硕士学位论文

基于 ICAP 的 SRAM 型 FPGA 抗单粒子翻转系统设计

作者姓名：王 番

指导教师：施敏华 研究员

中国科学院微小卫星创新研究院

学位类别：工学硕士

学科专业：计算机应用技术

培养单位：中国科学院微小卫星创新研究院

2022 年 6 月

Design of the SRAM-FPGA for anti-SEU system based on ICAP

A thesis submitted to

University of Chinese Academy of Sciences

in partial fulfillment of the requirement

for the degree of

Master of Science in Engineering

in Computer Application Technology

By

Wang Fan

Supervisor： Professor Shi Minhua

Innovation Academy for Microsatellite, Chinese Academy of Sciences

June 2022

摘 要

SRAM 型 FPGA 有着丰富的资源，计算性能强大，而且具有可重配置特性。凭借着这些优点，SRAM 型 FPGA 在航天领域应用非常广泛。但是在太空环境中存在着大量的高能粒子，而 SRAM 型 FPGA 对高能粒子非常敏感，易发生单粒子翻转现象（Single Event Upset，SEU），从而可能导致电路出现故障，引起卫星无法正常工作。针对此问题，国内外学者都做了广泛的研究。

本文采取的抗单粒子翻转方法是通过 FPGA 内部的 ICAP 接口，对配置数据进行回读刷新，并使用纠错能力更强的 RM(2,5)码来替代常用的 FRAME_ECC 纠错码，最高可以支持对 3 个比特位的翻转进行检错纠错，进一步提高了系统的纠错能力。除此外还对刷新器所在电路部分进行三模冗余加固处理，降低了刷新器出现单粒子翻转的可能性。并且对冗余模块增加错误判断，可及时识别冗余部分是否发生单粒子翻转，避免由于错误累积可能导致多个冗余模块发生故障的现象。通过以上措施的改进，提高了系统的可靠性。同时在布局布线过程中，将待刷新电路与刷新器电路部分进行分布式布局，使得对待刷新电路帧地址的确定更加方便。

最后通过在 Xilinx 公司开发的 Artix-7 芯片上，对设计的系统进行实验。通过 Chipscope 工具抓取系统信号并观察，以此验证系统功能的正确性。对系统进行故障注入来模拟单粒子翻转，并设计了三种不同类型的待测试电路，一是组合逻辑电路，二是时序逻辑电路，三是既有组合逻辑又有时序逻辑电路。以这三种电路来模拟真实情况下的用户电路，评估系统的性能。对这三种待测电路在传统内部配置刷新系统、三模冗余加固后的配置刷新系统以及本文所设计的配置刷新系统控制下，分别进行 100 次故障注入实验。最后实验发现本文所设计的内部刷新系统抗单粒子性能最好。实验数据表明本文提出的内部刷新系统容错率是传统内部刷新系统的 2.56 倍。

关键词：SRAM 型 FPGA，SEU，ICAP，纠错码，刷新

Abstract

SRAM-based FPGA has abundant resources, powerful computing performance, and it is reconfigurable. With these advantages, SRAM-based FPGA is widely used in the aerospace field. However, there are a large number of high-energy particles in the space environment. SRAM-based FPGA is very sensitive to high-energy particles, and are prone to Single Event Upset (SEU), which may cause circuit failures and cause satellites to fail to work normally. In response to this problem, domestic and foreign scholars have done widely research.

The anti-single event upset method adopted in this paper is to read back and scrub the configuration data through the ICAP interface inside the FPGA, and use the RM(2,5) code with stronger error correction ability to replace the commonly used FRAME_ECC error correction code. It supports error detection and error correction for the inversion of 3 bits, which further improves the error correction capability of the system. In addition, the circuit part where the scrubber is located is also reinforced with three-mode redundancy, which reduces the possibility of single event upset of the scrubber. Error judgment is added to the redundant module, which can identify whether a single event upset occurs in the redundant part in time, and avoid the phenomenon that multiple redundant modules may fail due to accumulation of errors. Through the improvement of the above measures, the reliability of the system is improved. At the same time, in the process of layout and wiring, the circuit to be scrubbed and the scrubber circuit are distributed in a distributed layout, which makes it more convenient to determine the frame address of the circuit to be scrubbed.

Finally, the designed system is experimented on the Artix-7 chip developed by Xilinx. The system signal is captured and observed by the Chipscope tool to verify the correctness of the system function. The system is injected with faults to simulate single event upset. Three different types of circuits to be tested are designed. One is a combinational logic circuit, the other is a sequential logic circuit, and third circuit has

both of the above circuits. Use these three circuits to simulate the user circuit under real conditions and evaluate the performance of the system. For the three kinds of circuits to be tested, 100 times fault injection experiments were carried out under the traditional internal scrubbing system, the three module redundancy reinforcement scrubbing system and the configuration scrubbing system designed in this paper. Finally, the experiment shows that the internal scrubbing system designed in this paper has the best anti single upset performance. Experimental data shows that the fault tolerance rate of the internal scrubbing system proposed in this paper is 2.56 times that of the traditional internal scrubbing system.

