源负担，提升 ARM 接口驱动可靠性。

(1) 综合电子系统与姿轨控分系统接口转换

综合电子系统与姿轨控分系统的备份总线传输数据时，为保证两个系统都能主动发起数据传输流程，硬件接口上使用两路单工 SPI 交互数据，接口信息如图 2. 4 所示。SPI_A 接口中姿轨控分系统为主机，综合电子系统为从机。由于 ARM 处理器搭载的 RT-Linux 官方接口驱动中未提供 SPI 从机驱动，个人开发的 SPI 从机驱动，频繁出现数据丢失现象，数据传输可靠性低，因此由 FPGA 完成接口转换。

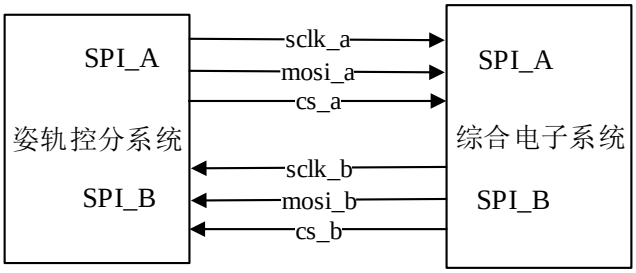

图 2. 4 综合电子系统与姿轨控系统 SPI 接口图

同步传输时由于随路时钟的存在，数据传输过程中总会存在主机与从机，因此采用异步传输的数据收发方式。姿轨控分系统与综合电子系统之间数据传输速率约为 1Mb/s，且数据收发量较小，因此使用波特率为 921600 的通用异步收发传输器（Universal Asynchronous Receiver/Transmitter,UART)实现接口转换，SPI 接口转换 UART 模块结构图如图 2. 5 所示。

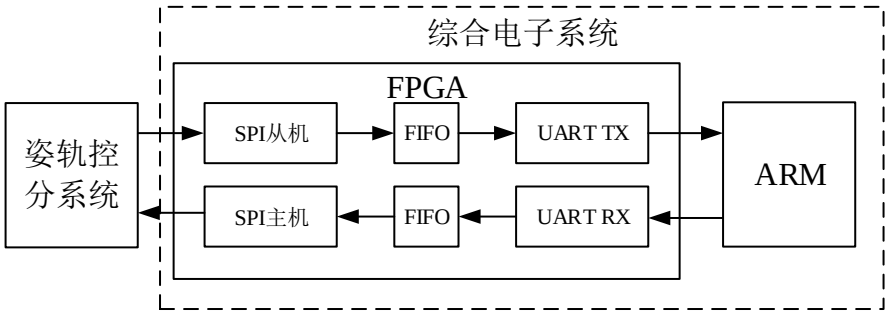

图 2. 5 SPI 转 UART 模块结构

FPGA 将到来自 ARM 处理器 UART 接口发送的数据存入先进先出存储器(First-In-First-Out,FIFO)中，并同通过空标志触发 SPI 主机将所接收到的数据发送给姿轨控分系统。反熔丝 FPGA 接收到来自姿轨控分系统 SPI 接口发送的数据，将数据存入 FIFO 中，通过空标志触发 UART_TX 模块将接收到的数据发送给 ARM 处理器。

(2) 综合电子系统与通信测量单元接口转换

综合电子系统与通信测量单元采用单工 SPI 的方式传输数据。在卫星分系统接口设计中规定了通信测量单元的 SPI 从机接口时钟速率为 10MHz，由于数传的传输数据量大，若 ARM 处理器以 10MHz 的时钟速率传输数据，将导致 ARM 处理器的数传 SPI 主机接口长时间占用 CPU 资源，降低处理器的整体效率。因此将 ARM 处理器数传 SPI 主机接口的时钟频率提升至 48MHz，并通过反熔丝 FPGA 实现数传 SPI 接口的频率转换与接口扩展。数传通路的接口结构图如图 2. 6 所示，反熔丝 FPGA 将一路 48MHz 的单工 SPI 接口转换为两路 10MHz 的单工 SPI 接口。

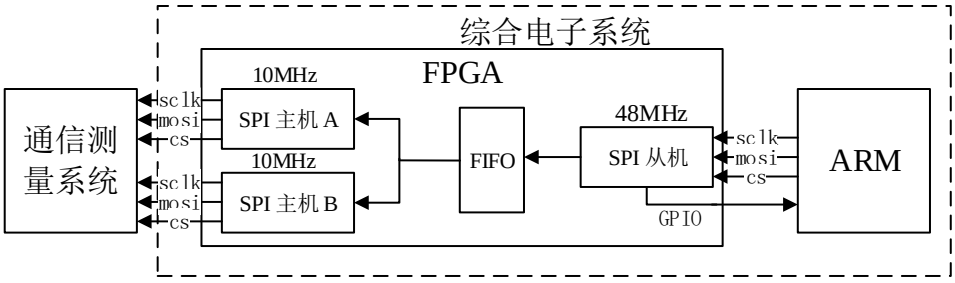

图 2. 6　48MHz SPI 转 10MHz SPI 模块图

反熔丝 FPGA 将 ARM 处理器 48MHz SPI 接口发送的的数据存入 FIFO 中缓存，并触发两个 10MHz SPI 接口模块，将数据转发至通信测量系统。

FIFO 通过反压的方式，中断 ARM 处理器开始、停止数据传输，保证 FIFO 不为空满状态。ARM 处理器程序通过中断响应的方式，实现 SPI 接口数据的发送与停止。经实验测得，ARM 处理器接收到来自 FPGA 的中断信号到响应中断实现数据的发送与停止，大约需要 170μs，因此 FIFO 需要留出足够的数据缓存余量，避免数据溢出或读空。

2.3 本章小结

本章详细地介绍了 ZHX 综合电子系统架构设计。首先介绍了 ZHX 综合电子系统的功能及性能指标，接着根据需求设计综合电子系统的冗余备份结构，并介绍了综合电子系统的硬件架构以及架构内主要电子设备的器件选型，然后介绍系统内主要功能模块划分，最后介绍了系统的容错总线设计与特殊功能接口设计。

3.操作系统星载可靠性加载设计

存储数据在太空中易发生单粒子翻转，当操作系统数据发生单粒子翻转，可能会引起ARM 处理器故障。传统卫星采用 EEPROM、PROM 等抗辐射性强的数据存储器，但这类存储器数据存储容量小难以满足大容量数据存储的需求。而目前大容量存储器抗辐射性差，易受单粒子翻转影响。本章结合 RT-Linux 操作系统启动方式，提出三种基于三模冗余的操作系统星载可靠性加载设计，提升系统可靠性。

3.1 可靠性模型分析

可靠性模型是对模块与系统可靠性估算所建立的数学模型，可以直观的对模块与系统可靠性进行定量分析，帮助寻找可靠性薄弱的环节，为可靠性设计提供思路。可靠性模型主要分串联结构模型、并联结构模型、仲裁结构模型三种类型[59]。

可靠性模型分析的假设条件如下：

（1） 模型内部各模块的可靠性概率服从失效率为 λ_i 的泊松分布，可靠性为 $P_i(t)=e^{-\lambda_i t}$

（2） 模型内部各模块正常工作的过程为相互独立事件。

串联结构模型由 n 个模块构成，当模型内任意模块发生故障都会导致结构出现故障，串联结构模型的可靠性概率 $P_s(t)$ 为公式(3.1)所示。

$$P_s(t)=\prod_{i=1}^{n}P_i(t)=e^{-\left(\sum_{i=1}^{n}\lambda_i\right)t} \tag{3.1}$$

从公式(3.1)看出串联结构模型的故障率大于模型内任一模块的故障率，因此在设计时应尽量避免使用串联结构模型。串联结构模型的结构如图 3. 1 所示。

图 3. 1 串联结构模型

并联结构模型有 n 个模块构成，当模型内所有模块都发生故障时，并联型结构模型才发生故障，并联结构模型的故障率 $P_P(t)$ 为公式(3.2)所示。

$$P_P(t)=1-\prod_{i=1}^{n}\left(1-P_i(t)\right) \tag{3.2}$$

从公式(3.2)看出并联结构模型的故障率小于模型内任一模块的故障率，因此在设计时模块应多采用并联结构模型。并联结构模型的结构如图 3.2 所示。

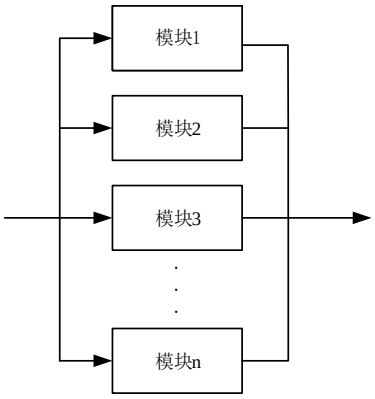

图 3.2 并联结构模型

仲裁结构模型由 n 个模块组成，当模型内至少 k 个模块正常工作，仲裁模型不输出故障结果，仲裁结构模型的可靠概率 $P_a(t)$ 为公式(3-3)所示。

$$P_a(t)=\sum_{m=k}^{n}\left[C_n^m P_i^m(t)\left(1-P_i(t)\right)^{n-m}\right] \tag{3.3}$$

仲裁结构模型的结构如图 3.3 所示。数据纠错及高实时性系统冗余主要采用仲裁结构模型。

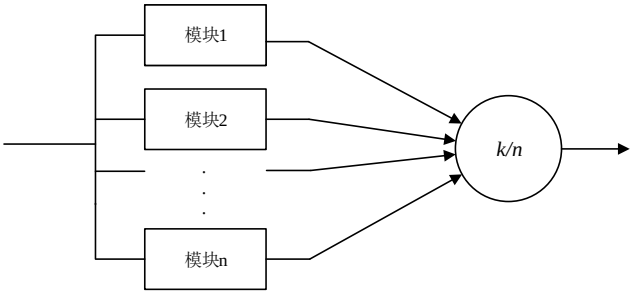

图 3.3 仲裁结构模型

3.2 存储数据三模冗余纠错机制

随着卫星在太空中执行任务时长的增长，综合电子系统数据存储器中的数据发生单粒子翻转概率随之增加。当操作系统数据发生单粒子翻转致使星载处理器出现故障时，系统将立即执行翻转数据纠错流程，纠正发生了单粒子翻转的操作系统数据，使星载处理器可靠加载操作系统。

翻转数据纠错主要采用备份数据重新覆盖方法，从备份数据存储器中读取备份数据并烧写覆盖发生单粒子翻转的原数据。由于备份数据存储器中存放的备份数据也会受到单粒子翻

转的影响，因此将备份数据分成相同的三份，并采用三模冗余(Triple Modular Redundancy,TMR)容错技术输出无单粒子翻转的备份数据。

三模冗余是一种仲裁纠错容错技术[68]，仲裁结构如图 3.4 所示。三份数据输入仲裁器，仲裁器按仲裁逻辑输出结果。

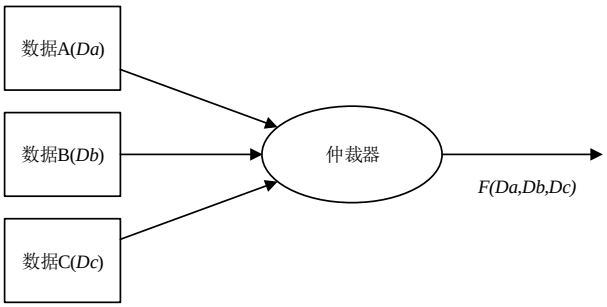

图 3.4 三模冗余仲裁结构

翻转数据纠错三模冗余的数据来源如图 3.5 所示，从内容相同的三份备份数据 A、B、C 的相同 bit 位取出数据D_a，D_b ，D_c。正常情况下D_a，D_b，D_c的值是相同的，当某一份数据受单粒子翻转的影响，数据值发生了变化，输入到仲裁器后，仲裁器仍能按照“多数优先”的表决机制输出正确值。

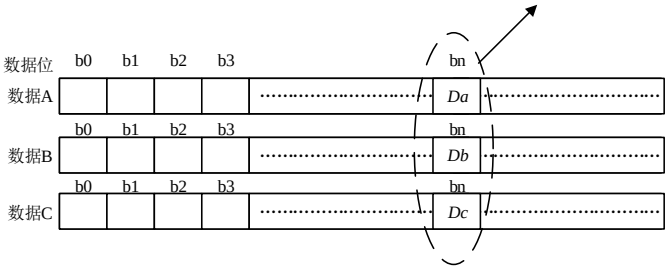

图 3.5 三模冗余数据来源图

由仲裁逻辑可得，当三份相同数据中存在两份及以上的同一 bit 位数据发生单粒子翻转，仲裁器将无法输出正确数据值，三模冗余失效。根据仲裁结构模型结合单粒子翻转概率可计算备份数据损坏概率。在太空环境下，三份备份数据的同一 bit 位数据第T天的翻转概率$R_{SEU}(T)$如公式(3.4)所示，其中P_m为存储介质单粒子翻转概率。

$$R_{SEU}(T)=\sum_{i=1,i\in odd}^{T} C_T^i P_m{}^i \left(1-P_m\right)^{T-i} \tag{3.4}$$

根据仲裁逻辑和每 bit 翻转概率$R_{SEU}(T)$可得第 T 天长度为 N bit 数据的三模冗余有效概率$R_{valid}(T)$如公式(3.5)所示

$$R_{valid}(T)=\left[\sum_{i=0}^{1} C_3^i R_{SEU}{}^i(T)\left(1-R_{SEU}(T)\right)^{3-i}\right]^N \tag{3.5}$$

相关文献显示，SRAM 的翻转概率约在$10^{-6}bit/day$数量级[69]，推测 Flash 的翻转概率在10^{-9}至10^{-7}数量级之间[70]。备份数据大小约为 53MB，Flash 翻转概率为$1\times10^{-7}bit/day$时结合三模冗余有效概率$R_{valid}(T)$进行仿真，仿真结果如图 3.6 所示。

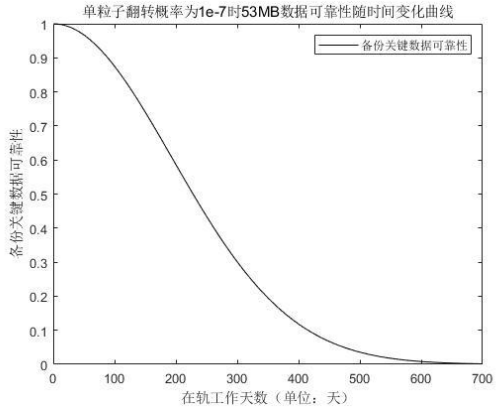

图 3.6 备份数据可靠性随时间变化图

从图中可以看出 Flash 大约经过 163 天，备份数据可靠性概率降为 70%。存放备份数据的存储器在卫星正常工作时处于断电状态，备份数据几乎不发生单粒子翻转，因此备份数据在轨工作天数为执行翻转数据纠错的天数。按三年的卫星任务周期计算，相当于卫星每 6.8 天可执行一整天的翻转数据纠错操作。

事实上，并不是 53MB 备份数据中任意一位数据损坏就会导致处理器无法正常工作，操作系统数据中混杂着一些不重要的数据，但由于无法细分出这部分数据的大小及存储位置，因此在分析备份数据可靠性概率时采用最悲观的方式。

3.3 基于 RT-linux 操作系统的启动方式

ARM 处理器在加载 RT-Linux 操作系统程序之前会从指定接口先加载一段 U-Boot 程序。U-Boot 程序本质上是一段裸机程序，主要完成计算机硬件的初始化，配置软件的运行环境，提供面向用户的命令式 shell 界面。ARM 处理器进入 U-Boot 阶段后，用户通过控制台命令可实现内存读写、访问数据存储器、加载操作系统等简单功能。执行加载操作系统命令时，U-Boot 程序会从选择的接口中引导操作系统加载。具体启动流程如图 3.7 所示

图 3.7 RT-Linux 操作系统启动流程

根据启动流程中 U-Boot 程序功能的不同，可分为一级 boot 和两级 boot 两种方式。

(1) 一级 boot 是正常启动操作系统的方式。U-Boot 程序和操作系统数据放在同一数据存

储器中。ARM 处理器加载 U-Boot 程序成功后，U-Boot 程序将自动执行 Autoboot 操作，ARM 处理器从同一数据存储器中加载操作系统。

(2) 两级 boot 是星载可靠性加载设计中重要的容错方式。U-Boot 程序和操作系统数据分别存储于不同数据存储器中。ARM 处理器加载 U-Boot 程序后可对不同数据存储器进行读写操作。该 boot 方式只需保证 U-Boot 程序完整正确，ARM 处理器通过运行 U-Boot 程序执行数据读写操作，完成翻转数据纠错，并最终实现操作系统可靠性加载。

AM5748 处理器支持 MMC，SD，QSPI，GPMC，USB 等多种接口启动操作系统，通过配置 system boot 管脚电平选择 ARM 处理器启动后加载 U-Boot 程序的接口。丰富的接口资源结合两种 boot 方式，可设计多种抗单粒子翻转的操作系统可靠性加载方案。

3.4 基于 eMMC 存储器的可靠性加载设计

搭载 RT-Linux 操作系统的 ARM 处理器所需数据存储容量大，传统卫星使用的高可靠性数据存储器无法满足大容量数据存储需求。因此数据存储介质采用 NAND Flash，但由于反熔丝 FPGA 性能有限，NAND Flash 的管理逻辑复杂，因此采用 eMMC 存储器以减少 FPGA 对 NAND Flash 的管理工作[71]。NAND Flash 抗辐射性能差，其中存储的操作系统数据易发生单粒子翻转，因此需采用容错技术纠正翻转数据，实现操作系统可靠性加载。

3.4.1eMMC 存储器介绍

eMMC (Embedded Multi Media Card）由硬件控制器和 NAND Flash 存储阵列组成，芯片内部的硬件控制器实现对 NAND Flash 的读写擦除、坏块管理、擦写均衡、地址映射、ECC 纠错、垃圾回收等管理工作[72]。由于芯片内部采用了 Cashe 缓存以及 Memory Array 技术，大幅度提升了 eMMC 读写速度，eMMC 结构如图 3. 8 所示。

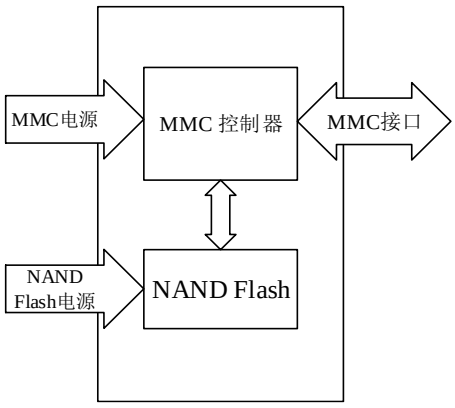

图 3. 8　eMMC 存储器结构图

3.4.2 基于 eMMC 的三模冗余设计

基于 eMMC 存储器的可靠性加载设计硬件结构如图 3.9 所示，启动 eMMC 和备份 eMMC 采用硬件冗余冷备份的方式提升容错设计的硬件可靠性。启动 eMMC 作为 ARM 处理器正常工作时的数据存储器，存放一级 boot 方式的 U-Boot 文件、RT-linux 操作系统与星务应用程序、应用相关文件。其中高可靠性要求的数据为 U-Boot 文件、RT-linux 操作系统与星务应用程序（共约为 53MB），这些数据称为综合电子系统关键数据。

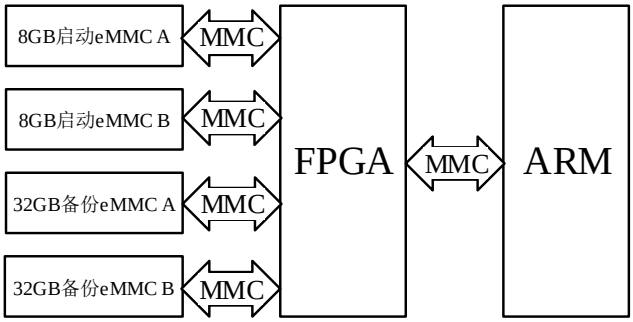

图 3.9 基于 eMMC 存储器的可靠性加载设计硬件结构

备份 eMMC 中存放备份关键数据，三份备份关键数据按不同地址区域存放于备份 eMMC 中。备份 eMMC 与启动 eMMC 数据存储结构如图 3.10 所示，图中的阴影部分表示无数据部分，作为数据之间的隔离保护带。