Key Words: SRAM-based FPGA, SEU, ICAP, ECC, Scrub

目 录

第 1 章 引言.....1
1.1 研究背景及意义.....1
1.2 国内外研究进展.....5
1.3 本文主要研究工作.....9
第 2 章 相关原理简介.....11
2.1 引言.....11
2.2 SRAM 型 FPGA 概述.....11
2.2.1 SRAM 型 FPGA 结构.....11
2.2.2 SRAM 型 FPGA 工作原理.....13
2.3 SRAM 型 FPGA 单粒子效应.....15
2.4 ICAP 接口简介.....18
2.4.1 ICAP 接口概述.....18
2.4.2 ICAP 接口信号概述.....19
2.4.3 ICAP 接口位交换原理.....20
2.4.4 ICAP 接口输入命令序列.....21
2.5 RM 纠错码原理概述.....22
2.5.1 RM 码编码原理分析.....22
2.5.2 RM 码译码原理.....25
2.6 可靠性分析.....26
2.7 本章小结.....28
第 3 章 抗 SEU 内部刷新系统设计.....29
3.1 引言.....29
3.2 基于 ICAP 的配置刷新系统设计.....29
3.2.1 配置刷新系统整体结构描述.....29
3.2.2 地址产生模块.....30
3.2.3 ICAP 接口控制模块.....33
3.2.4 RM 检错纠错模块.....42
3.2.5 刷新控制模块.....43
3.3 改进的三模冗余结构.....48
3.4 分布式布局.....49
3.5 配置刷新系统工作流程.....50
3.6 本章小结.....51
第 4 章 抗 SEU 内部刷新系统功能验证与性能评估.....53
4.1 引言.....53

4.2 系统功能验证.....53
4.2.1 ICAP 接口控制模块.....54
4.2.2 地址产生模块.....55
4.2.3 刷新控制模块.....56
4.2.4 三模冗余功能验证.....57
4.2.5 检错纠错功能验证.....58
4.3 系统性能评估.....60
4.4 本章小结.....63
第 5 章 总结与展望.....65
5.1 总结.....65
5.2 研究内容不足.....66
参考文献.....67

图目录

图 1.1 三模冗余结构.....6
图 1.2 典型的外部刷新结构.....8
图 2.1 CLB 内 Slice 的排列.....12
图 2.2 四输入 LUT 原理.....14
图 2.3 单粒子瞬态效应.....17
图 2.4 ICAPE2 模块电路图.....19
图 2.5 ICAPE2 例化模板.....19
图 2.6 ICAP 时序图.....20
图 2.7 位交换原理.....21
图 2.8 RM(2,5)码编码过程.....25
图 2.9 RM(2,5)码解码过程.....26
图 2.10 可靠性对比（未加任何缓解 SEU 措以及增加 TMR）.....27
图 2.11 可靠性对比（未加任何纠错码、 FRAME_ECC 以及 RM 码）.....28
图 3.1 系统结构.....29
图 3.2 FPGA 可编程资源和配置帧.....31
图 3.3 地址生成模块电路结构图.....32
图 3.4 帧缓冲区.....35
图 3.5 ICAP 接口控制模块电路图.....39
图 3.6 ICAP 接口控制模块状态机.....40
图 3.7 ICAP 接口控制模块算法.....42
图 3.8 RM 检错纠错模块电路结构.....42
图 3.9 带故障注入的系统结构图.....44
图 3.10 刷新控制模块状态机示意图.....45
图 3.11 故障注入流程图.....46
图 3.12 scrubber 电路结构图.....47

图 3.13 三模冗余结构.....48
图 3.14 错误判决器电路图.....49
图 3.15 布局布线（a: vivado 自动布局；b:使用 pblock 调整后）.....49
图 3.16 系统工作流程.....50
图 4.1 实验平台.....53
图 4.2 配置刷新系统电路图.....54
图 4.3 ICAP 回读配置数据波形.....54
图 4.4 ICAP 回读配置数据波形细节.....55
图 4.5 ICAP 接口写入配置命令序列波形.....55
图 4.6 地址产生模块波形.....56
图 4.7 故障注入模式波形.....56
图 4.8 故障注入模式波形细节.....57
图 4.9 1 位寄存器三模冗余波形.....57
图 4.10 多位寄存器三模冗余波形.....58
图 4.11 串口调试助手数据显示.....59
图 4.12 回读数据.....59
图 4.13 传统内部刷新系统.....60
图 4.14 最大故障注入数目统计图.....63

表目录

表 2.1 输入 LUT 真值表 14
表 2.2 空间辐射情况 15
表 2.3 单粒子效应简介 17
表 2.4 ICAPE2 端口信号描述 19
表 2.5 Type1 数据包格式 21
表 2.6 操作码格式 21
表 2.7 Type2 数据包格式 22
表 2.8 data_out 计算方式 23
表 3.1 FAR 结构 31
表 3.2 地址产生模块端口信号 32
表 3.3 配置寄存器介绍 33
表 3.4 CMD 寄存器操作命令介绍 34
表 3.5 回读配置存储器的命令序列 35
表 3.6 回写配置存储器的命令序列 37
表 3.7 ICAP 接口控制模块端口信号 39
表 3.8 RM 检错纠错模块端口信号 42
表 3.9 刷新控制模块端口信号 46
表 4.1 资源消耗统计 61
表 4.2 最大故障注入数目统计 62