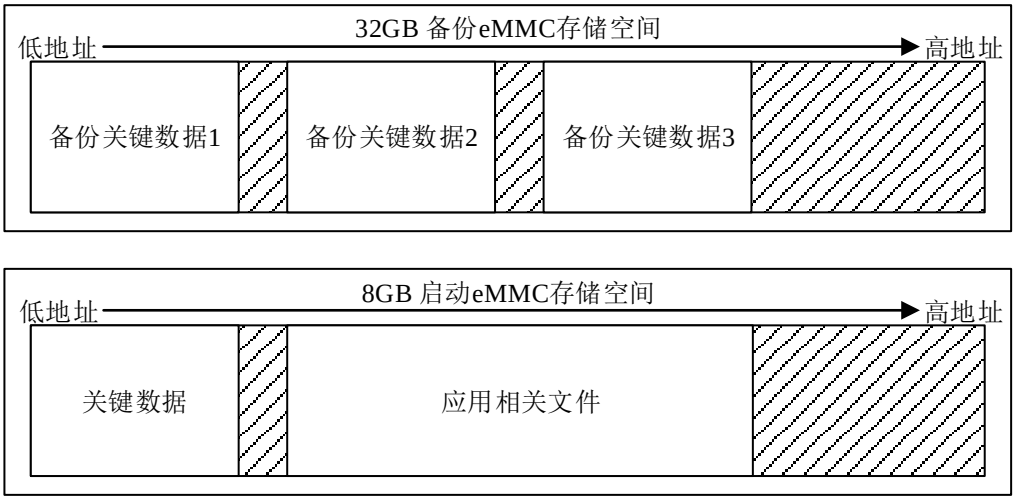

图 3.10 备份 eMMC 与启动 eMMC 数据存储结构

当启动 eMMC 中的关键数据发生单粒子翻转，ARM 处理器无法正常工作时， FPGA 从备份 eMMC 中读取三份备份关键数据并三模冗余仲裁出正确的备份关键数据，通过覆盖启动 eMMC 中发生单粒子翻转的关键数据[73]，实现翻转数据纠错。FPGA 翻转数据纠错模块如图 3.11 所示，FPGA 的 eMMC 控制器模块读取备份 eMMC 中的备份关键数据存入 FIFO 中缓存，备份关键数据三模冗余后存入 FIFO_TMR 中缓存，并通过 eMMC 控制器写入启动 eMMC。

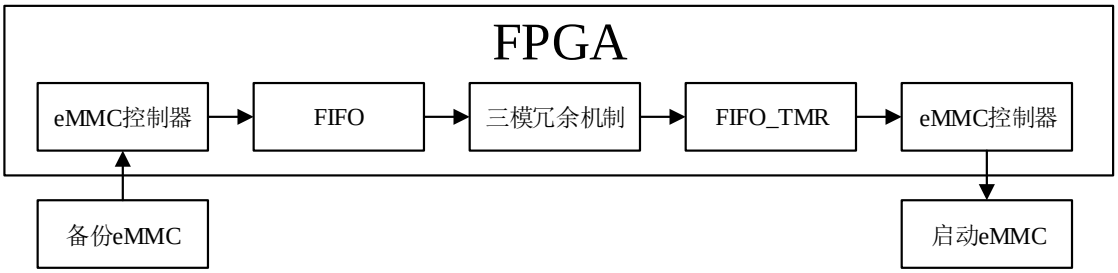

图 3.11 FPGA 翻转数据纠错模块

翻转数据纠错流程如图 3.12 所示。FPGA 根据设置片选寄存器的值选择使用备份 eMMC 与启动 eMMC。备份 eMMC 与启动 eMMC 都完成初始化识别后，备份关键数据进行三模冗余纠仲裁，仲裁输出的正确数据写入启动 eMMC。当关键数据纠错完成后， FPGA 重新建立 ARM 处理器与启动 eMMC 的数据通路。复位 ARM 处理器后操作系统将可靠性加载。该设计采用了一级 boot 方式，所有数据三模冗余仲裁及翻转数据纠错都由 FPGA 完成，ARM 处理器的 U-boot 程序不需要开发特定功能。

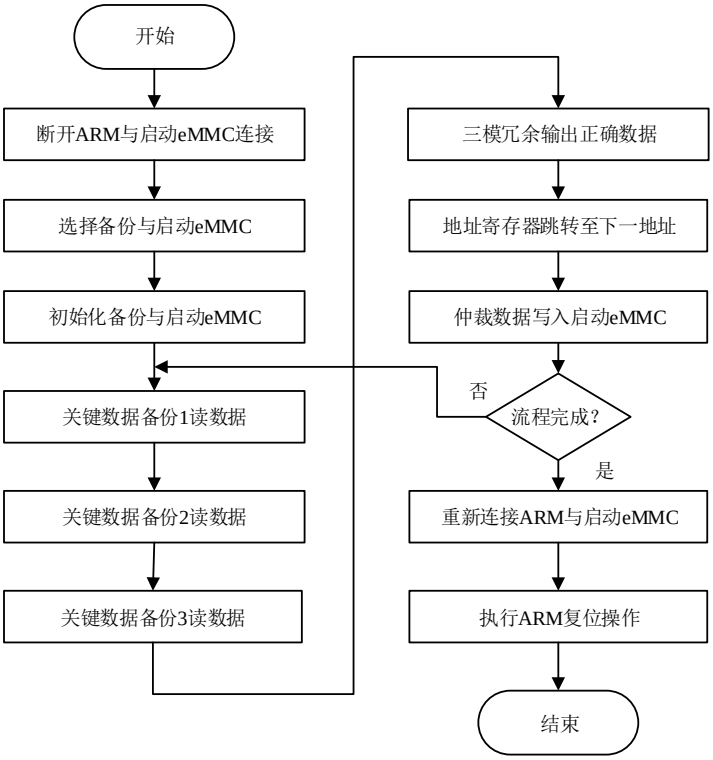

图 3.12 基于 eMMC 存储器的三模冗余纠错流程

3.4.3 基于 INOUT 接口的双向数据流向判决逻辑

综合电子系统正常工作时备份 eMMC 处于断电状态以降低功耗及减少单粒子翻转发生概率。此时 ARM 处理器作为通信主机，启动 eMMC 作为通信从机，FPGA 实现数据通路，通信数据流向如图 3.13 所示

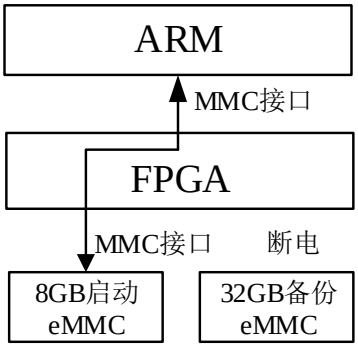

图 3. 13 ARM 与 eMMC 通信数据流向图

JEDEC eMMC 协议规定[74]，主机通过 CMD 管脚向从机发送命令，从机通过 CMD 管脚返回回复；读数据命令时，从机通过 DATA(0-7) 管脚返回数据；写数据命令时，主机通过 DATA(0-7) 管脚发送数据。eMMC 存储器接口引脚信息如表 3. 1 所示，从表中可得 CMD 引脚和 DATA(0-7)引脚为 INOUT 类型。FPGA 实现数据通路时，对应的 CMD 和 DATA(0-7)管脚状态也应为 INOUT 类型。

表 3. 1 eMMC 存储器接口引脚信息

名称	类型	功能
CMD	输入/输出	接收命令/发送回复
DATA(0-7)	输入/输出	发送/接收数据
CLK	输入	时钟
RST_n	输入	硬件复位
Vcc	电源	NAND Flash 电源
Vccq	电源	eMMC 控制器电源

FPGA 的 INOUT 接口通过三态门实现，三态门结构如图 3. 14 所示，FPGA 通过数据输出控制选择接口类型为输入或输出状态。ARM 处理器与启动 eMMC 通信时，FPGA INOUT 接口的输入输出状态需与双向的数据流向相同，否则会出现数据传输错误，ARM 处理器无法正确读写启动 eMMC 中的数据。

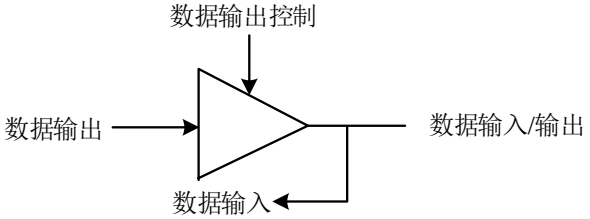

图 3. 14 三态门结构

本文设计一种双向数据流向判决逻辑，通过该判决逻辑可控制 FPGA INOUT 接口输入输

出状态与实际双向数据流向匹配，ARM 处理器可正确读写 eMMC 存储器数据且不产生时序上的延时。双向数据流向判决逻辑模块结构如图 3.15 所示。

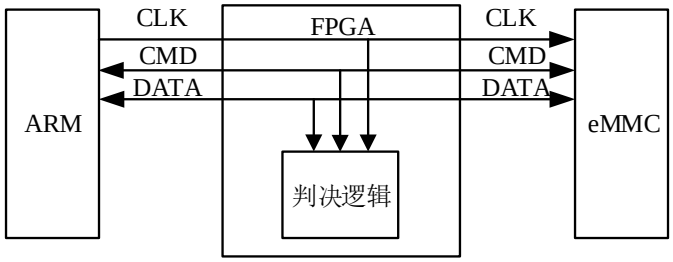

图 3.15 双向数据流向判决逻辑模块结构

判决逻辑作为数据传输通路上的“监测者”，监测通信过程中命令类型与数据传输情况并根据这些信息动态调整 INOUT 接口输入输出状态。判决逻辑通过采样 CMD 上的数据信息，解析主机发送命令的类型及参数，结合协议中定义的命令功能，预先判决出 CMD 与 DATA 的数据流向变化，并调整 INOUT 接口输入输出状态与接下来的数据流向相匹配。判决逻辑的时序如图 3.16 所示，图中的 Z 表示高阻态，D 表示传输的数据。CMD 输入输出状态信号为低电平时表示命令发送，为高电平时表示回复返回。DATA 输入输出状态信号为低电平时表示读数据返回，为高电平时表示写数据发送。

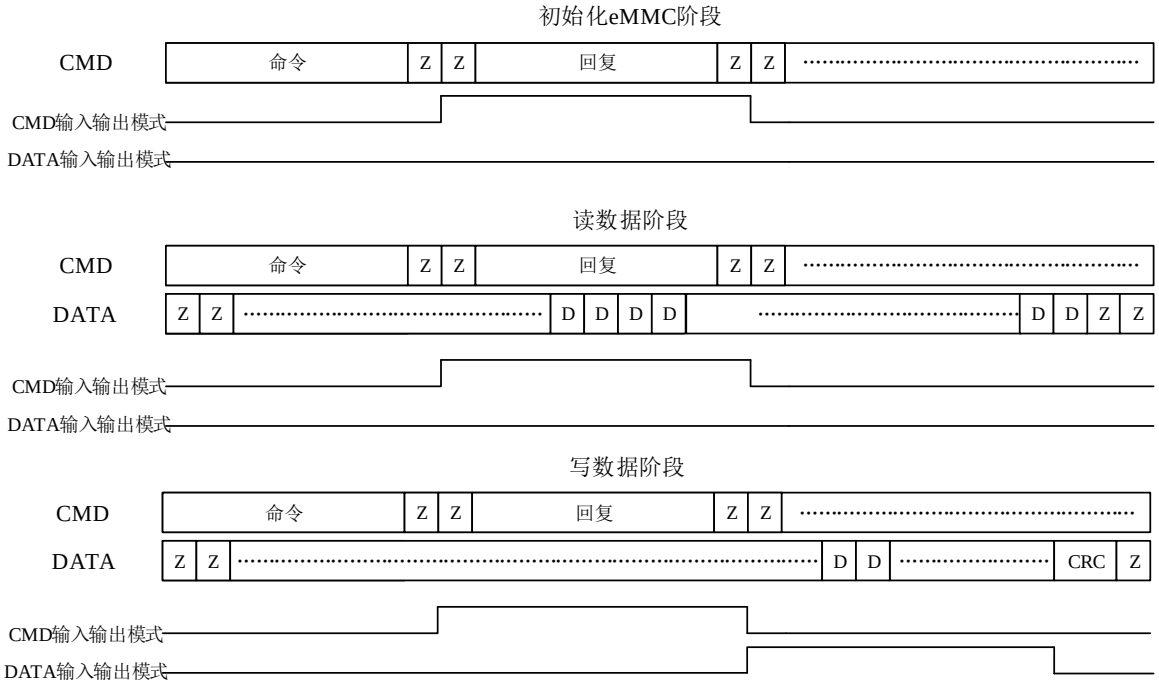

图 3.16 双向数据流向判决逻辑时序

3.5 基于 QSPI 接口的可靠性加载设计

为满足大容量数据存储需求，也可直接使用 NAND Flash 作为数据存储器[75]。资源紧张、

性能有限的反熔丝 FPGA 实现 NAND Flash 控制器代价较大，因此使用 ARM 处理器实现 NAND Flash 管理工作。结合 RT-Linux 两级 boot 方式， FPGA 保证 U-Boot 程序正确，利用加载了 U-Boot 程序的 ARM 处理器执行 NAND Flash 读写操作，间接的实现 FPGA 对 NAND Flash 翻转数据纠错。ARM 处理器支持从 QSPI 接口加载 U-Boot 程序，因此采用 QSPI 接口作为 U-Boot 程序加载接口。

3.5.1QSPI 接口介绍

QSPI 接口是 Quad SPI 的简称，由 Motorola 公司提出作为 SPI 接口的扩展接口。相比于标准 SPI 接口，QSPI 在接口上增加了 wp_n 与 rst_n 管脚，接口如图 3. 17 所示。与标准 SPI 支持全双工通信方式相比，QSPI 只支持半双工通信方式。QSPI 主要应用于数据存储器接口，数据通信的模式为主机发起操作，从机响应操作，读写分开进行。4 线制 QSPI 传输数据时，数据通过 mosi、miso、wp_n、rst_n 管脚同时传输，在相同通信时钟频率下 4 线制 QSPI 的数据传输速度是标准 SPI 的 4 倍。

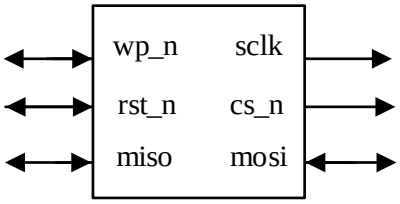

图 3. 17　QSPI 接口示意图

3.5.2 虚拟 SPI Flash 设计

存放备份关键数据的存储器介质选用 NOR Flash 型。Nor Flash 与 NAND Flash 不同，在使用过程中几乎不产生坏块，因此不需要复杂的硬件控制器与管理工作。NOR Flash 存储介质的物理特性决定了其读数据速度较快，数据擦除及数据写入存储阵列耗时较长。SPI NOR Flash 存储芯片数据手册中关于擦除与写入耗时信息如表 3. 2 所示

表 3. 2　NOR Flash 擦除、写入耗时信息表

操作类型	标准耗时	最长耗时
数据写入（256Byte）	120μs	1800μs
扇擦除（64KB）	0.15 s	1 s
分扇区擦除（32KB）	0.1 s	1 s
分扇区擦除（4KB）	0.05 s	0.4 s

写数据需要经历擦除、写入过程，写指令每次写入 256Byte 数据，擦除、写入耗时都按标准耗时可计算写出写入 53MB 关键数据总耗时为 153.25s。数据写入耗时长，因此采用虚拟 SPI Flash 设计避免 NOR Flash 写操作，提升操作系统加载效率。虚拟 SPI Flash 模块数据流如图 3. 18 所示，FPGA 读取 SPI NOR Flash 的数据并三模冗余获得无单粒子翻转数据存储于 FPGA 的 FIFO 中。ARM 处理器通过 QSPI 接口与 FPGA 连接后，将 FPGA 识别为 SPI Flash 存储器。FPGA 模拟 SPI Flash 的时序及功能，将缓存的数据发送给 ARM 处理器。

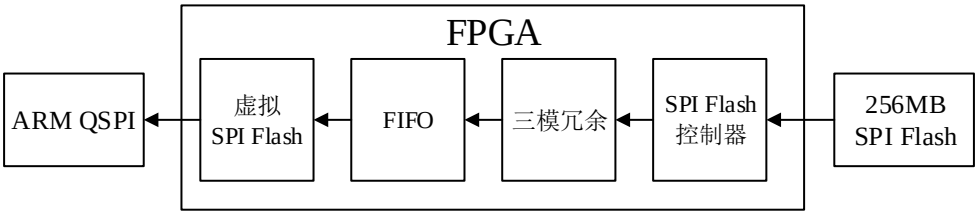

图 3. 18　虚拟 SPI Flash 模块数据流

SPI Flash 时序规定，标准 SPI 模式下读指令在地址信息发送后的下一个时钟周期，从机就需将读数据推上数据传输线。标准 SPI 模式读时序时序如图 3. 19 所示，T0 时刻地址信息传输完毕，T1 时刻前从机将数据推上 miso 信号线。数据传输事务完全由主机控制，从机无法控制传输事务的流程，因此虚拟 SPI Flash 模块必须及时准备需要传输的数据，避免出现主机已发出数据传输阶段的时钟信号而从机数据未准备完成的情况，导致数据传输错误。

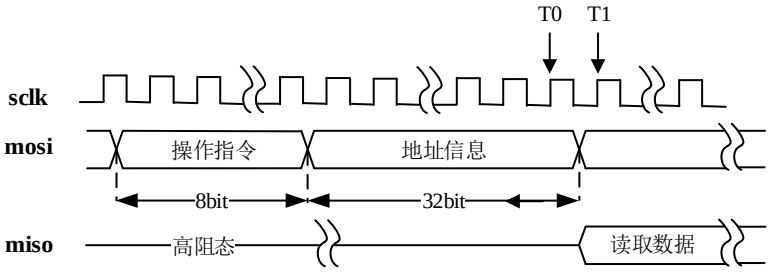

图 3. 19　SPI Flash 标准 SPI 模式读数据时序

AM5748 处理器数据手册显示，ARM QSPI 接口加载 U-Boot 程序时的时钟频率为 12MHz。在此时钟频率下，FPGA 采取解析读指令地址并按地址从 Flash 存储器中读取数据的策略，数据准备时间过长无法满足虚拟 SPI Flash 时序约束。因此 FPGA 采取预读取策略，从低地址到高地址，按顺序地从 Flash 存储器中预先读取数据，完成三模冗余后缓存入 FIFO 中。ARM 处理器发起读数据流程，FPGA 忽略 ARM 处理器发出的读数据地址，将 FIFO 中无单粒子翻转的数据推上数据传输线，完成传输事务。

反熔丝 FPGA 缓存资源有限，无法将所有关键数据都预读取入 FIFO 中，设计上采用数据流控制以保证 ARM 处理器读数据事务不出现数据流中断或 FIFO 溢出。数据流控制要求

FIFO 写入速度大于读出速度，且可启动、中断写入。当 FIFO 存储数据过少时，启动数据写入；当 FIFO 存储数据过多时，中断数据写入，以此形成不间断不溢出的数据流。为满足数据流控制条件，要求 ARM QSPI 接口的数据读取速度不能超过 FPGA 的 SPI Flash 控制器实际读数据速度，因此 ARM QSPI 接口采用标准 SPI 模式，时钟频率支持 12MHz、24MHz，FPGA 的 SPI Flash 控制器模块采用 4 线 QSPI 模式，时钟频率 20MHz。

3.5.3 基于 QSPI 接口的三模冗余设计

基于 QSPI 接口的可靠性加载设计硬件结构如图 3.20 所示，256MB SPI NOR Flash 采用 A、B 芯片硬件冗余冷备份的方式提升硬件可靠性，并通过 QSPI 接口与 FPGA 连接。两片 2GB NAND Flash 通过 GPMC 接口与 ARM 处理器连接，片选信号由 FPGA 控制，ARM 处理器 QSPI 接口与 FPGA 连接。系统正常工作时，ARM 处理器从 NAND Flash 中启动、运行操作系统及星务应用程序。当关键数据发生单粒子翻转，处理器出现故障时，FPGA 从 SPI NOR Flash 中获取正确数据并由 ARM 处理器写入 NAND Flash，完成翻转数据纠错。ARM 处理器从 NAND Flash 中重新加载操作系统，实现系统可靠性加载。

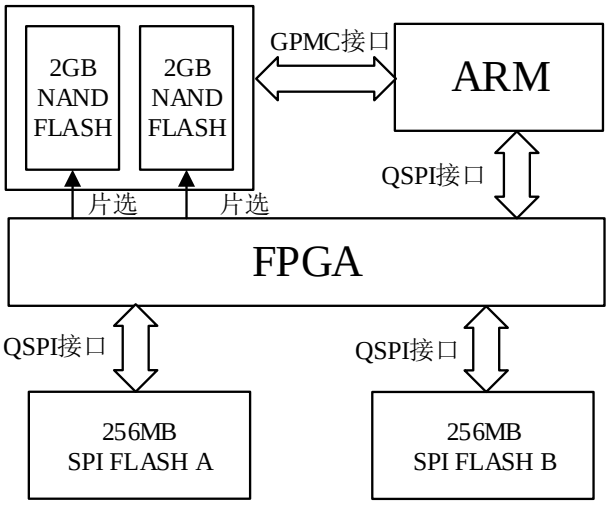

图 3.20 基于 QSPI 接口的可靠性加载设计硬件结构

U-Boot 程序分为 MLO 与 u-boot.img 两个部分，MLO 主要实现内存以及其他外部主要硬件的初始化，使 ARM 处理器可将 u-boot.img 加载进内存运行。u-boot.img 主要实现了对外部接口的初始化，如 Ethernet 的初始化，并可引导内核等操作系统文件的加载。U-Boot 加载阶段，ARM 处理器会从虚拟 SPI Flash 中先加载 MLO 文件后加载 u-boot.img 文件。

256MB SPI NOR Flash 的数据存储结构如图 3.21 与所示。三份备份关键数据按地址顺序存放，每份备份关键数据的内容相同，分区之间设置一定的数据隔离带。由于 FPGA 不解析

U-Boot 文件的内容，因此为获得 MLO 和 u-boot.img 的数据大小，使预读取策略的地址寄存器跳转正确，需要在 Flash 存储器的指定区域预存数据长度信息。U-Boot 文件编译完成后可获得 MLO、u-boot.img 的数据长度信息，写入三份备份关键数据的数据长度地址。ARM 处理器加载 U-Boot 程序时采用 512Byte 对齐，每次读数据量至少为 512Byte，因此数据长度信息共 4 个字节大小，前 2 个字节标志 MLO 数据大小的高 16 位，后 2 个字节标志 u-boot.img 数据大小的高 16 位。FPGA 按地址顺序依次读取三份备份关键数据，三模冗余后存入缓存中等待 ARM 处理器读取。

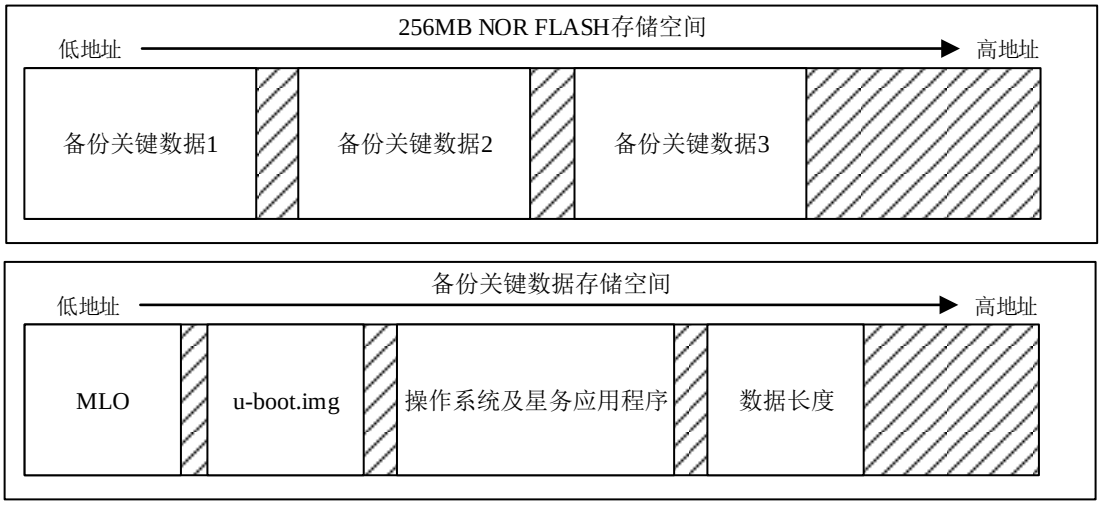

图 3. 21　256MB SPI NOR Flash 数据存储结构

2GB NAND Flash 中存放了 U-Boot、操作系统与星务应用程序外以及与星务应用程序相关约 1.5GB 的其他文件数据。2GB NAND Flash 数据存放结构如图 3. 22 所示。系统正常工作时，ARM 处理器从 NAND Flash 中启动操作系统、运行星务应用程序，完成星载任务。

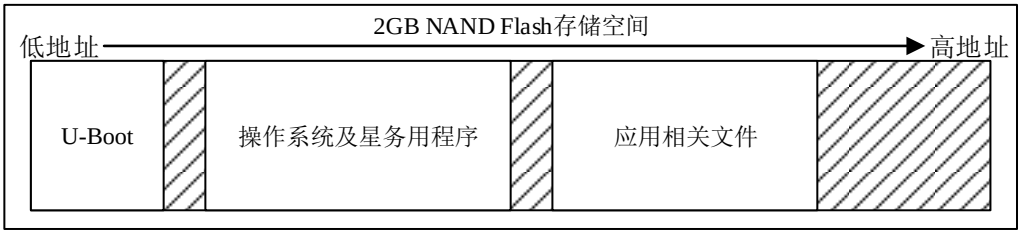

图 3. 22　2GB NAND Flash 数据结构

此可靠性加载设计采用两级 boot 启动方式，间接地实现 FPGA 对 NAND Flash 数据的纠错保护，基于 QSPI 接口的可靠性加载设计流程如图 3. 23 所示。当 2GB NAND Flash 存储器中的关键数据发生单粒子翻转，ARM 处理器无法正常工作时，FPGA 执行关键数据纠错流程。FPGA 配置 system boot 管脚电平设置 ARM 从 QSPI 接口启动加载 U-Boot 程序并执行 ARM 复位操作。ARM 处理器加载 U-Boot 文件完毕后执行 U-Boot 程序，QSPI 接口采用 4 字节地址寻址模式最大可读取 256MB 数据，时钟频率提升至 24Mhz。ARM 处理器从虚拟 SPI Flash

模块中读取 53MB 关键数据，并覆盖 NAND Flash 的关键数据区域完成翻转数据纠错。程序执行完毕后 ARM 处理器置高 GPIO，通知 FPGA U-Boot 程序运行结束，请求正常启动操作系统。FPGA 配置 system boot 管脚电平设置 ARM 从 NAND Flash 启动操作系统并复位 ARM 处理器，ARM 正常启动操作系统，实现系统可靠性加载。其中 U-Boot 具有启动接口识别功能，当识别到从 QSPI 加载时执行数据写入覆盖纠错功能，识别到从 NAND Flash 加载时执行正常启动一级 boot 流程。

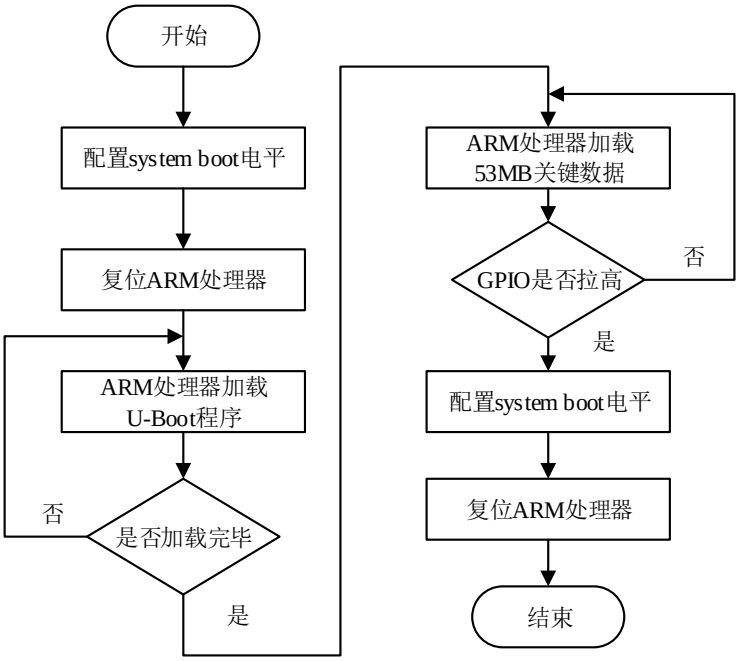

图 3. 23 基于 QSPI 接口的可靠性加载设计流程图

3.6 基于 SD 接口的可靠性加载设计

基于 QSPI 接口的可靠性加载设计采用的预读取策略不支持主机任意地址读数据且 ARM 处理器 QSPI 接口读数据速度慢，因此提出基于 SD 接口的可靠性加载设计解决以上问题，提升操作系统可靠性加载效率。

3.6.1 SD 接口介绍

安全数码存储卡 （Secure Digital Memory Card,SD）是一种基于 NAND Flash 的数据存储器，由松下电器、东芝和 SanDisk 于 1999 年 8 月联合推出，图 3. 24 为 SD 卡内部的功能结构。SD 卡内部分为 SD 控制器和 NAND Flash 阵列两部分，其中 SD 控制器负责解析主机发出的命令并进行响应, NAND Flash 阵列负责数据读取存储[76]。

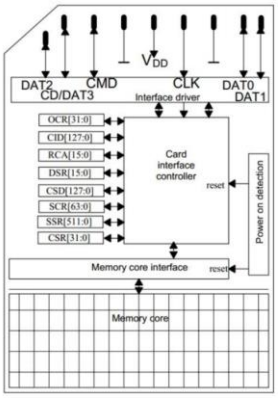

图 3.24 SD 卡功能结构

从硬件接口上看 SD 接口与 eMMC 接口十分相似，区别在于 SD 接口 DATA 有四根管脚而 eMMC 有八根管脚，SD 接口如图 3.25 所示。SD 卡是在 MMC 卡的基础上发展而来，SD 接口协议的内容也与 eMMC 相似，但 SD 主要侧重点在于存储数据的安全性。由于 SD 具有较大的存储容量、良好的读写速度与热插拔性，因此绝大部分处理器都支持 SD 接口。

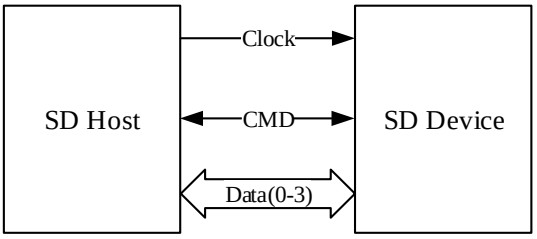

图 3.25 SD 接口示意图

3.6.2 虚拟 SD 卡设计

此可靠性加载设计采用两级 boot 方式。当综合电子系统正常工作时， FPGA 通过配置 system boot 管脚电平选择 ARM 处理器从 NAND Flash 启动操作系统。当 NAND Flash 中关键数据发生单粒子翻转，ARM 处理器无法正常工作时，FPGA 配置 system boot 管脚电平选择 ARM 处理器从 SD 接口加载 U-Boot 程序。加载了 U-Boot 程序的 ARM 处理器从虚拟 SD 卡模块读取已三模冗余的备份关键数据，覆盖 NAND Flash 中发生了单粒子翻转的关键数据，并重新从 NAND Flash 中启动操作系统，实现操作系统的可靠性加载。

整个设计的功能模块划分为虚拟 SD 卡模块、数据与触发信号跨时钟域模块、SPI Flash 控制器模块三个部分。虚拟 SD 卡模块的功能为根据 SD 时序响应 ARM 处理器主机命令，数据与触发信号跨时钟域模块功能为备份关键数据三模冗余及模块触发信号跨时钟域处理，SPI Flash 控制器模块功能为从 SPI Flash 的指定地址读取数据。

ARM 处理器通过 SD 接口与 FPGA 连接， FPGA 通过模拟 SD 卡的时序及功能将自身虚

拟为一张 7.0GB 大小的 SD-ROM 卡，并被 ARM 处理器识别，反熔丝 FPGA 的虚拟 SD 卡模块接口如图 3. 26 所示。模块左侧为反熔丝 FPGA 与 ARM 处理器 SD 接口连接的信号。模块右侧为虚拟 SD 模块与数据与触发信号跨时钟域模块、SPI Flash 控制器模块的连接信号，通过 rdvalid 和 rdready 信号实现数据传输的“握手”流程。

图 3. 26 虚拟 SD 卡模块接口

虚拟 SD 卡模块主要实现了初始化识别、读命令触发、地址映射、数据 CRC 校验码功能：

（1）初始化识别：SD 协议规定，主机在从 SD 卡读写数据之前需对 SD 卡进行初始化识别，内容包括软复位、工作电压范围匹配、SD 卡信息查询、数据传输模式设置等。虚拟 SD 卡模块在初始化识别阶段，根据 ARM 处理器的信息查询命令返回预置的 SD 寄存器参数值。ARM 将 FPGA 识别为 7.0GB 大小、SDHC 类型、最大可支持 25MHz 时钟频率的 SD-ROM 卡。此状态下 SD 卡读数据命令有效，写数据命令无效，以避免内核加载过程中，主机发出的写数据命令不被响应，导致虚拟 SD 卡挂载失败。

（2）读命令触发：虚拟 SD 卡模块在接收到 ARM 处理器 SD 接口发送的读命令后，产生一个持续置高的握手开始信号 rdvalid，触发数据与触发信号跨时钟域模块、SPI Flash 控制器模块执行相应操作。

（3）地址映射：SD 协议中规定 SDHC（4GB-32GB）类型的卡读命令采用块寻址方式，每个地址对应 512Byte 数据。虚拟 SD 卡模块接收到读命令后解析出主机读数据的逻辑地址，并映射成 SPI Flash 的实际物理地址，作为 SPI Flash 控制器模块的地址输入信号。

（4）数据 CRC 校验码：SD 协议中规定，命令与回复在参数后需附加 7 位 CRC7 校验码用于接收端校验。读数据时在数据后需附加 16 位的 CRC16 校验码用于数据传输校验。虚拟 SD 卡模块向 ARM 处理器返回回复、发送数据时，内部的 CRC7、CRC16 校验码模块根据数据值生成对应的校验码。

虚拟 SD 卡模块以“握手”方式实现信号触发与数据传递。当虚拟 SD 模块在接收到读数据命令后，解析出命令中的地址信息并输出到 SPI Flash 控制器模块，同时置高 rdvalid 信号，“握手”流程开始。虚拟 SD 卡模块在检测到 rdready 信息置高之前，rdvalid 信号会持续置高。当虚拟 SD 卡模块检测到 rdready 信号置高后表明读数据准备完毕，rdvalid 信号同时置低，

“握手”流程结束，虚拟 SD 卡模块将已三模冗余的数据推上传输线。

整个过程中各模块之间的执行串行度高，总耗时由 SD 卡读数据命令接收、SPI Flash 数据读取、数据三模冗余与控制信号跨时钟域、读数据返回四个部分组成。采用面积换取速度的并行设计思路提升加载效率，三份备份关键数据存放于三片 SPI Flash 存储器中，读数据时同时从三片 SPI Flash 存储器中读取备份关键数据，SPI Flash 实际读数据速度可提升三倍。

SD 协议允许主机等待从机准备数据，数据传输事务由主机、从机共同控制。因此可采用解析主机发送的读数据命令地址，从 SPI Flash 对应物理地址读取数据的策略，实现 ARM 处理器任意地址读取数据。SD 读数据命令时序如图 3. 27 所示，Z 表示高阻态，S、D、E 分别表示数据起始位、数据位与停止位，Nac 为 SD 主机发送读数据命令结束时到从机将读数据推上传输线的时间间隔，最长不能超过 100ms。SD 读数据命令传输数据时以 512Byte（1 Block）为单位，每 block 数据准备用时最长为 83.75μs。FPGA 收到 ARM 处理器发送的读数据命令到将数据推上数据总线的时间间隔最长为 83.84μs，满足 SD 协议关于读数据返回延时约束。

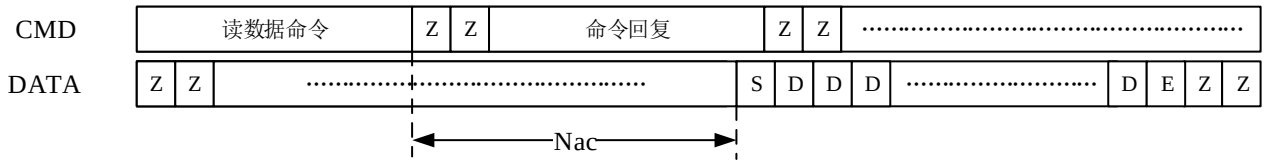

图 3. 27　SD 读数据命令时序

3.6.3 基于 SD 接口的三模冗余设计

基于 SD 接口的可靠性加载设计硬件结构如图 3. 28 所示。硬件设计上冗余冷备份一片 SPI Flash 存储器以提高设计的硬件可靠性。四片 256MB SPI Flash 存储器通过 QSPI 接口与 FPGA 相连。两片 2GB NAND Flash 通过 GPMC 接口与 ARM 处理器相连，片选信号由 FPGA 控制，ARM 处理器通过 SD 接口与 FPGA 相连。

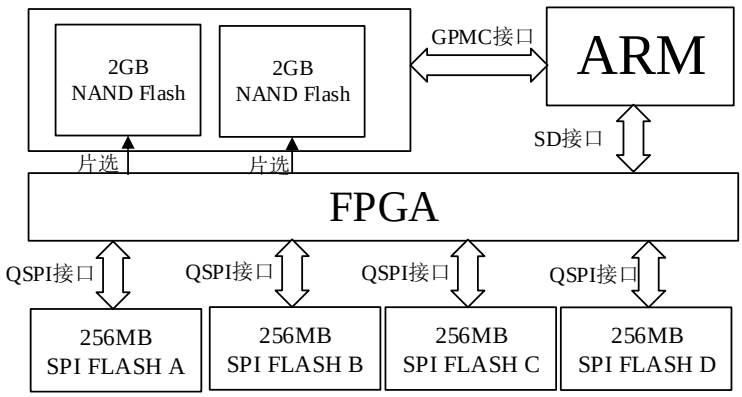

图 3. 28　基于 SD 接口可靠性加载设计硬件结构

256MB SPI Flash 与 2GB NAND Flash 的数据存储结构如图 3. 29 所示，四片 256MB SPI Flash 中存放相同的备份关键数据。FPGA 可同时读取其中三片 SPI Flash 的数据，以提升 Flash 控制器模块实际读数据速度。NAND Flash 中存放 U-Boot 程序、RT-Linux 操作系统及星务应用程序、应用相关文件。

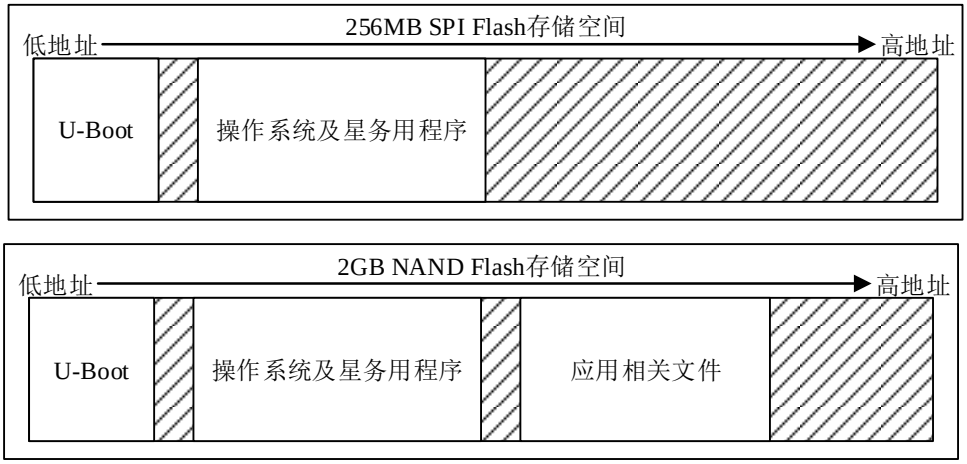

图 3. 29　256MB SPI Flash 与 2GB NAND Flash 的数据存储结构

数据与触发信号跨时钟域模块、Flash 控制器模块结构如图 3. 30 所示。虚拟 SD 卡模块与 Flash 控制器模块处于不同的时钟域，因此数据、触发信号在这两个模块之间传递时需要由数据与触发信号跨时钟域模块进行跨时钟处理。触发信号 rdvalid 与 rdready 信号采用打拍法实现跨时钟域。为尽可能减小触发信号亚稳态对模块的影响，打拍次数增加为 4 次。数据与触发信号跨时钟域模块同时实现三模冗余。Flash 控制器模块与虚拟 SD 卡模块的时钟域不同，为实现多 bit 数据安全跨时钟域，FIFO_TMR 采用异步 FIFO 形式。