第1章 引言

1.1 研究背景及意义

在现场可编程逻辑门阵列 FPGA（Field Programmable Gate Array）出现之前，电路系统通常使用的是一种专用集成电路 ASIC（Application Specific Intergrated Circuit）。ASIC 它是针对特定的电路功能进行定制的，一旦设计需求不再满足，则意味着该 ASIC 器件也将失去作用，需要重新进行定制。由此可以看出 ASIC 的通用性极差，开发成本高，并且开发周期较长。这对于需求量少的电路系统来说性价比非常低。这种现象一直持续到二十世纪七十年代，出现了世界上第一种可编程逻辑器件（Programmable Logic Device，PLD）（张荣生，2019），以此为代表开启了可编程时代。PLD 可以支持由用户自身来实现些电路设计功能，这是一种半定制可通用的电路。相较于 ASIC，PLD 以其自身的灵活性、开发周期短、设计成本低等优点，迅速占据了电子系统的很大市场。之后随着摩尔定律的发展，数字电路系统规模越来越大，内部复杂度越来越高，简单的 PLD 已经逐渐无法满足设计需求。因此后来又出现了复杂可编程器件（Complex Programmable Logic Device，CPLD）。直至二十世纪八十年代，由 Xilinx 公司发明了第一块 FPGA。CPLD 以及 FPGA 都是一种可以多次对其编程使用的半定制器件，解决了全定制电路不可更改、设计开发复杂的问题。CPLD 和 FPGA 的不同点在于，CPLD 内部更多的是组合逻辑资源，适合应用于需要组合逻辑较多、对时序要求不严格的电路中。而 FPGA 内部相对来说寄存器资源想多，更适合应用于在时序逻辑电路中。

FPGA 发展至今，其背后融合了大量微电子技术，包括微电子工艺生产、EDA 软件的开发、数字电路设计等等。而随着摩尔定律的推进，FPGA 内部资源也从最初的几十万门到现在的几百万门。这也使得 FPGA 内部资源越来越丰富，可以实现更加复杂的功能需求与运算。因此 FPGA 凭借着自身丰富的资源、可重复配置、计算能力强等优点，在航空航天、通信信息、医疗科研等领域应用越来越多。后续随着 FPGA 的不断发展，市面上出现许多中 FPGA，总体来

说可以大至可以划分为三种：SRAM 型 FPGA、反熔丝型 FPGA 以及 FLASH 型 FPGA（韩涛，2021）。

（1） SRAM 型 FPGA

相对于反熔丝型 FPGA 以及 FLASH 型 FPGA，SRAM 型 FPGA 是目前发展最快，应用最广泛的 FPGA。SRAM 型 FPGA 可以重复进行编程，并且每次配置编程速度都很快。并且随着技术的发展，Xilinx 公司提出了可以部分重配置的 SRAM 型 FPGA，这一技术使得 SRAM 型 FPGA 可以在不停止工作的情况下，完成对部分电路功能进行升级和维护，这也在一定程度上提高了 SRAM 型 FPGA 的应用范围。SRAM 型 FPGA 内部资源十分丰富，经常被用来实现计算复杂的运算。并且由于它支持部分可重构配置，使得 FPGA 可以用来进行一些并行计算。这一优势也使得 SRAM 型 FPGA 近年来开始被研究学者应用于加速卷积神经网络的运算中。但是 SRAM 型 FPGA 具有易失性，其系统掉电后配置数据不会被保留。所以如果需要再次使用，则要在使用前重新进行上电加载配置。除此外，由于器件工艺制作原因，SRAM 型 FPGA 易受到高能粒子作用，出现单粒子翻转的情况（吴楠等，2019），导致电路系统逻辑功能出现错误。鉴于此点，虽然其运算能力强大，计算速度快，但也限制了其在航天领域的发展。

（2） 反熔丝型 FPGA

反熔丝型 FPGA 的工艺原理是利用在配置时，会对配置区域部分产生大电流，通过大电流带来的热量来熔断器件内部的绝缘层，以产生永久性的电路，所以反熔丝 FPGA 只可以编程一次。相较于 SRAM 型 FPGA 以及 FLASH 型 FPGA，反熔丝 FPGA 具有非易失性，掉电后无需重新进行加载（杨海钢等，2010）。所以其不需要外部存储器来储存器配置文件，因此一定程度上减小了电路面积。除此外，反熔丝型 FPGA 因为在配置时，电路结构是靠熔断来形成的，因此具有非常高的可靠性，适用于航天领域等高辐射环境中，不存在单粒子翻转等事故。但是这也意味着，反熔丝型 FPGA 电路结构一旦配置完成，就无法进行改变。所以在配置之前，需要反复对电路功能以及配置文件进行检验，以确保在配置之后，电路功能正常，不存在任何问题，也因此反熔丝型 FPGA 所需的试错成本相对较高。由于反熔丝型 FPGA 制作工艺复杂，因此其发展进程

要落后于 SRAM 型 FPGA。

（3） FLASH 型 FPGA

FLASH 型 FPGA 顾名思义，配置数据是储存在 FLASH 存储容器中。同时它也具有 SRAM 存储设备，SRAM 主要用于控制 FPGA 的运行，而 FLASH 存储器则是用来对 SRAM 进行配置。FLASH 器件特性与 EPROM、EEPROM 类似（何宾，2011），同样具有电可擦除特性，以及非易失性。所以 FLASH 型 FPGA 与反熔丝型 FPGA 一样，掉电后数据仍然存在。重新上电后，无需重新进行配置。但与反熔丝型 FPGA 不同的是，FLASH 型 FPGA 可以进行重复多次配置。而相较于 SRAM 型 FPGA，FLASH 型 FPGA 可靠性更高，不会被反向解读。但是同样的，FLASH 型 FPGA 也是受制于制作工艺的复杂，发展程度要慢于 SRAM 型 FPGA。