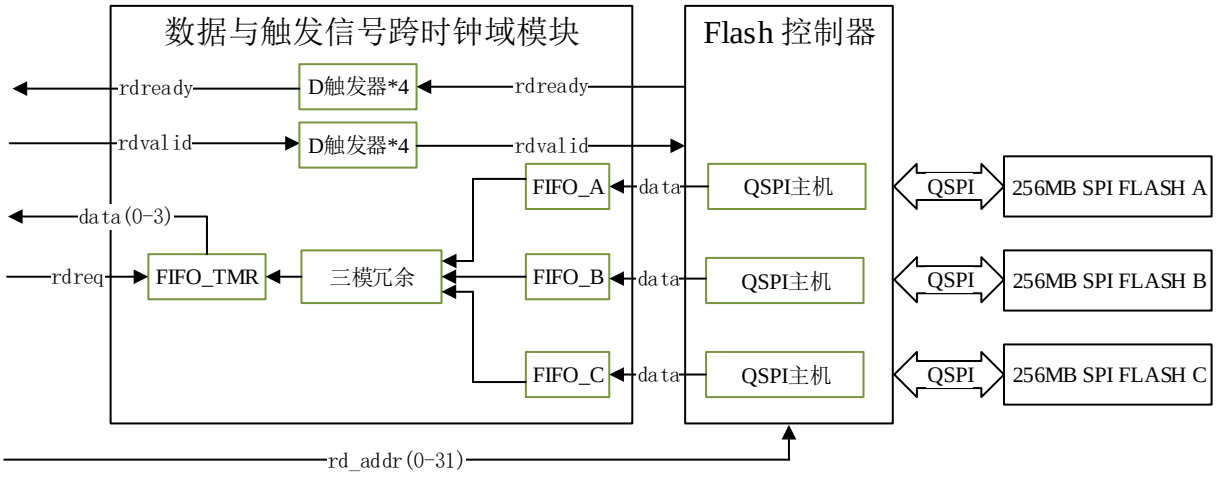

图 3. 30　数据与触发信号跨时钟域模块与 Flash 控制器模块结构

基于 SD 接口的可靠性加载设计流程如图 3. 31 所示。当 NAND Flash 中的关键数据发生单粒子翻转，ARM 处理器无法正常工作，FPGA 配置 system boot 管脚电平选择从 SD 接口启

动系统并复位 ARM 处理器。加载了 U-boot 程序的 ARM 处理器从虚拟 SD 卡中读取已三模冗余的备份关键数据并写入 NAND Flash 中，覆盖低地址发生了单粒子翻转的关键数据。U-Boot 程序执行完毕后，通过拉高 GPIO 通知 FPGA 程序运行结束，请求正常启动操作系统。FPGA 配置 system boot 管脚电平选择 ARM 处理器从 NAND Flash 启动操作系统并复位 ARM 处理器，ARM 处理器正常启动，实现系统可靠性加载。其中 U-Boot 具有启动接口识别功能，当识别到从 SD 卡加载时执行数据写入覆盖纠错功能，识别到从 NAND Flash 加载时执行正常启动一级 boot 流程。

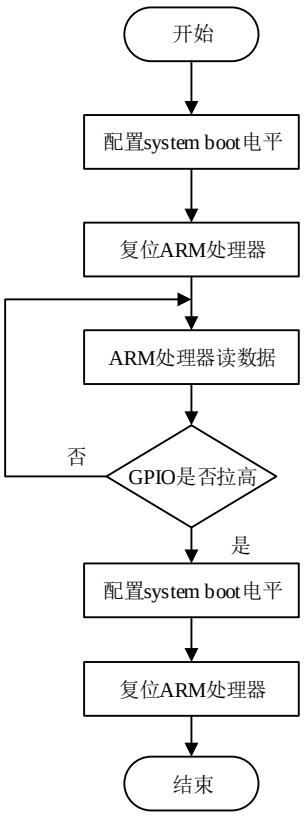

图 3.31 基于 SD 接口的可靠性加载设计流程图

3.7 本章小结

本章详细地介绍了基于操作系统的星载可靠性加载设计。首先介绍了可靠性分析的三种模型，然后在此基础上介绍了三模冗余容错技术及 RT-Linux 操作系统的启动方式与特点。最后根据功能需求提出了三种星载可靠性加载设计，并介绍各容错设计的功能及实现方法。

4.双系统冗余备份管理方案研究

受空间资源、低功耗需求的限制及系统可更换升级的功能需求，ZHX 卫星采用双机异构备份冗余架构，实现系统级冗余备份，提升系统整体可靠性。系统级冗余备份引发双系统搭载的多处理器竞争问题，因此需引入系统备份管理方式解决此类问题。

4.1 综合电子系统主从状态管理

双系统共搭载 2 片反熔丝 FPGA 与 2 片 ARM 处理器，4 个具有数据处理管理能力的器件，都具备冗余备份管理能力。无约束的冗余备份管理权会使得多个处理单元处于竞争状态，引起系统运行错误与功能紊乱。这种故障状态在 CAN 总线数据传输上表现尤为明显。

在 ZHX 综合电子系统当前设计中，若不对冗余备份管理权进行约束，综合电子系统主备系统在 CAN 总线仲裁阶段都会认为自己获得总线控制权。此时 CAN 总线上会出现两个通信主机，这将导致 CAN 总线数据传输错误。

表 4.1 为单、双通信主机状态下 CAN 总线上截取的数据。可以看出双通信主机状态下的 CAN 数据内容与单通信主机 CAN 数据内容完全不同。序号 3 的单通信主机 CAN 数据表示通信测量单元 A 机返回的工程遥测帧，而双通信主机 CAN 数据内容无意义，综合电子系统的工程遥测查询功能丧失。

表 4.1 单、双主机状态下 CAN 总线的数据

序号	双通信主机 CAN 数据(HEX)	单通信主机 CAN 数据(HEX)
1	00 00 00 29 00 00 0b 00	00 00 00 00 00 1b 25 55
2	21 23 00 00 00 01 32 00	55 55 cb 6a aa aa aa 32
3	00 8e ab 76 00 00 00 00	55 04 03 89 21 aa aa 5a
4	0f e0 aa aa aa aa aa 9e	01 02 0f 05 eb 90 00 f7
5	21 23 00 00 00 01 32 00	55 55 cb 6a aa aa aa 32

综合电子系统采用主从状态形式约束多处理器的冗余备份管理权以解决管理权冲突问题。主状态的综合电子系统处于工作状态，从状态的综合电子系统处于备份状态。为简化多处理器主从状态管理及控制逻辑，综合电子系统冗余备份采用 FPGA 不区分主从状态、ARM 处理器区分主从状态与 FPGA 区分主从状态、ARM 处理器不区分主从状态两种管理模式。

(1) FPGA 不区分主从状态、ARM 处理器区分主从状态

当综合电子系统进入主状态时，ARM 处理器也为主状态；当综合电子系统系统进入从状态时，ARM 处理器也为从状态。从状态 ARM 处理器将不再竞争 CAN 总线，不再发起数据传输事务，从状态综合电子系统作为一个从设备节点挂载进 CAN 总线。主状态综合电子系统可通过 CAN 总线获取从状态综合电子系统的工程遥测信息。

(2) FPGA 区分主从状态、ARM 处理器不区分主从状态

当综合电子系统进入主状态时，FPGA 也为主状态；综合电子系统进入从状态时，FPGA 也为从状态。ARM 处理器无主从状态，因此 ARM 处理器上电处于工作状态时将持续竞争 CAN 总线。为保证 CAN 总线不出现双通信主机，从状态 FPGA 请求 ARM 处理器软关机或执行 ARM 处理器断电操作，保证 CAN 总线有且仅有一个通信主机。主状态综合电子系统可以通过 CAN 总线获取从状态综合电子系统的工程遥测信息。

4.2 故障信息检测

双系统冗余备份结构需要一个高可靠性、高准确度的检测切换单元以实现故障精准判断，管理权平稳交接，因此使用高可靠性的反熔丝 FPGA 完成故障信息检测与执行主从切换流程，提升切换单元的可靠性[77]。检测切换单元通过检测、分析故障信息判断系统内是否发生故障，故障信息主要包括心跳信号、ARM 处理器及状态健康信息检测[78-79]。

(1) 心跳信号

星务计算单元 ARM 处理器通过 UART 串口按 2s 一次的节拍与 FPGA 通信，发送数据、状态及控制指令信息。FPGA 将 ARM 处理器 2s 一次的数据发送过程视为 ARM 处理器的心跳信号。若 FPGA 连续 4s 都未收到 ARM 处理器发送的心跳信号，则认为 ARM 处理器发生软件故障。

(2) ARM 处理器状态

ARM 处理器状态分为设备状态与主从状态两类。通过心跳信号与 ARM 处理器的电流值判断 ARM 处理器的设备状态。ARM 处理器设备状态判决逻辑如表 4.2 所示。

表 4.2 ARM 处理器设备状态判决逻辑

	ARM 处理器电流正常	ARM 处理器无电流	ARM 处理器高电流
心跳信号存在	ARM 处理器工作正常	/	ARM 处理器硬件故障
心跳信号消失	ARM 处理器软件故障	ARM 处理器断电	ARM 处理器硬件故障

当健康状态信息中 ARM 处理器的电流过高，表明 ARM 处理器出现严重的硬件故障，FPGA 立即关闭 ARM 处理器的电源开关。当心跳信号消失而 ARM 处理器电流正常，则表明

ARM 处理器出现软件故障，无法正常运行星务应用软件、执行卫星任务。当心跳消失而 ARM 处理器无电流，则表明 ARM 处理器处于断电关机状态。

ARM 处理器使用 UART 串口按卫星数据传帧格式将处理器主从状态信息发送至 FPGA。当 ARM 处理器发送的主从状态信息与综合电子系统主从状态信息不匹配，表明 ARM 处理器出现故障。

(3) 健康信息检测

综合电子系统在工作运行过程中持续收集系统内的健康状态信息用于判断是否有故障发生。健康信息检测主要用于判断硬件是否工作正常，反熔丝 FPGA 收集健康状态信息，并发送至检测切换单元，健康状态信息如表 4.3 所示。

表 4.3 健康状态信息

设备	健康状态
启动 eMMC	芯片 A：000 正常、001 损坏、011 未使用 芯片 B：100 正常、101 损坏、111 未使用
备份 eMMC	芯片 A：000 正常、001 损坏、011 未使用 芯片 B：100 正常、101 损坏、111 未使用
RTC	0 正常、1 损坏
SPI Flash	芯片 A：0000 正常、0001 损坏、0011 未使用 芯片 B：0100 正常、0101 损坏、0111 未使用 芯片 C：1100 正常、1101 损坏、1111 未使用 芯片 D：1000 正常、1001 损坏、1011 未使用
温传	0 温度正常、1 温度过高
F-RAM	0 正常、1 损坏。
电流检测芯片	00 ARM 处理器电流正常 01 ARM 处理器无电流 11 ARM 处理器高电流

4.3 系统冗余备份管理权切换设计

综合电子系统通过主从状态管理解决系统级冗余备份下多处理器竞争管理权的问题。为完成系统故障时管理权平稳切换，实现双系统冗余备份功能，切换单元按照实时性、可靠性

与自主性不同，设计了父级仲裁切换和同级自主切换两种系统备份管理方式。

4.3.1 父级仲裁切换

父级仲裁切换的检测切换单元由指令执行单元实现。由于指令执行单元属于卫星分系统中可靠性最高的系统，且具备综合电子系统电源开关控制能力，在控制管理逻辑及可靠性上的排序都高于综合电子系统，因此将由指令执行单元实现的管理权切换称为父级仲裁切换。

父级仲裁切换采用 FPGA 不区分主从状态、ARM 处理器区分主从状态的管理方式。父级仲裁切换主要解决 ARM 处理器主从状态软件故障导致的系统冗余备份管理权混乱问题。当出现 ARM 处理器的主从状态信息与综合电子系统主从状态信息不符时，父级仲裁切换执行操作。

综合电子系统通过反熔丝 FPGA 的 SPI 接口向指令执行单元发送健康监测信息。指令执行单元进行故障判断后，通过 SPI 接口发送指令控制主从切换流程，父级仲裁切换结构结如图 4. 1 所示。

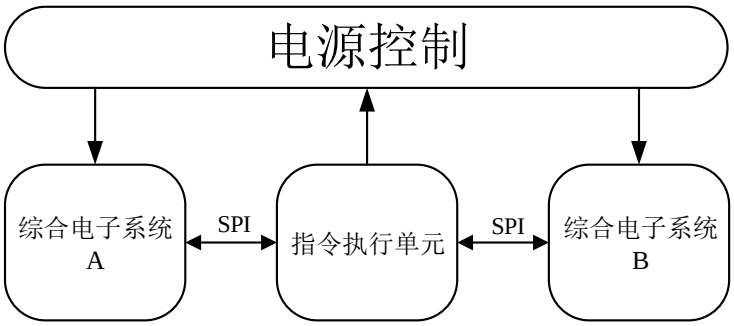

图 4. 1　父级仲裁切换结构

主从状态切换流程由综合电子系统 FPGA 执行。指令执行单元发送的控制指令内容分为系统主从状态与 ARM 处理器电源操作两类。

(1) 指令执行单元通过指令信息中的主从标志位告知 FPGA 当前系统的主从状态。当主从标志位为 0 时，综合电子系统为从状态，主从标志位为 1 时，综合电子系统为主状态。FPGA 解析出主从标志位信息后，会通过 UART 接口向 ARM 处理器发送控制指令，请求 ARM 处理器进入对应的主从状态，并通过检查 ARM 处理器主从状态信息确认是否进入正确状态。

(2) ARM 处理器电源操作是指指令执行单元通过指令信息中的电源标志位控制 ARM 处理器电源。当电源标志位为 0 时，FPGA 对 ARM 处理器执行断电操作，当电源标志位为 1 时，FPGA 对 ARM 处理器执行上电操作。

为简化父级仲裁切换的判决、切换逻辑，父级仲裁切换只处理管理权混乱引发的故障状态。为方面描述，将 A 板卡、B 板卡上的综合电子系统用 A 系统、B 系统表述。A、B 系统主从状态为一主一从或一主一关机时，双系统冗余备份管理权正常。故障状态分 A、B 系统双主，A、B 系统双从、A、B 系统双关机、A、B 系统一从一关机四类。

当故障状态为 A、B 系统双主时，A 系统、B 系统都处于主状态，双系统冗余备份管理权发生冲突，CAN 总线处于数据堵塞状态。父级仲裁切换的流程如图 4.2 所示，其中通信测量单元下传的简单遥测包含卫星主要分系统的部分工程遥测信息，如星敏传感器信息、电源信息、姿轨控状态等。当 CAN 总线传输的正常遥测无效时，通过下传简单遥测，地面仍能获取卫星的部分状态，避免卫星处于失联状态。检测到故障后，指令执行单元执行通知通信测量单元下传简单遥测，设置 A 系统为主状态 B 系统关闭 ARM 电源，管理权冲突状态解除时通知通信测量单元下传正常遥测。

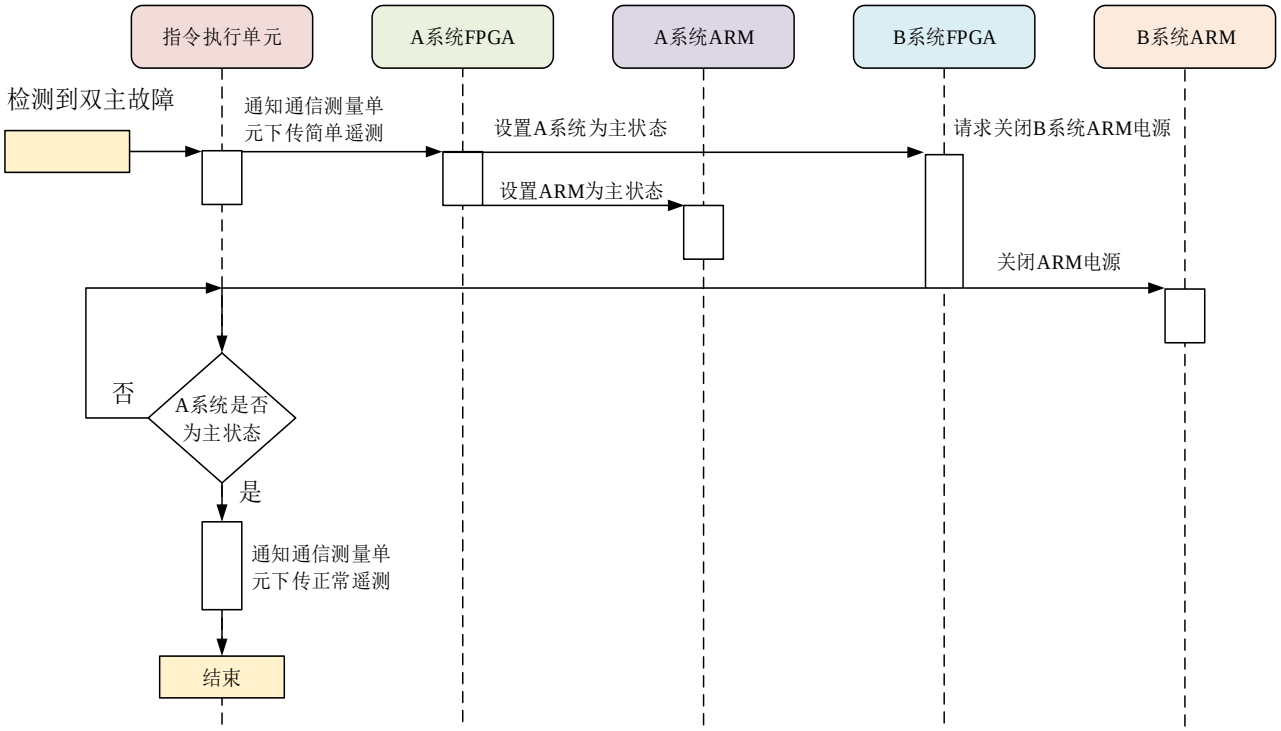

图 4.2 双主故障切换流程

当故障状态为 A、B 双从时，A 系统、B 系统都处于从状态，所有处理单元都丢失双系统冗余备份管理权，CAN 总线处于无数据传输状态。父级仲裁切换的流程如图 4.3 所示，检测到故障后，指令执行单元通知通信测量单元下传简单遥测，并优先设置 A 系统进入主状态，B 系统保持从状态。若 A 系统在 4s 内成功进入主状态，则 A 系统获得管理权，切换流程结束通知通信测量单元下传正常遥测，否则设置 B 系统进入主状态。若 B 系统在 4s 内成功进入主状态，则 B 系统获得管理权，切换流程结束通知通信测量单元下传正常遥测。否则表明

双系统都无法正常进入主状态，获取管理权失败，父级仲裁切换功能失效，等待地面研判上注组合指令使综合电子系统摆脱故障状态。

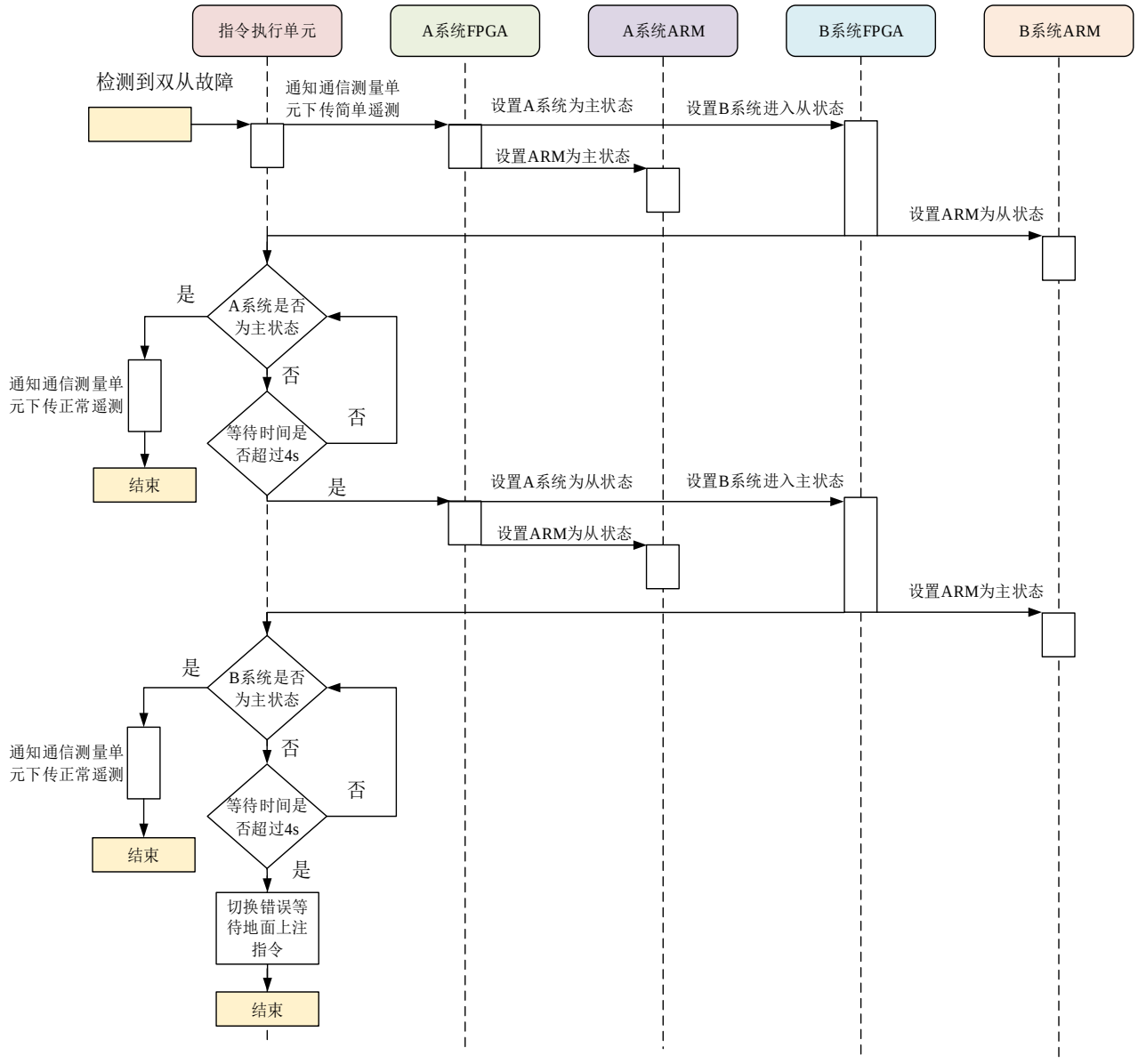

图 4.3 双从故障切换流程

当故障状态为 A、B 系统双关机时，A 系统、B 系统都处于关机状态，所有处理单元都无法获取管理权，CAN 总线上无数据传输。其中 ARM 处理器无心跳信号及 ARM 处理器处于断电关机都属于关机状态。父级仲裁切换的流程如图 4. 4 所示。检测到故障时，指令执行单元通知通信测量单元下传简单遥测并优先请求打开 A 系统 ARM 电源，设置 A 系统为主状态。若 A 系统在 25s 内进入主状态，则 A 系统获得管理权，切换流程结束通知通信测量单元下传正常遥测，否则请求打开 B 系统 ARM 电源并设置为主状态。若 B 系统在 25s 内进入主状态，则 B 系统获得管理权，切换流程结束通知通信测量单元下传正常遥测。否则表明双系统都无

法正常进入主状态，获取管理权失败，父级仲裁切换功能失效，等待地面研判上注组合指令使综合电子系统摆脱故障状态。

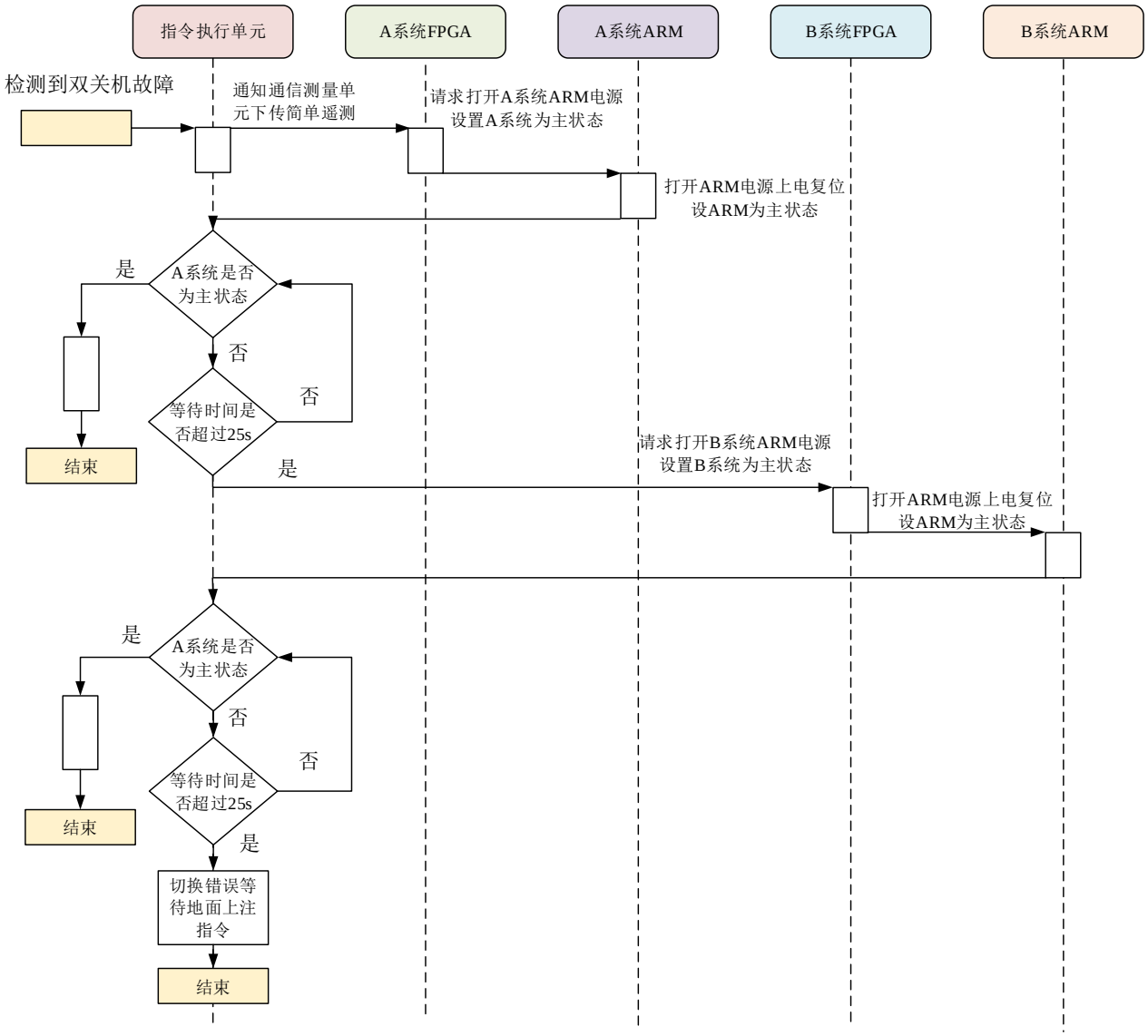

图 4.4 双关机故障切换流程

当故障状态为 A、B 系统一从一关机时，A、B 系统中一系统为从状态，一系统为关机状态，所有处理单元都丢失双系统冗余备份管理权，CAN 总线上无数据传输。父级仲裁切换流程如图 4.5 所示，以 A 系统为从状态、B 系统为关机状态为例，B 系统为从状态、A 系统为关机状态切换流程相同。检测到故障时，指令执行单元知通信测量单元下传简单遥测并设置 A 系统进入主状态。若 A 系统在 4s 内进入主状态，则 A 系统获得管理权，切换流程结束通知通信测量单元下传正常遥测，否则请求打开 B 系统 ARM 电源并设置 B 系统进入主状态。若 B 系统在 25s 内进入主状态，则 B 系统获得管理权，切换流程结束通知通信测量单元下传正常遥测。否则表明双系统都无法正常进入主状态，获取管理权失败，父级仲裁切换功能失

效，等待地面研判上注组合指令使综合电子系统摆脱故障状态。

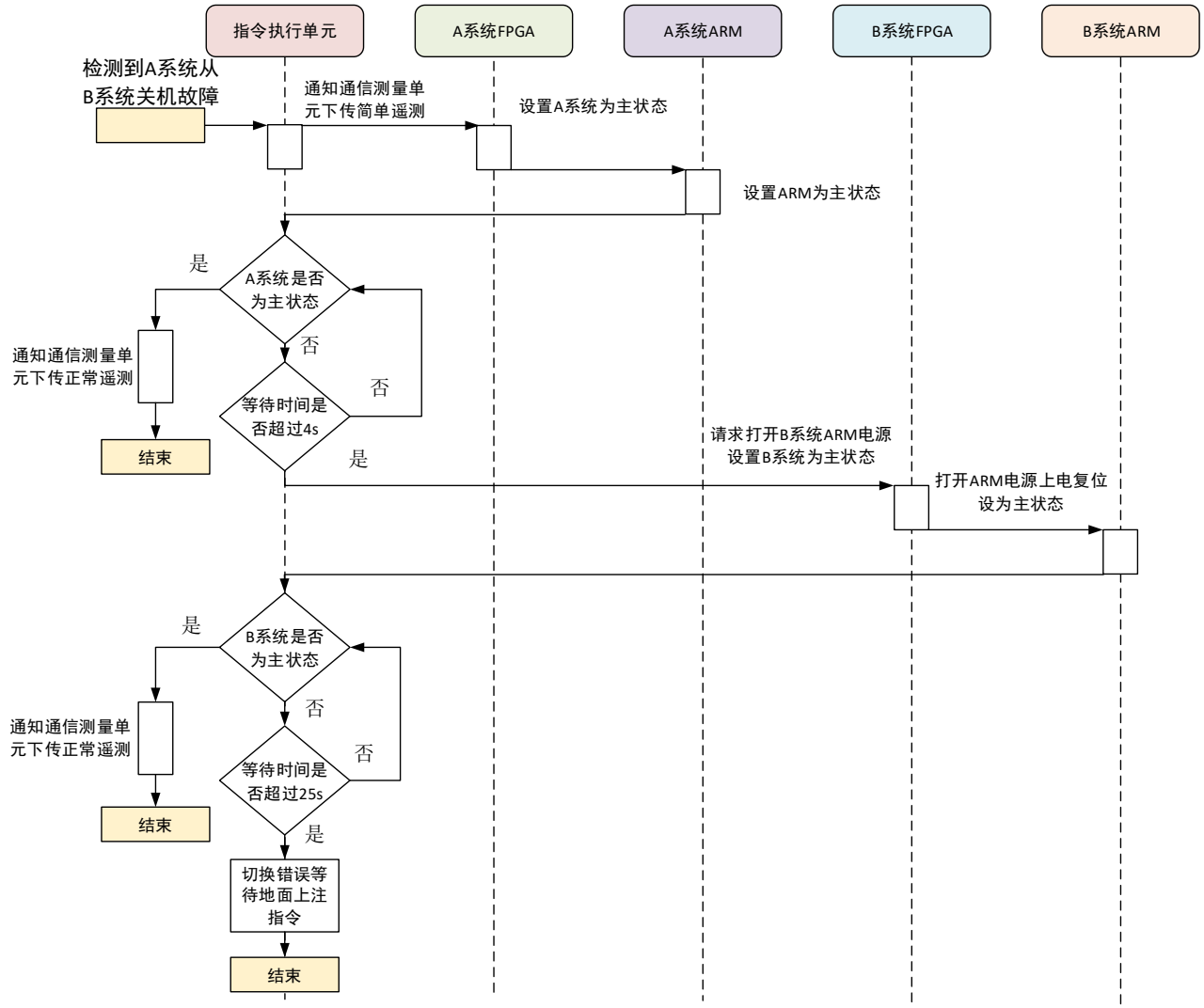

图 4.5 一从一关机故障切换流程

4.3.2 同级自主切换

同级自主切换是指检测切换单元由综合电子系统自身实现，故障发生后的主从切换流程由综合电子系统控制执行。同级自主切换采用 FPGA 区分主从状态，ARM 处理器不区分主从状态的管理模式。当 FPGA 为主状态且 ARM 处理器故障时 FPGA 将接替 ARM 处理器成为 CAN 总线通信主机继续传输正常遥测。FPGA 为从状态时作为通信从机挂载进 CAN 总线并响应主机的工程遥测查询指令。ARM 处理器不区分主从状态，正常上电工作时便会竞争 CAN 总线通信主机，因此为避免主从系统管理权冲突引发 CAN 总线数据传输故障，规定从系统的 ARM 处理器必须处于关机状态。

同级自主切换通过 CAN 总线实现主从切换流程信息的传输并采用指令帧形式控制切换

流程，系统切换结构如图 4.6 所示。主系统作为通信主机通过 CAN 总线发送工程遥测查询帧获取从系统的工程遥测信息，主系统接收到从系统返回的工程遥测帧即认为从机系统 FPGA 功能正常。

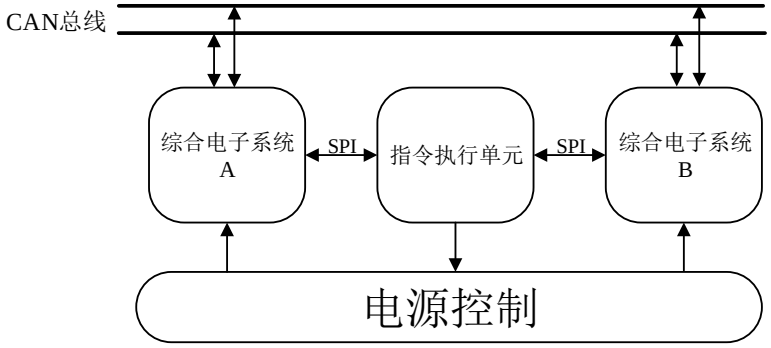

图 4.6 同级自主切换结构

同级自主切换采用“两次握手”方式，实现切换流程中主从系统状态信息的同步。主从切换开始时，主系统发起主从切换帧，请求从系统切换成为主系统，并等待从系统应答。主系统接收到来自从系统的主从切换回复帧后进入从状态，关闭 ARM 处理器电源，并等待从系统 ARM 处理器发送工程遥测请求帧，实现第一次“握手”。从系统的 FPGA 接收到主从切换请求帧后，打开 ARM 电源，并返回一次主从切换回复帧。从系统 ARM 处理器正常启动后，向主系统发送工程遥测请求帧，实现第二次“握手”。此时主系统稳定进入从状态，从系统稳定进入主状态，主从切换流程结束，管理权平稳切换。

主系统向从系统发送工程遥测查询帧以获取从系统的健康信息，并检测判断主、从系统是否有故障发生。同级自主切换支持较为复杂的检测切换逻辑，根据故障器件功能的不同，将故障检测类别分为星载任务支持与星务应用支持两类，通过地面上注的准禁指令选择执行的故障检测类别。准禁指令为 1 时表示故障检测类别为星载任务支持，将检测主系统硬件是否支持星载任务执行，若不支持则执行主从切换，由从系统接替主系统工作；准禁指令为 0 时表示故障检测类别为星务应用支持，同级自主切换只需保证综合电子系统中的 ARM 处理器工作正常，系统获取管理权，CAN 总线存在通信主机。

故障检测类别为星载任务支持并发生故障时，主从切换流程如图 4.7 所示。ARM 处理器作为 CAN 总线通信主机发起主从切换帧，切换流程由 ARM 处理器和 FPGA 共同控制执行。主系统检测到系统中存在故障的硬件时，启动主从切换流程。主系统 ARM 处理器通知指令执行单元打开从系统电源，并向从系统发送工程遥测查询。若从系统返回工程遥测，则认为从系统 FPGA 上电且工作正常，主系统 ARM 处理器执行断电操作，主系统 FPGA 接管 CAN 总线，并向从系统发送主从切换帧。主系统 FPGA 接收到来自从系统的主从切换指令回复帧

后进入从状态。从系统 FPGA 接收到来自主系统发送的主从切换帧后返回主从切换回复帧，并打开 ARM 电源，从系统 ARM 处理器接管 CAN 总线。从系统 ARM 处理器正常启动后向主系统发送工程遥测查询帧，表示从系统已进入主状态，此时主系统稳定进入从状态，从系统稳定进入主状态，系统冗余备份管理权平稳交接。若从系统 ARM 故障，无法发送工程遥测帧，主系统 FPGA 等待一段时间后重新进入主状态并开启 ARM 处理器电源。

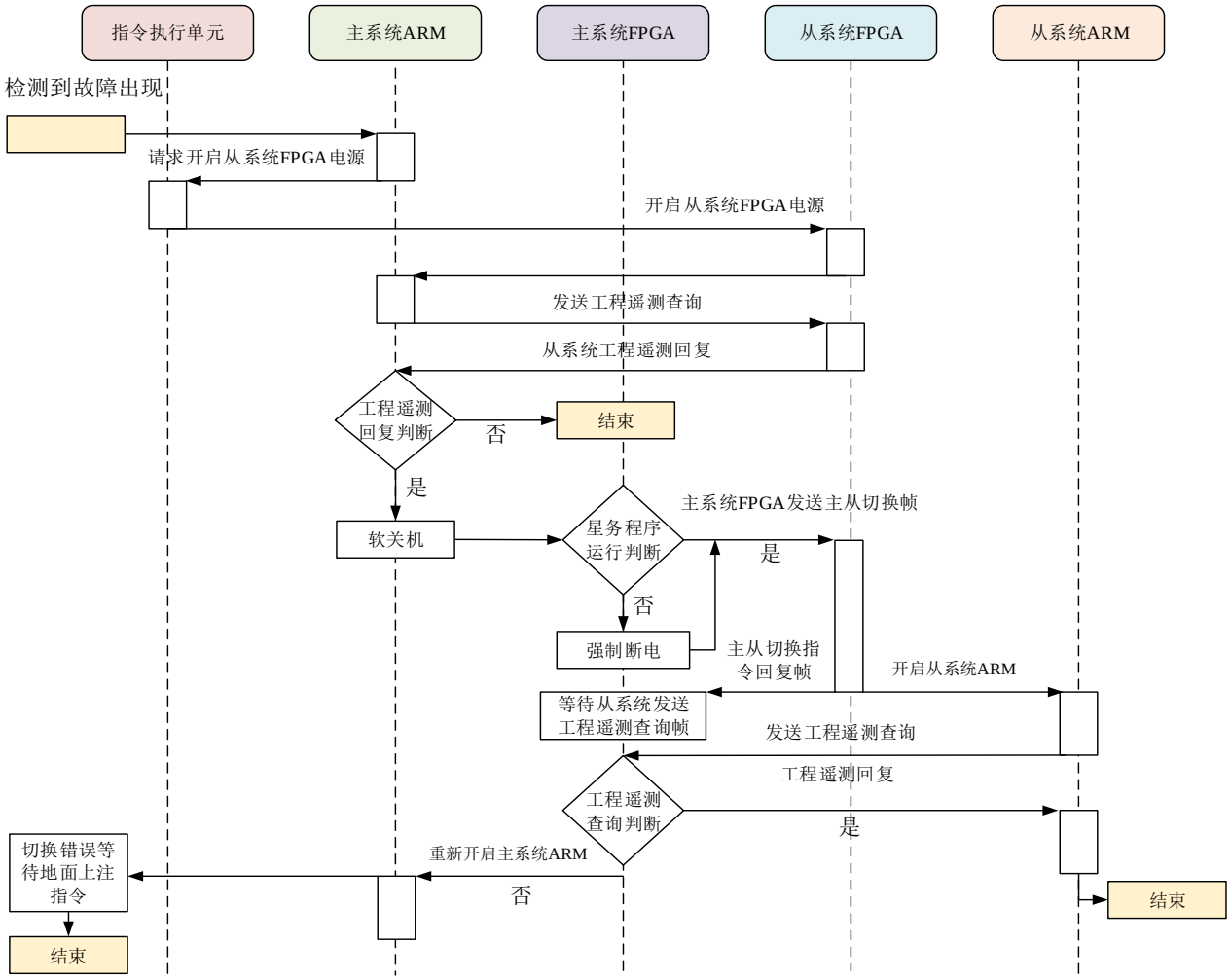

图 4.7 ARM 处理器和 FPGA 主从切换流程

当故障检测类别为星务应用支持且发生故障时，故障表现为主系统 ARM 处理器星务功能丧失，管理权丢失。主系统 ARM 处理器无法作为 CAN 总线通信主机，反熔丝 FPGA 接替 ARM 处理器成为 CAN 总线的通信主机，整个切换流程由 FPGA 控制，主从切换流程如图 4.8 所示。主系统 FPGA 检测到系统中存在故障时，启动主从切换流程。主系统 FPGA 接管 CAN 总线，关闭 ARM 处理器电源，通知指令执行单元打开从系统电源，并向从系统发送工程遥测查询。若从系统返回工程遥测，则认为从系统 FPGA 上电且工作正常，主系统 FPGA 向从系统发送主从切换帧。主系统 FPGA 接收到来自从系统的主从切换指令回复帧后，进入从状态。