目前世界上 FPGA 发展较好的龙头企业基本都是美国的公司，如 Intel 公司旗下的 Altera 公司、AMD 公司旗下的 Xilinx 公司、以及 Actel、Lattice 等。其中 Xilinx 以及 Altera 这两个公司提供了世界上绝大多数的 FPGA 生产。Xilinx 公司较为代表的产品为 SRAM 型 FPGA 的 Virtex 系列产品，该系列产品内部资源丰富，计算能力强，并且推出了相应的高抗辐射型号，在航天领域应用非常广泛。而 Actel 则是在反熔丝型 FPGA 以及 FLASH 型 FPGA 领域独树一帜，占据一席之地。今年来，随着国家科技的进步，国内也陆续出现了一些 FPGA 公司，目前技术水平相对靠前的有复旦微电子集团、国微电子等企业。但是总的来说，因受制于国外技术封锁，国内 FPGA 技术的发展要远落后于国外。

随着航天领域任务越来越复杂，电路功能越来越复杂，对性能要求越来越高，FPGA 在航天领域的应用也越来越多。比如在航天遥感上，可以通过 FPGA 控制 CCD 图像传感器进行高速数据采集。除此外 FPGA 在国内外的深空探测领域、科学卫星、商用卫星甚至军用卫星中，都有广泛的应用。美国 NASA 发射的火星探索漫游者（Mars Exploration Rover, MER）上，使用了 Actel 公司的 FPGA 来实现对照相机的控制，除此外还有无线通信中，也使用了该公司的生产的 FPGA。之后欧洲航天局发射的火星快车轨道卫星中，更是使用了接近 30 个 Actel 公司的 FPGA（宋克菲，2010）。在德国航天领域（DLR）双光谱红外

探测卫星（BIRO）中的有效载荷数据处理部分、接口、红外照相机传感器的控制等，都用到了 Actel FPGA。与 Actel FPGA 相比，虽然 SRAM 型 FPGA 在航天领域应用的较晚，但是其优异的性能和可重配置特性，使其在航天上的应用迅速发展。澳大利亚在 2002 年发射了一颗型号为 Fedsat 的科学卫星，该科学卫星主要作用是用来研究地球的磁层分布。这颗卫星中有效载荷就运用了 SRAM 型 FPGA 来提高运算能力。在 2003 年发射了型号为 Optus CL 的卫星，主要目的为与地面进行通信。它的有效载荷中使用了 SRAM 型 FPGA 来处理来自地球的数据信号。NASA 发射的各类火星探测器中都能看到 SRAM 型 FPGA 的身影，主要运用在相机控制，各类电机控制、机械臂控制等。

随着航天任务需求的不断提高，航天器内电子系统承担的任务逐渐增加，功能也越来越多，而为了控制电子系统的所占空间，因此导致内部电路密度越来越大。这也意味着电子系统受到高能粒子辐射发生故障的概率逐步增大。1982 年，日本航天工作者选取了 350 颗卫星进行研究，最后发现只有 83 颗没有出现过故障，其余卫星在运行时都或多或少出现过问题（卜雷雷，2010），由此可见出现故障的卫星占比达到 76%，而其中出现故障的原因大多是由高能粒子辐射所导致（Li J 等，2017），这也证明了太空中高能粒子辐射对航天器的寿命健康有着很大的影响。1997 年太阳活动剧烈，释放了很多宇宙射线，致使美国卫星出现大批故障（雷利华等，2005）。法国发射的型号为 SPOT-1 的卫星，在其在轨运行 30 年间，由高能粒子辐射所导致的故障平均每年至少出现 3 次，而这种故障每次都会导致卫星无法正常工作，短则 1 天，长则 3 天都使卫星处于失效状态（李昕，2017）。美国也曾对卫星故障进行过统计，在其曾经发射过的卫星中选取了 39 颗作为统计目标。最后统计结果发现，这 39 颗卫星在轨运行的 10 年间，共出现 1589 次故障（赵磊等，2013）。在 1589 次故障中，由高能粒子辐射导致的单粒子翻转所引发的故障共有 621 次，占比大概在 39%左右。无论是从各个国家卫星故障统计结果来看或者卫星在轨运行情况，都足以说明高能粒子辐射给航天器电子系统健康运行带来极大威胁。而 SRAM 型 FPGA 由于其工艺制作原因，高能粒子会穿过介质层进入器件内部，产生电离等一系列动作，从而影响到器件内部电荷分布，从而极易发生单粒子翻转而导致电路功

能异常。若不能很好的降低单粒子翻转对 SRAM 型 FPGA 所带来的影响，将会极大地限制 SRAM 型 FPGA 在航天领域的应用。所以研究如何对 SRAM 型 FPGA 进行抗单粒子加固是极其重要的。

1.2 国内外研究进展

为了解决 SRAM 型 FPGA 单粒子翻转问题，国内外学者都给出了很多解决方案。总的来说可以将这些解决方案分为三种，第一是在制造工艺上面增加抗辐射处理，第二是通过冗余技术，第三是通过 SRAM 型 FPGA 可重构特性，对 FPGA 进行配置刷新（赵磊等，2013）。