从系统 FPGA 接收到来自主系统发送的主从切换帧后返回主从切换回复帧，并打开 ARM 电源，从系统 ARM 处理器接管 CAN 总线。从系统 ARM 处理器正常启动后向主系统发送工程遥测查询帧，表示从系统已进入主状态，主系统稳定进入从状态，系统冗余备份管理权平稳交接。若从系统 ARM 故障，无法发送工程遥测帧，主系统 FPGA 等待一段时间后重新进入主状态成为 CAN 总线主机。

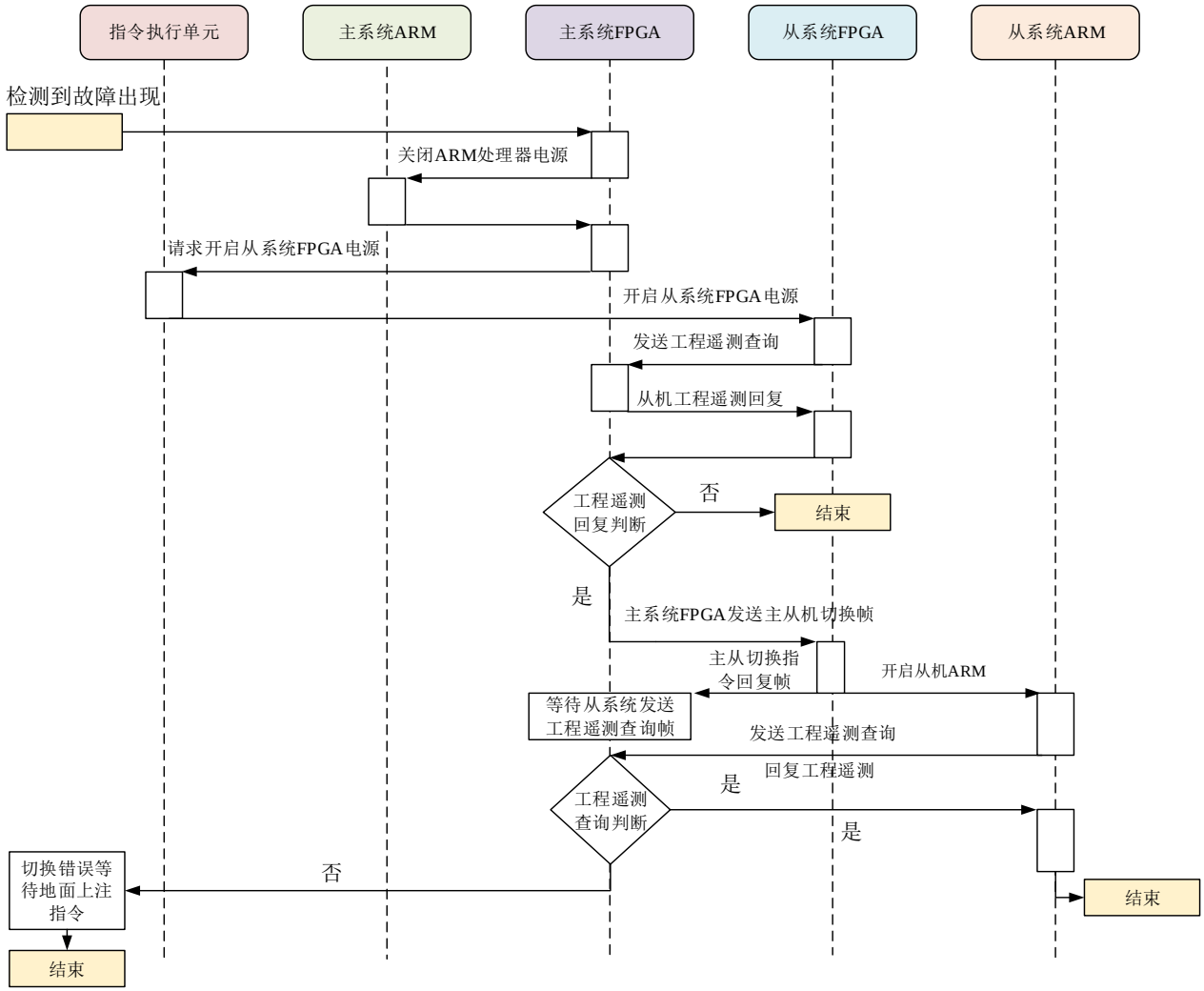

图 4.8 FPGA 主从切换流程

4.3.3 主从切换设计分析

父级仲裁切换流程每执行一步操作，综合电子系统都要将新的健康信息发送至指令执行单元作为判决依据，判决完成后指令执行单元再发送下一条指令控制切换流程继续执行。这种多次的状态与控制信息交互使得综合电子系统的自主性不强。但指令执行单元在卫星各分系统中的可靠性级别高，因此采用父级仲裁切换的方式具有更高的可靠性。

同级自主自切换方案下，主从切换流程由综合电子系统控制，不依赖其他分系统检测切

换，相比于父级仲裁切换减小了模块之间的耦合性且支持更为复杂的故障检测仲裁逻辑。该方案下综合电子系统的自主性高，通过 CAN 总线通信的可靠性也比指令执行单元 SPI 接口通信的高。但由于 ARM 处理器不区分主从状态，从系统的 ARM 处理器处于关机状态，因此主从切换速度慢。

两种备份管理方式的实时性也不同。从地面上注指令进行双系统主从状态切换时采用不同备份管理方式的耗时方面可以只直观的看出实时性差异。不同管理方式的地面上注指令切换耗时如表 4.4 所示，可以看出父级仲裁切换的实时性强于同级自主切换。

表 4.4 地面上注指令切换用时

模式	主从切换操作	次数	用时
父级仲裁切换	A 系统切换 B 系统	9	8.3±2.1s
	B 系统切换 A 系统	8	7.9±2.6s
同级自主切换	A 系统切换 B 系统	12	25.6±1.5s
	B 系统切换 A 系统	12	26.1±1.3s

综合 ZHX 功能、性能需求，选择父级仲裁切换模式作为综合电子系统的系统级冗余备份管理方式。

4.4 主从系统信息同步

综合电子系统实现星务管理、执行卫星任务的过程中会产生一些系统信息。星务处理单元上电时通过读取系统信息配置当前的工作模式与即将执行的卫星任务等星务状态。当系统主从切换后，从机系统接替主系统工作，需要继续执行主系统原有的卫星任务及星务管理工作，因此需要同步主从系统的系统信息。

系统信息主要分为系统公有信息与系统私有信息两部分。系统公有信息是指主、从系统需要进行同步的信息，包括软件版本号、遥测模式选择、整星模式选择、准禁标识、姿轨控数据等参数信息。系统私有信息是指主从系统执行星务任务过程中独有的不需要同步的信息，包括 IP 端口号，TCP 交互是否重发等参数信息。

系统信息存储于铁电(F-RAM)中。F-RAM 是一种随机存取存储器，具有非易失性，且读写速度快，写入数据过程中无等待数据写入存储阵列时间。同时 F-RAM 具备良好的抗辐射性，其抗辐射能力相比于 EEPROM 高出一个数据级以上[80-81]，且可读写次数为10^{14}次。 F-RAM 数据容量为 512KB，系统公有信息约为 8KB，私有信息约为 1KB 左右，数据存储结构

如图 4.9 所示。公有信息与私有信息都由 ARM 处理器生成、修改，数据均采用 CRC16 编码校验。

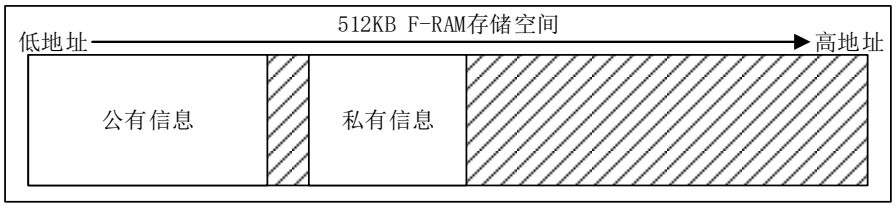

图 4.9 系统信息存储结构图

主从信息同步功能由 FPGA 实现。主机系统将系统信息发送给指令执行单元，指令执行单元转发至从机系统，从机系统接受到来自主机系统的系统信息后写入 F-RAM。

系统信息同步的功能结构如图 4.10 所示，FPGA 通过 MUX 实现主从系统信息同步及系统信息修改功能。FPGA 同步主从系统信息时，MUX 选择 F-RAM 控制器模块与 F-RAM 通信；ARM 处理器修改系统信息时，MUX 选择 ARM 处理器与 F-RAM 通信。

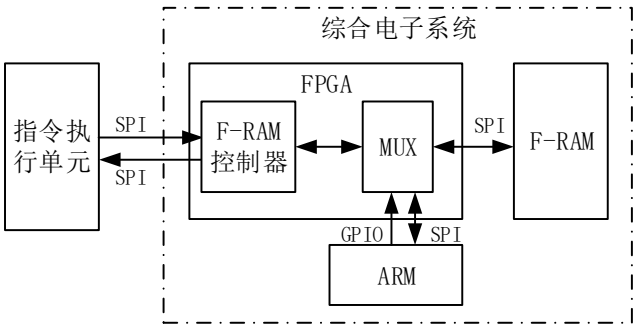

图 4.10 系统信息同步功能结构

主从系统信息同步与系统信息修改发生冲突时，优先执行系统信息修改操作。F-RAM 的硬件特性使得写数据无等待数据写入存储阵列耗时，因此通过 ARM 处理器 SPI 接口的 CS 信号中断主从系统信息同步过程。FPGA 检测到 CS 拉低后中断系统信息同步流程，MUX 选择 ARM 处理器与 F-RAM 通信。系统信息修改结束后，FPGA 继续中断的主从系统信息同步流程。ARM 处理器修改公有信息后，置高 GPIO 通知 FPGA 同步系统信息；修改私有信息后，置低 GPIO 通知 FPGA 无需同步系统信息。

4.5 本章小结

本章详细地介绍了双系统冗余备份管理方案。首先介绍了双系统冗余备份结构下管理权冲突可引发的故障。接着介绍了冗余备份的管理方法和故障判决的信息来源。在此基础上介绍了两种双系统切换检测仲裁单元设计以解决管理权冲突问题及平稳可靠交接，并从可靠性、工作状态与实时性方面评估两种设计。最后介绍了主从系统的系统信息同步机制。

5.综合电子系统容错技术测试与分析

前面的章节细致的介绍了操作系统星载可靠性加载设计与双系统冗余备份管理研究，本章节将对这些内容进行原理样机测试，介绍测试系统、测试方案，并根据测试结果综合评估容错设计性能。

5.1 综合电子系统容错技术原理样机

综合电子系统的原理样机硬件设计如图 5.1 所示。原理样机的 ARM 处理器采用的 TI 公司的 AM5748。调试 FPGA 采用的是 Microsemi 公司的 A3PE3000L 并以接插件的形式安装在综合电子系统板卡上，一旦功能调试完成取下接插件即可落焊反熔丝 FPGA。eMMC 存储器放置于 TX2 下，采用的是 Micron 公司的的 MTFC32GJVED-4M IT(8GB)和 MTFC32GJVED-4M IT(32GB)；SPI NOR Flash 放置于板子背面，采用 Micron 公司的 MT25QL02GCBB(256MB)；NAND Flash 放置于板子背面，采用 Micron 公司的 MT29F16G16ADACA(2GB)。

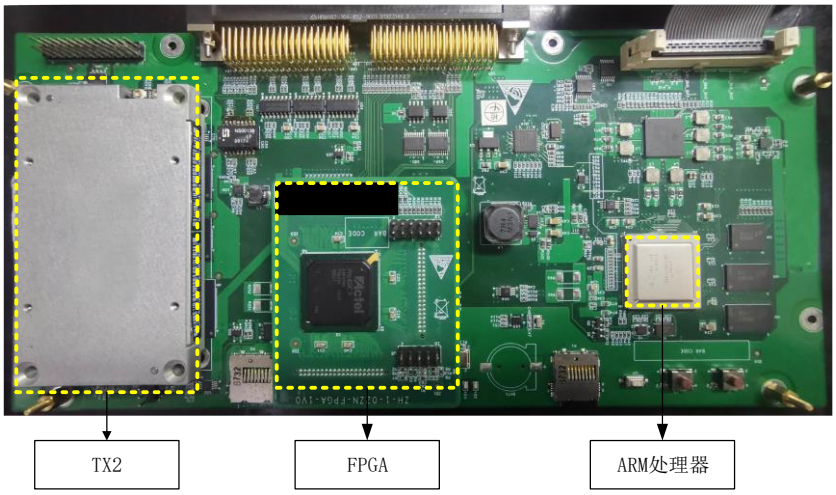

图 5.1 可靠性加载设计原理样机

双系统冗余备份管理方案研究原理样机由综合电子系统、指令执行单元和收发处理组件组成，图 5.2 为系统冗余研究原理样机结构。PC 机通过遥控、遥测软件上注指令、接收遥测。

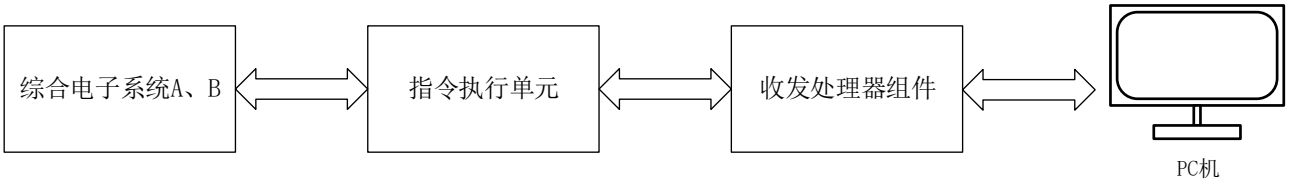

图 5.2 双系统冗余备份管理方案测试原理样机

5.2 操作系统星载可靠性加载设计测试与分析

操作系统星载可靠性加载主要解决单粒子翻转故障。地面进行单粒子翻转测试的实验条件苛刻、实验成本较高，因此采用软件模拟的方式进行操作系统星载可靠性加载设计的功能测试[82]。实验测试系统包括 PC 主机和测试原理样机两部分，PC 主机通过控制台串口与原理样机上的 ARM 处理器连接，执行翻转数据注入及数据读写功能，并接收返回的测试结果。

5.2.1 原理样机测试方案及内容

测试方案分为完备性测试和速度性能测试、资源消耗三个部分。完备性测试是为验证操作系统星载可靠性加载设计能否对关键数据所有 bit 位实现翻转数据纠错功能，通过软件模拟单粒子翻转的最坏情况，即备份关键数据每一 bit 位都发生了单粒子翻转。生成的测试备份关键数据特点如表 5.1 所示，表中 U 表示模拟该位数据发生单粒子翻转。序号 1 的数据为正常的关键数据，通过软件对序号 1 数据以 bit 形式随机替换数据形成序号 2 数据，再对序号 2 数据逐位取反得到序号 3 的数据，由此可得每一 bit 位都发生单粒子翻转的测试备份关键数据。

表 5.1 测试备份关键数据

序号	数据内容						
1	0	1	1	0	0	0	1
2	0	U	1	U	0	0	U
3	U	1	U	1	U	U	1

速度性能测试用于测试接口的读写速度及操作系统可靠性加载效率。在原理样机的 FPGA 内设置计时模块。星载可靠性加载设计工作时触发计数模块开始计时，星载可靠性加载设计结束时触发计时模块停止计时并将时间值通过串口模块输出至 PC 机。根据读写数据大小及用时可以得出接口实际读写速度及操作系统可靠性加载用时。

资源消耗用于衡量可靠性加载设计的资源代价，反熔丝 FPGA 的资源有限，因此在设计时需要综合考虑资源消耗量。FPGA 代码编译工具在完成代码编译后可生成资源消耗表，根据表中的信息可得不同可靠性加载设计的资源消耗量。

5.2.2 基于 eMMC 存储器的可靠性加载设计测试与分析

在完备性测试前先要进行双向数据流向判决逻辑的功能测试。测试内容分为操作系统启动、U-Boot 阶段读写、RT-Linux 操作系统阶段读写、擦除三个部分。ARM 处理器进入 U-Boot

阶段通过控制台 mmc read 和 mmc write 命令测试 eMMC 读写功能。进入操作系统后，通过格式化分区命令制作 eMMC 存储器启动分区,并将关键数据写入 eMMC 存储器对应分区中。从 eMMC 存储器重启操作系统，通过 PC 机控制串口打印信息可确认 ARM 处理器是否从 eMMC 存储器启动操作系统。经测试，ARM 处理器 MMC1 接口时钟频率 48MHz、8bit DDR 传输模式下，双向数据流向判决逻辑功能正常符合设计预期。

完备性测试主要采用数据对比的方式验证，备份 eMMC 存储器的测试备份关键数据三模冗余后写入启动 eMMC 存储器。流程结束后比对启动 eMMC 存储器的关键数据与原数据是否一致验证翻转数据纠错功能。经测试，基于 eMMC 存储器的可靠性加载设计可实现翻转数据纠错功能，操作系统实现可靠加载。

eMMC 数据读写速度和使用的命令有关。主机读 eMMC 可使用单块读、多块读+终止传输、预设块数+多块读三种方法，写可采用单块写、多块写+终止传输、预设块数+多块写三种方法。实验测试的结果表明：单块读<多块读+终止传输<预设块数+多块读，单块写<多块写+终止传输<预设块数+多块写，且不同预设块数下读写速度也不同。

速度性能测试采用 FPGA 按地址顺序连续读写 53MB 数据的方法测试不同预设块数下数据读写速度，并对 53MB 测试备份关键数据进行数据翻转数据纠错，测试不同预设块数下的用时，每组共测试 8 次。测试时 eMMC 主机采用 8bit SDR 高速传输模式，通信时钟频率为 40MHz，测试结果如表 5.2 所示。预设块数较大超过 FIFO 缓存上限时，测试过程中忽略 FIFO 空满标志，过量读入的数据将溢出，过量写出的数据为 0x00。

表 5.2 基于 eMMC 存储器的可靠性加载设计读写性能测试结果

预设块数	读数据平均用时	写数据平均用时	翻转数据纠错平均用时
4	12.38s	48.52s	114.9s
8	6.65s	26.07s	61.89s
16	3.91s	13.60s	51.94s
32	2.56s	9.41s	47.68s
256	1.64s	6.39s	16.17s
8196	1.75s	4.35s	11.29s

将表 5.2 中的数据处理后得到图 5.3。从图 5.3 的(a)、(b)可以看出小块数时随着预设块数的增加，读速度提升，大块数时随着预设块数的增加，读速度不升反降；从(c)、(d)可得，随着预设块数的增加，写速度提升；从(e)可以看出随着预设块数的增加，翻转数据纠错耗时减少。预设块数的增加对应的需要更大的缓存空间，但反熔丝 FPGA 的 RAM 资源有限，可

靠性加载设计的性能因此受限。

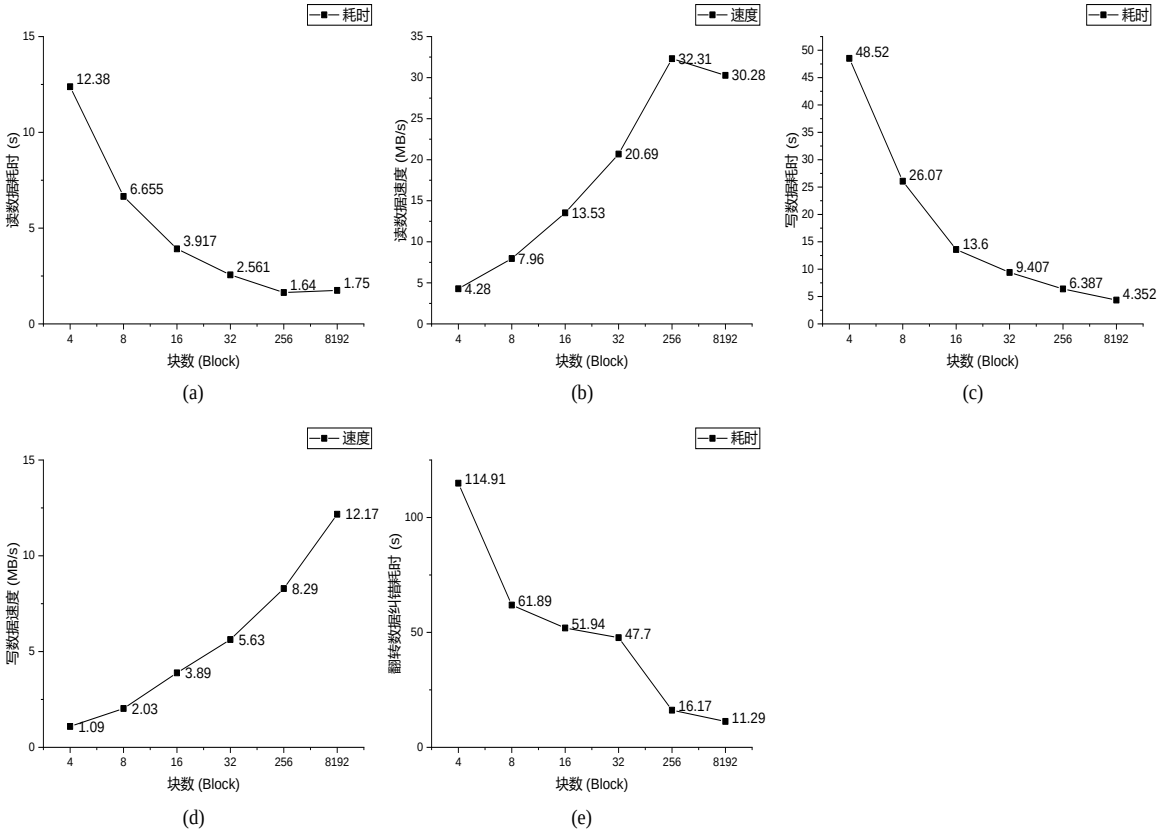

图 5.3 基于 eMMC 存储器的可靠性加载设计读写性能

使用 Libero SOC V11.8 工具对基于 eMMC 存储器的可靠性加载设计代码综合、编译，代码使用 VHDL 语言编写，功能板级测试使用 A3P1000L 系列 FPGA，多块读写预设块数为 4 块，三模冗余设计资源消耗结果如图 5.4 所示，双向数据流向判决资源消耗如图 5.5 所示。

CORE	Used:	3620	Total:	24576	(14.73%)
IO (W/ clocks)	Used:	24	Total:	300	(8.00%)
Differential IO	Used:	0	Total:	74	(0.00%)
GLOBAL (Chip+Quadrant)	Used:	6	Total:	18	(33.33%)
PLL	Used:	1	Total:	1	(100.00%)
RAM/FIFO	Used:	16	Total:	32	(50.00%)
Low Static ICC	Used:	0	Total:	1	(0.00%)
FlashROM	Used:	0	Total:	1	(0.00%)
User JTAG	Used:	0	Total:	1	(0.00%)

图 5.4 三模冗余设计资源消耗

CORE	Used:	641	Total:	24576	(2.61%)
IO (W/ clocks)	Used:	24	Total:	300	(8.00%)
Differential IO	Used:	0	Total:	74	(0.00%)
GLOBAL (Chip+Quadrant)	Used:	2	Total:	18	(11.11%)
PLL	Used:	0	Total:	1	(0.00%)
RAM/FIFO	Used:	0	Total:	32	(0.00%)
Low Static ICC	Used:	0	Total:	1	(0.00%)
FlashROM	Used:	0	Total:	1	(0.00%)
User JTAG	Used:	0	Total:	1	(0.00%)

图 5.5 双向数据流向判决资源消耗

可以看出三模冗余设计寄存器消耗量为 3620 个核，FIFO 资源消耗为 8KB，双向数据流

向判决逻辑寄存器消耗量为 641 个核。两个模块的功能独立，资源无复用部分，因此寄存器总消耗量为 4261 个核，FIFO 资源消耗 8KB，资源总体消耗量大。且由于缓存资源的限制，可靠性加载设计无法发挥最大性能。

eMMC 存储器的硬件特性降低了自身的硬件可靠性。eMMC 存储器留有固件 Firmware 烧写接口，固件信息存在单粒子翻转的可能。除此之外 eMMC 硬件控制器执行对 NAND Flash 的管理工作时产生的部分信息会存放在 eMMC 存储器中的制造商区域，这部分区域对用户屏蔽，用户无法通过命令寻址访问，综合电子系统无法纠错制造商区域的翻转数据。因此，eMMC 存储器的硬件可靠性较低。

5.2.3 基于 QSPI 接口的可靠性加载设计测试与分析

完备性测试主要采用系统启动及数据对比的方式验证。PC 机通过 ARM 处理器控制台串口将测试备份关键数据写入 SPI Flash 中。执行翻转纠错流程后，ARM 处理器从 QSPI 接口加载 U-Boot，通过串口打印信息可知 U-Boot 是否正常启动，串口打印信息如图 5.6 所示。ARM 处理器在 U-Boot 阶段读取 53MB 关键数据与原数据对比进一步验证可靠性加载设计的翻转数据纠错功能。经测试，基于 QSPI 接口的可靠性加载设计实现翻转数据纠错功能。

```
U-Boot SPL 2018.01 (Dec 23 2020 - 19:56:48)
DRA762-GP ES1.0 ABZ package
spl_boot_list[0] = a
Trying to boot from SPI
spi_flash_probe----43
ti_qspi_ofdata_to_platdata: regs=<0x4b300000>, max-frequency=12000000
ti_qspi_probe
spi_get_bus_and_cs:dev_name = spi_flash@0:0
ti_qspi_set_speed:max_hz = 12000000
spi_flash_scan2
ti_qspi_set_speed:max_hz = 12000000
spl_spi_load:payload_offs = 40000
spl_spi_load:offs_override = 0
no pinctrl state for default mode
no pinctrl state for default mode
Card did not respond to voltage select!
*** Warning - MMC init failed, using default environment

headermagic = d00dfeed
Found FIT
Jumping to U-Boot
loaded - jumping to U-Boot...
display_options_get_banner_priv:------

U-Boot 2018.01 (Dec 23 2020 - 19:56:48 +0800)



U-Boot 2018.01 (Dec 23 2020 - 19:56:48 +0800)

U-Boot code: 80800000 -> 80873D40  BSS: -> 808B1120
CPU  : DRA762-GP ES1.0 ABZ package
Model: TI AM5748 IDK
Board: AM574x IDK REV
DRAM:  Monitor len: 000B1120
Ram size: 40000000
Ram top: C0000000
TLB table from bfff0000 to bfff5000
```

图 5.6 QSPI 启动串口打印信息

速度性能测试内容分为可靠性加载准备用时、U-Boot 启动用时，QSPI 接口读数据速度、NAND Flash 数据写入用时四个部分。可靠性加载准备用时和 U-Boot 启动用时通过 FPGA 计时模块获得，QSPI 接口读数据速度通过 U-Boot 阶段控制台 sf read 读命令测试获得，NAND Flash 数据写入用时通过 nand write 写命令测试获得。测试时 ARM 处理器 QSPI 接口采用标

准 SPI 模式，Flash 控制器采用 4 线-SPI 模式，时钟频率为 20MHz，具体性能如表 5.3 所示。

表 5.3 基于 QSPI 接口可靠性加载速度性能测试结果

功能	性能指标
可靠性加载准备用时	345μs
U-Boot 启动用时	2.106s
QSPI 12MHz 时钟读速度	0.768MB/s
QSPI 24MHz 时钟读速度	1.263MB/s
Nand Flash 53MB 数据写入用时	23.6s

可靠性加载设计翻转数据纠错总耗时约 69.77s，主要耗时来自于 ARM 处理器在 U-Boot 阶段读取 53MB 备份关键数据。此阶段下 ARM 处理器 QSPI 接口速度提升至 24MHz，采用标准 SPI 模式，实际读数据速度约为 1.26MB/s。受限于 ARM 处理器接口驱动的性能，翻转数据纠错耗时较长。

使用 Libero SOC V11.8 工具对基于 QSPI 接口的可靠性加载设计代码综合、编译，代码使用 VHDL 语言编写，板级功能测试采用 A3PE3000L 系列 FPGA，模块的资源消耗结果如图 5.7 所示。从图中信息可得虚拟 SPI Flash 模块的寄存器资源消耗为 1964 个核，FIFO 缓存资源只使用了 5 个区域 RAM(1.75KB)，总体资源消耗较少。

CORE	Used:	1964	Total:	75264	(2.61%)
IO (W/ clocks)	Used:	14	Total:	341	(4.11%)
Differential IO	Used:	0	Total:	168	(0.00%)
GLOBAL (Chip+Quadrant)	Used:	5	Total:	18	(27.78%)
PLL	Used:	1	Total:	6	(16.67%)
RAM/FIFO	Used:	5	Total:	112	(4.46%)
Low Static ICC	Used:	0	Total:	1	(0.00%)
FlashROM	Used:	0	Total:	1	(0.00%)
User JTAG	Used:	0	Total:	1	(0.00%)

图 5.7 基于 QSPI 接口的可靠性加载设计资源消耗

虚拟 SPI Flash 设计相比于启动、备份 SPI Flash 设计方案在硬件可靠性上也有较大提升。采用启动、备份 SPI Flash 设计方案实现三模冗余功能时构成串联型结构模型，启动、备份 SPI Flash 任一存储器损坏都将导致三模冗余功能丧失。设 SPI Flash 可靠性概率服从损坏率为 λ_F 的泊松分布，可靠性概率 $P_F(T)=e^{-\lambda_F T}$。启动、备份 SPI Flash 正常工作视为随机独立事件，SPI Flash 可靠性概率 $P_F(T)$ 结合公式(3-1)可计算出启动、备份 SPI Flash 设计的硬件可靠性 $P_{SB}(T)$ 如公式(5.1)所示。

$$P_{SB}(T)=P_F^{\,2}(T) \tag{5.1}$$

相同硬件数量下虚拟 SPI Flash 设计为并联结构，根据 SPI Flash 可靠性概率 $P_F(T)$ 结合公式(3-2)可计算出硬件可靠性 $P_{VF}(T)$ 如公式(5.2)所示。

$$P_{VF}(T)=1-\left(1-P_F(T)\right)^2 \tag{5.2}$$

虚拟 SPI Flash 与启动、备份 SPI Flash 设计的可靠性差异 $P_{VF}(T)/P_{SB}(T)$ 如图 5. 8 所示。

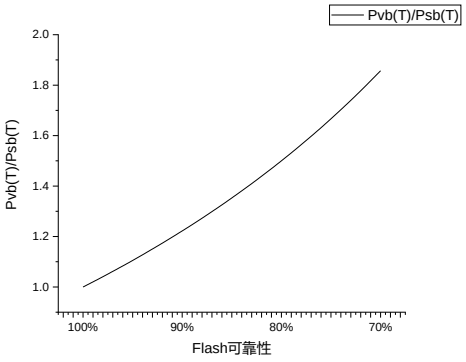

图 5. 8　可靠性差异 $P_{VF}(T)/P_{SB}(T)$ 关系图

从图中可得随着 SPI Flash 的硬件可靠性降低，虚拟 SPI Flash 设计的硬件可靠性高于启动、备份 SPI Flash 设计，在最悲观的硬件可靠性需求条件下，即三年期硬件可靠性为 70%，虚拟 SPI Flash 设计的硬件可靠性约比启动、备份 SPI Flash 设计提升了 85%。

5.2.4 基于 SD 接口的可靠性加载设计

完备性测试主要采用系统启动及数据对比的方式验证。通过 PC 的控制台串使 ARM 处理器将测试备份关键数据分别写入 3 片 SPI Flash 中。执行翻转纠错流程后，ARM 处理器从 SD 接口加载系统，通过串口打印信息可得知系统是否正常启动，串口打印信息如图 5. 9 所示。

```
U-Boot SPL 2018.01 (Dec 23 2020 - 19:56:48)
DRA762-GP ES1.0 ABZ package
spl_boot_list[0] = 5
Trying to boot from MMC1
no pinctrl state for default mode
no pinctrl state for default mode
Card did not respond to voltage select!
*** Warning - MMC init failed, using default environment

Jumping to U-Boot
loaded - jumping to U-Boot...
display_options_get_banner_priv:------

U-Boot 2018.01 (Dec 23 2020 - 19:56:48 +0800)



U-Boot 2018.01 (Dec 23 2020 - 19:56:48 +0800)

U-Boot code: 80800000 -> 80873D40  BSS: -> 808B1120
CPU  : DRA762-GP ES1.0 ABZ package
Model: TI AM5748 IDK
Board: AM574x IDK REV
DRAM:  Monitor len: 000B1120
Ram size: 40000000
Ram top: C0000000
TLB table from bfff0000 to bfff5000
```

图 5. 9　SD 接口启动串口打印信息

ARM 处理器在 U-Boot 阶段读取 53MB 的关键数据与原数据对比进一步验证翻转数据纠错功能。经测试，基于 SD 接口的可靠性加载设计实现翻转数据纠错功能。

速度性能测试内容为 SD 接口读数据速度。SD 接口读数据速度通过 U-Boot 阶段控制台 mmc read 读命令测试获得。ARM 处理器的 SD 接口时钟频率为 24MHz，SPI Flash 控制器采用 4 线 QSPI 模式，钟频率为 20MHz 时的测试结果如表 5. 4 所示。

表 5. 4 基于 SD 接口的可靠性加载设计速度性能测试结果

功能	性能指标
SD 接口读 32MB 数据用时	5.41s
SD 接口读数据速度	5.91MB/s

翻转数据纠错总耗时约为 34.75s。基于 SD 接口的可靠性加载设计充分利用了 NAND Flash 写数据速度快，NOR Flash 读数据速度快的物理特点，最大程度的优化了操作系统可靠性加载效率。

使用 Libero SOC V11.8 工具对基于 SD 接口的可靠性加载设计代码综合、编译，代码使用 VHDL 语言编写，板级功能测试采用 A3PE3000L 系列 Flash 型 FPGA，模块资源消耗的结果如图 5. 10 所示。从资源消耗结果中可以看出，模块的寄存器资源消耗较多共消耗了 4892 个核，这主要是因为虚拟 SD 模块中需要预置的 SD 寄存器参数值较多，但 FIFO 缓存资源的消耗较少仅使用了 4 个区域的 RAM(2KB)。

CORE	Used:	4892	Total:	75264	(6.50%)
IO (W/ clocks)	Used:	36	Total:	341	(10.56%)
Differential IO	Used:	0	Total:	168	(0.00%)
GLOBAL (Chip+Quadrant)	Used:	6	Total:	18	(33.33%)
PLL	Used:	1	Total:	6	(16.67%)
RAM/FIFO	Used:	4	Total:	112	(3.57%)
Low Static ICC	Used:	0	Total:	1	(0.00%)
FlashROM	Used:	0	Total:	1	(0.00%)
User JTAG	Used:	0	Total:	1	(0.00%)

图 5. 10 基于 SD 接口的可靠性加载设计代码资源消耗

为提升模块硬件可靠性，硬件结构上额外设计了一片 Flash 用于硬件冗余冷备。从 5.2.4 节得单片 Flash 存储器的可靠性为 $P_F(T)=e^{-\lambda_F T}$，则基于 SD 接口的可靠性加载设计的硬件可靠性 $P_{SD}(T)$ 如公式(5.3)所示。

$$P_{SD}(T)=\sum_{i=3}^{4}C_4^i P_F{}^i(T)\left(1-P_F(T)\right)^{4-i} \tag{5.3}$$

相同硬件数量下基于 QSPI 接口的可靠性加载设计的硬件可靠性 $P_{VF'}(T)$ 如公式(5.4)所示。

$$P_{VF'}(T)=\left(1-P_F(T)\right)^4 \tag{5.4}$$

$P_{SD}(T)/P_{VF'}(T)$用于表示基于 SD 接口的可靠性加载设计的硬件可靠性与基于 QSPI 接口的可靠性加载设计的硬件可靠性差异，结果如图 5. 11 所示。

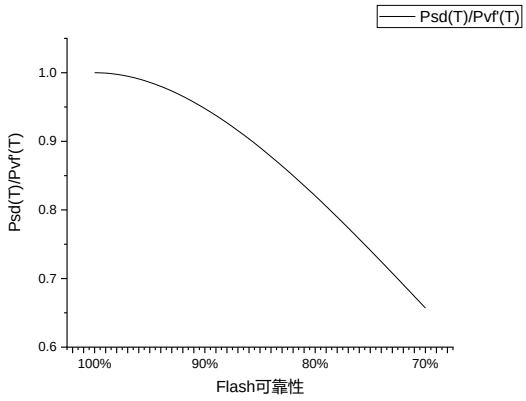

图 5. 11 可靠性差异$P_{SD}(T)/P_{VF'}(T)$关系图

从图中可得出随着单片 Flash 的可靠性降低，采用的相同硬件数量，基于 SD 接口的可靠性加载设计的硬件可靠性远低于基于 QSPI 接口的可靠性加载设计的硬件可靠性，在最悲观的预估方式下，即三年期硬件可靠性为 70%，基于 SD 接口的可靠性加载设计的硬件可靠性约为基于 QSPI 接口的可靠性加载设计的 65%。

5.2.5 可靠性加载设计小结

综合上述的测试与分析，整理出三种可靠性加载设计性能汇总表，如表 5. 5 所示。

表 5. 5 三种可靠性加载设计性能汇总

可靠性加载设计	翻转数据纠错用时	硬件可靠性	资源消耗量	可移植性
基于 eMMC 存储器	约 114.9s	低	多	良好移植性
基于 QSPI 接口	约 69.77s	高	少	弱移植性
基于 SD 接口	约 34.75s	中等	多	强移植性

从表中可看翻转数据纠错用时基于 SD 接口的可靠性加载设计用时最少，而基于 eMMC 存储器的可靠性加载设计因缓存资源的限制，无法充分发挥性能用时最长。硬件可靠性方面基于 QSPI 接口的可靠性加载设计最高，基于 eMMC 存储器的可靠性加载设计因芯片特性致使硬件可靠性最低。资源消耗方面基于 eMMC 存储器消耗的可靠性加载设计缓存资源最多，基于 SD 接口的可靠性加载设计寄存器资源消耗最多，基于 QSPI 接口的可靠性加载设计资源消耗最少。可移植性方面，基于 eMMC 存储器的可靠性加载设计采用一级 boot 方式具有良好的移植性；基于 QSPI 接口的可靠性加载设计需要预设长度信息且执行预读取策略，可移植

性弱；基于 SD 接口的可靠性加载设计实现了处理器接口与物理存储器接口的脱钩，不同的数据存储器都能以 SD 接口的形式被处理器访问，因此具有强可移植性。

综合评估三种可靠性加载设计的性能、特点，ZHX 采用基于 SD 接口的可靠性加载设计实现操作系统的星载可靠性加载。

5.3 双系统冗余主从架构测试

5.3.2 父级仲裁切换测试

父级仲裁切换测试主要验证检测仲裁逻辑功能是否正确。通过软件、硬件设置模拟故障类型，通过查看遥测软件信息验证仲裁逻辑功能是否正常。硬件上断开 ARM 处理器电源模拟关机故障状态，软件上将双系统星务应用程序的主从初始状态都设置为主系统或从系统模拟系统双主、双从故障状态;将星务应用程序的主从状态标志信息强制设置为 0 表示 ARM 处理器只为从状态，模拟不响应主从状态切换的故障状态。测试结果如表 5.6 所示，表中的“/”表示忽略状态信息。