（1） 制造工艺

最常见的抗单粒子翻转措施是在生产制造 FPGA 过程中，对其进行抗辐射加固处理。通常是选择抗辐射能力强的材料，比如石墨多氰酸酯。或者是对器件结构进行改进，比如采用更为先进的绝缘体上硅（Silicon on Insulator, SOI）工艺。由于 SOI 器件中 Si 膜厚度小于体 Si 器件，并且隐埋的氧化层使得衬底区产生的电子不能被收集，仅仅有顶层 Si 膜内的电子才能被收集，所以对高能粒子敏感的区域相比较于体 Si 器件小得多，从而可以有效提升器件的抗辐射能力（李亚军，2021）。另外，由于体区硅 MOSFET 与衬底完全隔离开，不存在传统硅衬底 CMOS 引入的 PNPN 通路，根本上解决了 CMOS 电路的闩锁效应，提升了工作电路的抗辐射能力（L. Palkuti 等，2014）。相较于传统的体硅工艺，SOI 表现出了非常优越的抗辐射能力，其抗瞬时剂量率要比体硅工艺高的多。由 Honeywell 公司提出的基于 SOI 工艺的 MOS 器件，在提高了抗单粒子效应同时，也改善了器件的总剂量效应。国外 FPGA 龙头企业 Xilinx 公司发布的宇航级 Virtex4Q（Xilinx，2014）、Virtex5Q（Xilinx，2018）系列芯片中，这两款芯片在抗辐射的性能方面非常优越。它们的存储单元通过设计进行辐射加固处理。主要的设计原理是通过对单个存储单元增加更多的晶体管，这样从概率层面上，便降低了离子在某个存储单元内多次撞击到同一个晶体管的可能性。该设计称为 RHBD（Radiation-hardened by Design）设计（薛晓良，2019）（Balasubramanian 等，2006）（Mohr K C 等，2007）。在实际应用中，该系列芯

片表现出非常强的抗辐射能力（厉明坤，2017）。但这也提高了该芯片的价格，并且受制于技术封锁，此类型的芯片目前并不对我国出售。总的来讲，在制造工艺上进行抗辐射加固处理，仍是最主要、最有效、最直接的抗单粒子措施，也是现在国内外主流的选择与研究方向。

（2） 冗余技术

冗余技术是对原始用户电路进行备份，因为多个备份同时发生单粒子翻转的概率较低，所以通过此手段可以提高系统的容错率。较为简单的有双备份比较技术（Duplication with comparison, DWC）（Bolchini C 等，2007），该方法是对用户电路进行一次复制，同时对这两份电路施加相同输入，并在输出端设置比较器，若输出结果一致，则说明电路不存在问题，若输出结果不一致，则说明电路出现错误。除了 DWC 外，还有三模冗余（Triple Modular Redundancy, TMR）技术，顾名思义该方法是将同一逻辑模块复制成三份，并对这三份模块保持相同的输入，再将三个模块的输出结果进行多数表决（周宇澄，2017），表决后的输出为该模块的最终输出结果，如图 1.1 中显示了一种最简单的 TMR 结构。Xilinx 公司设计开发了 TMRtool（Xilinx，2017），可以通过该工具自动为所需加固的电路插入 TMR 结构。之后于 2017 年，在 TMRtool 基础上，发布了更为先进的 XTMR 工具。Xilinx 公司注意到三模冗余结构中，多数表决器部分仍存在单粒子翻转的可能性。而若多数表决器出现单粒子翻转，那么会直接影响到三模冗余的输出。因此在 XTMR 中对表决器也用到同三模冗余相同的保护原理，即将多数表决器也复制为三份，降低了多数表决器出现故障的概率。并增加少数表决器，意在当多数表决器出现了故障时，少数表决器会输出高阻态来对其进行关闭。

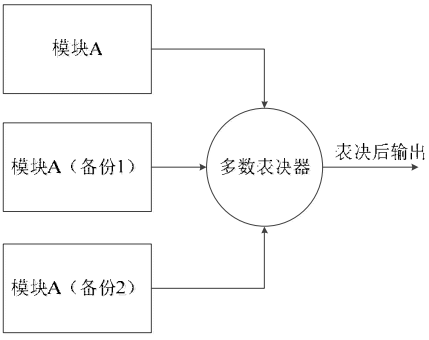


图 1.1 三模冗余结构

Figure 1.1 Three-mode redundant structure

除此外，研究学者还在传统的 TMR 结构基础上进行了很多优化和改进。比如 Rahul Parhi 等人在 2015 年提出的局部三模冗余技术（Partial Triple Modular Redundancy, PTMR）（Rahul Parhi 等，2015）。该方法原理是在整个系统设计中，不同比特位对系统的重要性不同，比特位越高，对系统的影响越大，相应的权重就会越高，反之比特位较低的，权重就会相对较低，之后对权重相对较高的部分增加三模冗余。最后通过对行波进位加法器实验，可以发现系统面积开销降低 75%左右，同时也减少了系统产生的功耗。

Giane Ulloa 等人提出了 DTMR（Diverse Triple Modular Redundancy, DTMR）技术（Gianc Ulloa 等，2017）。该方法不同于传统 TMR 的地方在于，TMR 通过对模块进行复制来达到冗余效果，模块的冗余采用的是同一方式，而 DTMR 则是采用了不同的方式来对模块进行冗余操作。并且该研究团队最后还通过实验证明了，在 DTMR 对 SET 的屏蔽效果与 TMR 一致的情况下，DTMR 所花费的面积以及系统产生的功耗都比 TMR 少。

采用冗余技术虽然可以提高电路系统的容错率，但是由于对电路备份，导致资源消耗量增加，并且冗余技术存在的最大问题是，自身不存在修复能力，所以会使单粒子翻转造成的错误逐渐累积。当累积到多个冗余块都存在故障时，此时经过表决器输出的结果就可能会出错。故而目前一般不会单独使用三模冗余技术，通常会与配置刷新同时使用，增加修复功能，避免错误累积。