表 5.6 父级仲裁切换测试结果

模拟故障类型	A 系统状态	B 系统状态	系统状态结果
双主	/	/	A 系统主 B 系统从
双从	不响应主从切换	响应主从切换	A 系统从 B 系统主
	响应主从切换	/	A 系统主 B 系统从
	不响应主从切换	不响应主从切换	A 系统从 B 系统从
A 从 B 关机	不响应主从切换	响应主从切换	A 系统从 B 系统主
	响应主从切换	/	A 系统主 B 系统关
	不响应主从切换	不响应主从切换	A 系统从 B 系统从
B 从 A 关机	响应主从切换	/	A 系统主 B 系统从
	不响应主从切换	响应主从切换	A 系统关 B 系统主
	不响应主从切换	不响应主从切换	A 系统从 B 系统从
双关机	响应主从切换	/	A 系统主 B 系统关
	不响应主从切换	响应主从切换	A 系统从 B 系统主
	不响应主从切换	不响应主从切换	A 系统从 B 系统从

从表中的结果可看出在不同的模拟故障状态下，父级仲裁切换都实现了正确的主从状态

切换，双系统冗余备份的管理权平稳交接过渡，功能满足设计需求。

5.3.3 同级自主切换测试

同级自主切换测试主要验证检测仲裁逻辑的功能是否正确。通过软件、硬件模拟故障类型，查看遥测信息验证功能是否正常。硬件上控制 ARM 处理器电源开关情况模拟故障类型。故障检测类型为星载任务支持时，打开 ARM 电源；故障检测类型为星务应用支持时，关闭 ARM 电源。软件上在 FPGA 内设置测试使用的故障注入模块，PC 机通过串口向 FPGA 注入故障参数。注入的故障参数可选择故障检测类型，控制从状态 FPGA 是否响应工程遥测查询以及是否打开 ARM 处理器电源，以此遍历主从切换流程中所有故障状态。测试结果如表 5.7 所示，表中的“/”表示忽略状态信息。

表 5.7 同级自主切换测试结果

故障检测类型	工程遥测查询状态	ARM 处理器电源状态	测试结果
星载任务支持	响应工程遥测查询	关闭电源	主从切换失败
	响应工程遥测查询	开启电源	主从切换成功
	不响应工程遥测查询	/	主从切换失败
星务应用支持	响应工程遥测查询	关闭电源	主从切换失败
	响应工程遥测查询	开启电源	主从切换成功
	不响应工程遥测查询	/	主从切换失败

表中的测试结果可以得出，同级自主切换针对不同的模拟故障执行的结果符合功能预期。同级自主切换实现了双系统冗余备份的管理权平稳交接过渡，功能满足设计需求。

5.4 本章小结

本章将综合电子系统容错技术在原理样机上进行联调测试，以验证操作系统星载可靠性加载设计与双系统冗余备份管理方案功能。本章首先对原理样机、测试系统进行简单介绍，接着介绍了可靠性加载设计的功能、性能测试方案与双系统冗余备份管理方案测试方案，最后根据测试结果综合分析评估各容错设计。测试结果表明，容错技设计的性能指标均满足功能需求。

6 总结与展望

6.1 本文总结

随着卫星技术的发展与卫星任务日益复杂，对星载处理器的性能提出了更高的要求。传统的航天级处理器的性能无法满足需求，而搭载操作系统的主流工业级处理器因其成本低、研发周期短、性能处理器等特点，成为星载处理器的一个热门选择方向。但由于工业级的电子设备抗辐射性能差，在太空高能粒子辐射环境中可靠性不足，易发生故障，因此需采用一定的容错技术增强其抗辐射性能。

本文基于 ZHX 卫星项目，旨在设计具有搭载高性能处理器、支持操作系统、在轨智能计算、系统可更换、可升级的综合电子系统硬件架构，并针对大容量操作系统数据可靠性加载及双系统冗余备份多处理器竞争机制，提出容错设计与管理方案提升系统可靠性。

本课题的主要工作总结如下：

(1) 根据 ZHX 卫星功能需求，设计了双系统冗余温备份架构，针对大数据量传输与总线容错需求，设计了高速、低速、备份总线结合的容错总线。针对特殊功能接口设计接口转换模块，提升 ARM 处理器软件驱动可靠性。

(2) 结合 RT-linux 两级 boot 启动方式与三模冗余容错技术，提出了三种基于不同接口的可靠性加载设计以满足 ARM 处理器大容量操作系统星载可靠性加载需求。三种容错设计都实现了关键数据的翻转数据纠错功能，提升 ARM 处理器在轨工作可靠性。

(3) 针对 ZHX 卫星可更换、可升级的需求，采取系统级冗余备份容错技术。为解决双系统多处理单元冗余备份管理混乱问题，采用主从模式管理，并根据检测切换单元的逻辑不同，提出仲裁式、自主式两种系统级冗余备份管理方案实现系统主从状态切换。针对双系统之间重要系统信息的交互需求，提升双系统冗余备份下备份系统接替工作系统的实时性与准确性，设计了系统信息同步机制。

(4) 在 ZHX 综合电子系统原理样机与系统冗余备份管理方案研究的原理样机上进行容错设计测试，并从翻转数据纠错用时、资源消耗、硬件可靠性、可移植性几个维度综合评估星载可靠性加载设计。实验与评估结果表明，操作系统星载可靠性加载设计与双系统冗余备份管理方案功能符合 ZHX 卫星需求。

6.2 工作展望

本文在对 ARM 处理大容量操作系统星载可靠性加载需求方面提出了三种容错设计，对系统级冗余温备份管理机制方面提出了两种管理机制设计并都通过原理样机测试进行了功能测试时，但是在整个研究、设计过程中仍有一些问题，值得进一步的研究。

(1) 在操作系统星载可靠性加载设计中 FPGA 处理数据时未使用 EDAC 信息冗余容错技术，为进一步提升数据存储、传输的可靠性，可根据 FPGA 资源增加一定的 EDAC 纠错功能。

(2) 基于 SD 接口的可靠性加载设计中虚拟 SD 卡模块只具备读数据功能，在功能上仅支持系统启动加载阶段的数据容错。未来可增加写数据功能，实现对 ARM 处理器操作系统运行全过程的读写数据实时三模冗余保护，大大地提升了 ARM 处理器在轨工作的可靠性。

(3) 系统级冗余备份管理方案中的管理逻辑较为简单，面对复杂的系统故障可能会失效，未来可本文研究的基础上针对更为复杂的故障情况设计更完善的管理逻辑。

参考文献

[1] Toorian A, Diaz K, Lee S. The cubesat approach to space access[C]. 2008 IEEE Aerospace Conference. IEEE, 2008: 1-14.

[2] 王鑫, 张妍, 尹玉明, 尹佳琪. 微小卫星标准化现状的分析及思考[J]. 中国航天, 2018(12): 36-40.

[3] 刘帅, 王虎妹. 卫星综合电子系统体系结构总体技术研究[J]. 空间电子技术, 2015, 12(6): 90-94.

[4] 王晓海. 蓬勃发展的现代小卫星[J]. 中国航天, 2002(1): 9-14.

[5] 王九龙. 卫星综合电子系统现状和发展建议[J]. 航天器工程, 2007(5): 68-73.

[6] 李孝同, 施思寒, 李冠群. 微小卫星综合电子系统设计[J]. 航天器工程, 2008(1): 30-35.

[7] Hillman R, Swift G, Layton P, et al. Space processor radiation mitigation and validation techniques for an 1,800 MIPS processor board[C]. Proceedings of the 7th European Conference on Radiation and Its Effects on Components and Systems, 2003. RADECS 2003. IEEE, 2003: 347-352.

[8] LaBel K A, Gates M M, Moran A K, et al. Commercial microelectronics technologies for applications in the satellite radiation environment[C]. 1996 IEEE Aerospace Applications Conference. Proceedings. IEEE, 1996, 1: 375-390.

[9] 张景楠, 李华旺, 朱野. 微小卫星星务分系统的硬件设计与实现[J]. 电子设计工程, 2014, 22(15): 177-179.

[10] 冯田雨, 陈健, 王峰. 微纳卫星高性能综合电子系统设计[J]. 光学精密工程, 2020, 28(9): 2056-2064.

[11] 史简, 宋智, 李国军. “天绘一号”卫星星务分系统研究与实现[J]. 遥感学报, 2012, 16(S1): 74-77.

[12] Speer D, Jackson G, Raphael D. Flight computer design for the Space Technology 5 (ST-5) mission[C]. Proceedings, IEEE Aerospace Conference. IEEE, 2002, 1: 1-1.

[13] Perriault N, Célérier B, Dussy S. Medium to Low Class Gyros for Spacebus 4000 Application[C].

Guidance, Navigation and Control Systems. 2006, 606.

[14] 曲峰, 崔刚, 杨孝宗. TS—1．1 小卫星星务计算机系统设计[J]. 计算机工程与科学, 2002, 24(2): 96-98.

[15] 盖建宁, 樊友诚, 沈莉, 李杰. 萤火一号火星探测器综合电子分系统方案[J]. 上海航天, 2013, 30(4): 139-146.

[16] 叶伟松, 刘海颖, 陈志明,等. "天巡一号"微小卫星数据综合系统设计与在轨性能评估[J]. 南京航空航天大学学报, 2012, 44(6): 797-802.

[17] 陈建新, 张志, 王磊, 李志平, 余志鸿, 官经刚, 裴楠, 魏然. 嫦娥三号巡视器综合电子系统的设计与实现[J]. 中国科学:技术科学, 2014, 44(5): 450-460.

[18] 褚佳承. 皮卫星柔性综合电子系统技术研究[D]. 浙江大学, 2018.

[19] 张佳慧. 基于 Smart-OSEK OS 的综合电子系统设计[D]. 浙江大学, 2017.

[20] 徐喆垚, 陈宇坤, 张义超, 曹梦磊, 孙戎. LH-2 微小卫星综合电子系统设计及姿态控制研究[J]. 载人航天, 2020, 26(1): 88-93.

[21] 刘承坤. 基于 FPGA 在轨数据处理技术研究[D]. 北京邮电大学, 2015.

[22] 王大鹏, 赵培, 刘立祥. 空间信息网发展态势及关键技术[J]. 国际太空, 2016(4): 42-47.

[23] Bradley B, Block C, Chavez R, et al. SPACENET: On-Orbit Support in 2025[R]. AIR WAR COLL MAXWELL AFB AL, 1996.

[24] 陈世淼, 倪淑燕, 廖育荣. 微小卫星综合电子系统综述[J]. 空间电子技术, 2020, 191(5): 86-91.

[25] Surratt M, Loomis H H, Ross A A, et al. Challenges of Remote FPGA Configuration for Space Applications[C]. Aerospace Conference. IEEE, 2005: 1-9.

[26] Ramesh B, Bretschneider T, McLoughlin I V. Embedded linux platform for a fault tolerant space based parallel computer[C]. Proceedings of the Real-Time Linux Workshop. 2004: 39-46.

[27] Chien S, Doubleday J, Thompson D, et al. Onboard autonomy on the intelligent payload experiment (IPEX) CubeSat mission: a pathfinder for the proposed HyspIRI mission intelligent payload module[C]. Proc 12th International Symposium in Artificial Intelligence, Robotics and Automation in Space, Montreal, Canada. 2014.

[28] 黄影. 星载 COTS 计算机的体系结构设计及其抗 SEU 研究[D]. 国防科学技术大学. 2006.

[29] Benton E R, Benton E V. Space radiation dosimetry in low-Earth orbit and beyond[J]. Nucl Instrum Methods Phys Res B, 2001, 184(1-2): 255-294.

[30] 王同权, 沈永平, 王尚武, 张树发. 空间辐射环境中的辐射效应[J]. 国防科技大学学报, 1999(4): 36-39.

[31] 李桃生, 陈军, 王志强. 空间辐射环境概述[J]. 辐射防护通讯, 2008(2): 1-9.

[32] Underwood C I. Observations of radiation in the space radiation environment and its effect on commercial off-the-shelf electronics in low-Earth orbit[J]. Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 2003, 361(1802): 193-197.

[33] 丁义刚. 空间辐射环境单粒子效应研究[J]. 航天器环境工程, 2007, 24(5): 283-290.

[34] 李毅, 李瑞, 黄影, 刘东, 张春元. 基于 COTS 的空间信息处理系统单粒子闭锁保护技术实现[J]. 宇航学报, 2007(5): 1283-1287.

[35] Koga R, Ferro R J, Mabry D J, et al. Ion-induced sustained high current condition in a bipolar device[J]. IEEE transactions on nuclear science, 1994, 41(6): 2172-2178.

[36] 张昊, 王新升, 李博, 等. 微小卫星单粒子闩锁防护技术研究[J]. 红外与激光工程, 2015, 44(5): 1444-1449.

[37] 王长河. 单粒子效应对卫星空间运行可靠性影响[J]. 半导体情报, 1998, 35(1): 1-8.

[38] Dodd, E P., Sexton, et al. Impact of technology trends in SEU in CMOS SRAMs.[J]. IEEE Transactions on Nuclear Science, 1996, 43(6): 2797-2797.

[39] Amusan O A, Witulski A F, Massengill L W, et al. Charge Collection and Charge Sharing in a 130 nm CMOS Technology[J]. IEEE Transactions on Nuclear Science, 2006, 53(6): 3253-3258.

[40] Heidel D F, Rodbell K P, Oldiges P, et al. Single-Event-Upset Critical Charge Measurements and Modeling of 65 nm Silicon-on-Insulator Latches and Memory Cells[J]. IEEE Transactions on Nuclear Science, 2006, 53: 3512-3517.

[41] 郭林. 基于 FPGA 的星载机容错技术研究与设计[D]. 清华大学, 2009.

[42] 周宇杰. 基于双模冗余的立方星高可靠星载计算机设计[D]. 南京理工大学, 2016.

[43] 丁志远, 刘波, 刘超伟, 王婧, 王振华. 航天器热备份控制计算机软故障前向恢复方法[J]. 空间控制技术与应用, 2016, 42(6): 47-51.

[44] 李利军. 星载双机热备份计算机系统设计[D]. 西安电子科技大学, 2010.

[45] Dijck G V. Information Theoretic Approach to Feature Selection and Redundancy Assessment (Informatietheoretische benadering voor selectie van kenmerken en inschatting van redundantie)[J]. Publication List XIII, 2008.

[46] Kanoun K, Kaaniche M, Beounes C, et al. Reliability growth of fault-tolerant software[J]. IEEE Transactions on Reliability, 1993, 42(2): 205-219.

[47] Bentoutou Y. A Real Time EDAC System for Applications Onboard Earth Observation Small Satellites[J]. IEEE Transactions on Aerospace & Electronic Systems, 2012, 48(1): 648-657.

[48] Longden L, Thibodeau C, Hillman R, et al. Designing a single board computers for space using the most advanced processor and mitigation technologies[J]. EUROPEAN SPACE AGENCY-PUBLICATIONS-ESA SP, 2002, 507: 313-316.

[49] 王平, 孙宁, 李华旺,等. 创新一号小卫星星载计算机控制系统设计[J]. 计算机工程, 2006, 32(18): 255-257.

[50] 李攀. 基于 eMMC 的星载大容量存储关键技术研究[D]. 哈尔滨工业大学,2017.

[51] 陈斌, 沈卫华, 朱岩, 梁耀明, 陈晓敏. 嫦娥二号卫星大容量存储器设计[J]. 航天器工程, 2011, 20(05): 99-104.

[52] 鲁芳旭, 刘翠海. RS 码的性能分析与仿真[J]. 数字技术与应用, 2020, 38(8): 25-27.

[53] Pignol M. How to cope with SEU/SET at system level?[C]. IEEE International On-line Testing Symposium. IEEE, 2005.

[54] Caffrey M, Morgan K, Roussel-Dupre D, et al. On-Orbit Flight Results from the Reconfigurable Cibola Flight Experiment Satellite (CFESat)[C]. FCCM 2009, 17th IEEE Symposium on Field Programmable Custom Computing Machines, Napa, California, USA, 5-7 April 2009, Proceedings. IEEE, 2009.

[55] CAN K, Teney D, Denis A. Design, construction, and test of a reliable, redundant on-board computer (OBC) for the OUFTI-1 CubeSat of the University of Li`ege[C]. Benelux URSI Forum. 2010.

[56] Lee Y K, Kim J. On-Board Computer design & implementation for Korean LEO Satellites[C]. 2014 14th International Conference on Control, Automation and Systems (ICCAS 2014). IEEE,

2014: 1484-1488.

[57] Mu Y, Hao W, Changju W U, et al. Space flight validation of design and engineering of the ZDPS-1A pico-satellite[J]. Chinese Journal of Aeronautics, 2012, 25(5): 725-738.

[58] 李日和. 微纳卫星高可靠星务计算机容错系统设计[D]. 南京理工大学, 2017.

[59] 朱明俊. 立方星星载计算机系统容错技术研究[D]. 南京理工大学, 2016.

[60] 吴生龙. 基于 COTS 的星载计算机硬件容错体系架构设计与实现[D]. 哈尔滨工程大学, 2019.

[61] 曹东坡, 胡晓惠, 赵军锁, 毛劲松. 基于 BM3803 的星载计算机系统软件开发[J]. 计算机工程与设计, 2011, 32(2): 524-526.

[62] 贾文涛. 高可靠星载双机备份系统的设计与评估[D]. 国防科学技术大学, 2010.

[63] 沈奇, 韦杰, 纪丙华, 王志国, 崔培林. 低成本高可靠综合电子系统集成技术[J]. 航天标准化, 2020(2): 1-4.

[64] 袁素春, 璩泽旭, 邵应昭. 一种低成本高可靠 FPGA 在轨可重构加载管理方案[J]. 空间电子技术, 2017, 14(3): 92-96.

[65] 齐健. NVIDIA Jetson TX2 平台:加速发展小型化人工智能终端[J]. 智能制造, 2017(5): 20-21.

[66] 崔阳, 赵笙罡, 周文妹, 刘彬, 王文川, 梁广. 一种高性能星载综合电子系统设计[J]. 现代电子技术, 2020, 43(12): 119-121,126.

[67] 刘蕾, 常亮, 李华旺. 星载计算机双冗余 CAN 总线模块设计与实现[J]. 电子设计工程, 2015, 23(21): 99-102.

[68] Vaidya N H. Comparison of duplex and triplex memory reliability[J]. IEEE Transactions on Computers, 1996, 45(4): 503-507.

[69] Seidleck C M, LaBel K A, Moran A K, et al. Single event effect flight data analysis of multiple NASA spacecraft and experiments; implications to spacecraft electrical designs[C]. Proceedings of the Third European Conference on Radiation and its Effects on Components and Systems. IEEE, 1995: 581-588.

[70] Oldham T R, Friendlich M, Howard J W, et al. TID and SEE Response of an Advanced Samsung 4Gb NAND Flash Memory[C]. Radiation Effects Data Workshop, 2007 IEEE. IEEE, 2007.

[71]谭维凤, 王淦, 窦骄, 崔耀中. 采用 eMMC 的新型小卫星大容量存储技术研究[J].西北工业大学学报, 2020, 38(S1): 107-112.

[72]包飞军. 硬件系统及器件级 eMMC 失效分析研究[D]. 华东师范大学, 2015.

[73]付剑. 星载计算机的硬件容错设计与可靠性分析[D]. 国防科学技术大学, 2009.

[74]JESD84-B51, Embedded Multi-Media Card (eMMC) Electrical Standard (5.1)[S]. USA:JEDEC SOLID STATE TECHNOLOGY ASSOCIATION, 2015.

[75]张永伟. 基于 COTS 元器件的星载综合电子设备研制[D]. 哈尔滨工业大学, 2017.

[76]朱席鼎. SDIO 主机/设备控制器设计[D]. 武汉科技大学, 2017.

[77]朱明俊, 周宇杰.一种低成本纳卫星星载计算机容错方法[J]. 航天器工程, 2016, 25(2): 52-57.

[78]王伟成. 星载计算机多机并行系统容错技术研究与设计[D].国防科学技术大学, 2010.

[79]Fayyaz M, Vladimirova T. Fault-tolerant distributed approach to satellite on-board computer design[C]. 2014 IEEE Aerospace Conference. IEEE, 2014: 1-12.

[80]单祥茹. 抗辐射性让 FRAM 技高一筹[J]. 中国电子商情(基础电子), 2015, 0(3): 23-23.

[81]高占占, 钟向丽, 侯鹏飞, 宋宏甲, 李波, 王金斌. 2T2C 铁电存储单元读写电路的单粒子翻转效应研究[J]. 湘潭大学学报(自然科学版), 2019(4): 76-83.

[82]王挥, 潘海燕, 沙李鹏. 嵌入式星载计算机故障注入系统[J]. 计算机测量与控制, 2011(10): 2335-2336.