（3） 配置刷新

随着 FPGA 动态重构技术的发展与成熟，越来越多的航天项目中，采用配置刷新的方式来提高系统的抗单粒子翻转能力。配置刷新主要原理是对 FPGA 的配置数据通过接口进行回读与重新写入。通过配置刷新对配置数据进行重写，覆盖原始数据，以此来纠正单粒子翻转导致的错误。最初的配置刷新系统是通过联合测试工作组接口（Joint Test Action Group, JTAG）或 SelectMAP 接口这两个外部接口来实现的。JTAG 作为 FPGA 最常用的接口，其主要是用来与上位机之间进行配置信息的读写。在 FPGA 加载上电初期，往往会使用 JTAG 接口来将比特流文件从上位机中下载至 FPGA 内。而由于 JTAG 是一种串行接口，所

以其数据传输速度受到限制（田原，2019）。SelectMAP 作为另一种外部接口，它可以支持的位宽有 8 bit、16 bit 以及 32 bit（Xilinx，2014），所以传输速度相较于 JTAG 要快，也是目前常用的外部刷新接口。外部刷新，顾名思义，其刷新控制器（scrubber）是放在芯片外部运行。通常情况下会将刷新控制器配置在抗辐射能力较强的芯片中，如反熔丝型的 FPGA，并将原始比特流文件存储在外部 flash 或 prom 中（U. Legat 等，2012）（庞波等，2017），如图 1.2 所示。最简单的外部刷新方式是对系统进行定时刷新（Xilinx，2000），即以特定的频率，将原始比特流文件重新写入 FPGA 中，这种方法虽然简单，但是效率很低，每完成一次刷新都需要使用很长的时间，并且也需要很大的内存来存放原始比特流文件。因此又有学者提出采用回读检测刷新的方式（M. Kumar 等，2017），该方法首先对原始配置数据进行 ECC 编码（Xilinx，2000），并对编码后的结果进行存储，之后在系统运行时对其中的配置数据进行回读，并将回读到的数据 ECC 编码后与原先存储的数据进行比较，若一致则说明没有出现故障，若不一致则说明存在故障，并且可以根据编码结果对故障位置进行定位，之后再对该故障位置进行重新写入（P. K. Samudrala 等，2004）。经过 ECC 编码后，可以大大降低所需存储的数据，且刷新时不会造成系统功能中断。但是这也增加了系统设计的复杂度。

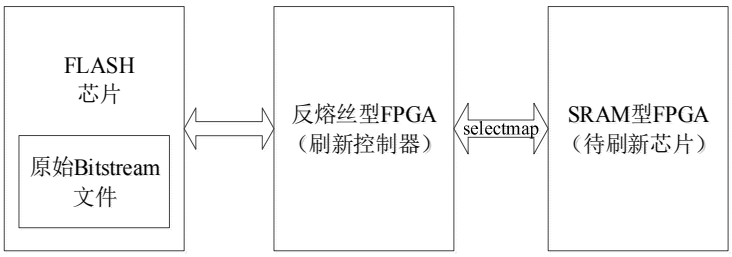


图 1.2 典型的外部刷新结构

Figure 1.2 Typical external scrubbing structure

之后 Xilinx 公司在 FPGA 中又增加了内部配置访问接口（Internal Configuration Access Port, ICAP）（王喆等，2020），它是 FPGA 的内部接口，所以刷新器 scrubber 存放在 FPGA 中，不需要额外的外部存储器来存放，这被称之为内部刷新。内部刷新相较于外部刷新而言，最大的优点就在于节省电子系

统的面积。典型的内部刷新方式是通过 ICAP 接口对 FPGA 配置数据进行回读，之后经过 ECC 校验，判断是否出错，出错后进行重写。Mahsa Mousavi 等人在传统内部刷新基础上，提出了一种分区刷新的方式（Zheng. S 等，2019）。该方法通过判断配置存储器在 FPGA 内部的重要程度，并结合智能算法，来对配置区域进行最优划分，以此来降低刷新系统的平均修复时间（Mean Time To Repair, MTTR）。文献（斯涛，2019）中提出采用多频度刷新的方式，对不同的配置帧采用不同的刷新频度，以此来提高刷新系统的针对性。

对比内外刷新两种刷新方式，外部刷新因为刷新控制器放在抗辐射器件中运行，所以外部刷新系统的可靠性相较于内部刷新来说要更高。而内部刷新刷新速度却比外部刷新要快。因此也有学者试图将二者同时使用，以发挥二者的优点，如文献（A. Stoddard，等 2017）中便是将内部刷新与外部刷新方法相结合，既保留了外部刷新的高抗辐照性，又利用内部刷新提高了刷新速度，但该方法设计十分复杂，不易实施。

1.3 本文主要研究工作

如前文所述，SRAM 型 FPGA 内部资源丰富，计算能力强，在航天领域被广泛使用。但是太空环境中存在着大量高能粒子，会导致 SRAM 型 FPGA 被辐射，从而导致器件出现单粒子翻转等故障（徐文波等，2012）。针对这一现象，国内外学者都提出了很多解决方案，目前常见的方案有三模冗余和配置刷新两种。正如 1.2 章节中的分析，三模冗余无法对错误进行修复，会导致错误累积，（张文龙，2014）并且消耗大量的面积资源。外部刷新占用电子系统面积，内部刷新刷新器部分对高能粒子仍然敏感。鉴于此，本文提出了一种将三模冗余与内部刷新相结合的方案。对内部刷新刷新控制器部分采用三模冗余加固，提高刷新器抗辐射能力，以此来弥补内部刷新的缺点。除此外本文还使用了纠错能力更强的 RM 纠错码来替代内部刷新系统常用的 FRAME_ECC 纠错码，将系统的纠错能力从原来的纠 1 查 2，提高到了纠 3 查 4。最后还对三模冗余进行了改进，增加了错误判断器，一旦三模冗余内其中一部分出现故障，便可及时给出信号，促使刷新器可以及时对三模冗余进行刷新，避免出现错误累积的情况。

使系统更具有可靠性。

本文中第一章节主要介绍了本课题研究的背景和意义，介绍了 FPGA 的发展历史以及主要的 FPGA 类型。重点介绍了 SRAM 型 FPGA 在航天领域的应用以及面临的单粒子翻转问题。之后详细讨论了目前国内外针对 SRAM 型 FPGA 抗单粒子翻转问题的研究情况，分析了各类解决方案目前存在的优缺点。最后针对所提出的优缺点，简单介绍了本课题所设计的内部刷新系统。

第二章节首先介绍了关于 SRAM 型 FPGA 的基本情况，包括它的内部结构以及它的工作原理。之后介绍了 SRAM 型 FPGA 在太空环境下所面临的单粒子效应情况以及 RM 纠错码的编码译码原理。还介绍了 ICAP 接口的基本原理。最后分析了 SRAM 型 FPGA 对于抗单粒子翻转自身的可靠性，以及经过抗单粒子翻转加固后所对应的可靠性情况，并通过对比分析，可以得出经过抗单粒子翻转加固后的 SRAM 型 FPGA 系统可靠性大大增强。

第三章详细介绍了本文所设计的 SRAM 型 FPGA 抗单粒子翻转的内部刷新系统。分别介绍了系统的构成、各个模块是如何设计的以及系统的工作流程。除此外，还介绍了本文所提出的改进的三模冗余设计结构。该三模冗余可以对冗余模块错误进行判断，以此来避免冗余模块错误的累积。最后还提出了对系统刷新器以及待测电路部分进行分布式布局的设计。

第四章主要介绍了对系统采取的相关实验。首先对系统功能进行了验证，然后通过故障注入的方式来模拟单粒子翻转，对系统性能进行评估。

最后第五章节对论文工作进行总结，并对系统的不足之处进行分析，为未来系统的改进提供了方向。

第2章 相关原理简介

2.1 引言

本文就 SRAM 型 FPGA 抗单粒子翻转情况进行研究，因此本章节将会介绍 SRAM 型 FPGA 的结构以及其工作原理。不仅如此还需要对 SRAM 型 FPGA 产生单粒子翻转的原因进行了解，所以本章节也会介绍到空间环境中存在的辐射问题以及这些辐射对 SRAM 型 FPGA 产生的影响，重点将介绍 SRAM 型 FPGA 中的单粒子现象。除此外为了进一步提高所设计的系统抗单粒子性能，提出了使用纠错能力更强的 RM 纠错码，在此也会介绍该 RM 纠错码的编译码原理。最后对所设计的系统在理论层面上进行了可靠性分析。

2.2 SRAM 型 FPGA 概述

2.2.1 SRAM 型 FPGA 结构

SRAM 型 FPGA 最具代表的是由 Xilinx 公司设计生产的，典型的构造是基于查找表的形式，其结构主要包含六个部分，可编程输入输出单元（Input Output Block, IOB）、基本可编程逻辑单元（Configurable Logic Block, CLB）、块随机存取存储器（Block Random Access Memory, BRAM）、丰富的布线资源和嵌入式功能块（徐文波等，2012）（冯兴，2016）。

（1） 可编程输入输出单元

FPGA 通过可编程输入输出单元与外界进行数据交换。不同型号的 FPGA 可以使用的输入输出单元数量不同。用户在对 FPGA 开发过程中，可以通过 FPGA 开发工具，将可编程输入输出单元根据需求进行配置。首先可以配置的是其方向，也就是可以灵活将其配置为输入端口或者输出端口。除此外，还可以将可编程输入输出单元与不同的电气标准进行匹配。常见的电气标准由 LVCMOS、LVDS、SSTL 等。用户可以自行对其特性进行配置。可编程的输入输出单元使得 FPGA 的应用更加灵活。

（2） 基本可编程逻辑单元

CLB 主要由查找表（Look Up Table, LUT）和寄存器组成。查找表是主要用于实现组合逻辑功能，寄存器主要用于实现时序逻辑相关电路。每一个 CLB 都被连接到一个开关矩阵，主要用于访问通用路由矩阵，如图 2.1 所示。CLB 由 Slice 构成，Slice 中包含多输入的查找表、触发器、多路选择器以及进位链（邓双敏，2021）。每个 CLB 包含的 Slice 数量也不同。不同类型的 FPGA 中，每个 CLB 所包含的 Slice 数量是不同的。以 Aritex7 系列芯片 xc7c100tfgg484-2 为例，CLB 中包含 15850 个 Slice。资源丰富的 FPGA，CLB 的数量以及 CLB 内部包含的 Slice 数量越多，以此来保证 FPGA 内部可供配置的资源。

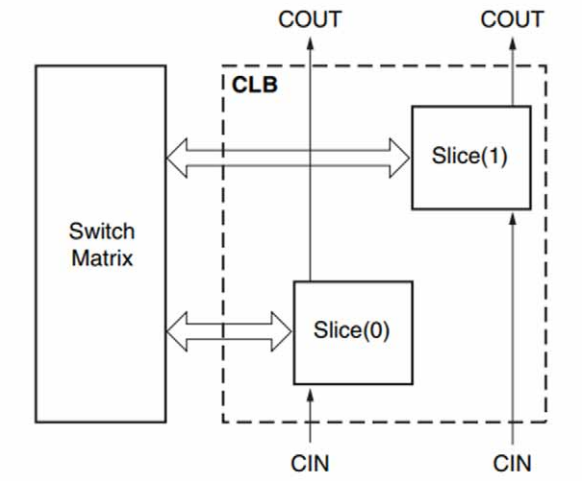


图 2.1 CLB 内 Slice 的排列

Figure 2.1 Arrangement of Slices within the CLB

（3） 块随机存取存储器

目前常见的 FPGA 内部都包含许多块随机存储器 BRAM，它可以灵活的被配置成所需要的功能，比如单端口 RAM、ROM、双端口 RAM、ROM、同步 FIFO 或异步 FIFO 等存储逻辑结构。这极大的提高了 FPGA 内部的灵活性，可以降低内部资源的消耗。

（4） 嵌入式功能块

为了更加方便对电路进行设计，FPGA 在开发过程中被嵌入了很多较为成熟且通用的逻辑功能块。比如锁相环 PLL 用于实现对时钟的分频操作。常见的还有 DSP、CPU 等，帮助 FPGA 实现更高级的操作。

（5） 布线资源

FPGA 内部的布线资源主要是用来连接各个逻辑单元。FPGA 内部的布线资源非常丰富，这些布线资源可以根据布线长度、布线宽度、布线位置区域进行区别。长线布线资源主要用于完成不同 bank 之间的连接。短线布线资源是用于完成逻辑电路内部信号的连接。还有全局布线资源，顾名思义，其主要用于连接全局性的信号，比如时钟和复位信号（舒德刚，2019）。由此可以看出不同类型的布线资源发挥着不同的作用，以使 FPGA 可以应对不同的应用场景。FPGA 内部的布线资源越丰富，该 FPGA 的性能越高。随着 FPGA 开发流程的不断完善，用户只需要通过 FPGA 对应的开发工具即可自动完成对 FPGA 的布局布线操作，无需手动进行布局布线，不需要了解布线具体操作流程，大大减少了用户的工作量。

2.2.2 SRAM 型 FPGA 工作原理

SRAM 型 FPGA 作为一种可以多次进行重复配置的可编程器件，其常见的实现方式是基于查找表的形式。在实际设计实现中，用户通过使用 verilog 或者 vhdl 等硬件描述语言，对电路行为级进行建模，设计实现出所需的逻辑功能。设计完成后还可通过 modelsim 等 EDA 工具进行仿真。仿真不存在问题后再通过 FPGA 对应的开发工具，进行逻辑综合，布局布线，最终生成比特流文件，最后将生成的比特流文件配置到 FPGA 内，即完成了一次对 FPGA 的配置（陆启帅等，2014）。而 FPGA 内部在接收到比特流文件后，基于查找表内容，得到最终的输出结果，以此来实现逻辑功能。

在数字电路中，对于逻辑门电路，若其有 n 个输入端口，那么可以得知该逻辑门电路将会有 2n 种输出结果，而可以确定的是每一种输入组合情况，都将有且仅有唯一一个对应都输出结果。根据这一结论，那么就可以将一个已确定好输入端口数量的逻辑门电路，其每一种输入以及所对应的输出组合起来，存储至存储中。那么之后的所有关于该逻辑门的运算，都可以转变为根据逻辑门的输入信息，在存储器中查找对应的输出结果，以此便可以查找的方式来替代逻辑门的计算。这一原理便是 SRAM 型 FPGA 的基于查找表的实现方式。查找表实际上相当于 RAM 存储器，它把计算好的数据实现存入 RAM 存储器中，之

后再输入的数据相当于输入地址，只需要根据地址进行查表，再将对应的结果进行输出。常见的 FPGA 是基于 4 输入的查找表，其原理如图 2.2 所示，而 4 输入查找表的真值表如表 2.1 所示。从中可以看出，4 输入的查找表实际上就是另一种形式的 RAM 存储器，并且该存储器具有 4 位地址位，1 位数据位，它可以存储 16 位数据。

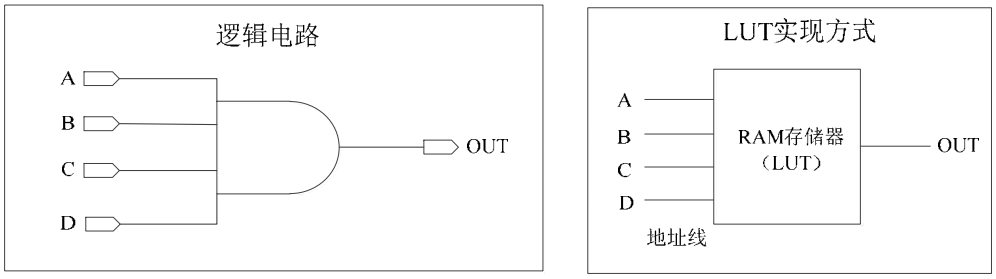


图 2.2 四输入 LUT 原理

Figure 2.2 4-Input LUT principle

表 2.1 输入 LUT 真值表

Table 2.1 4-Input LUT truth table

逻辑电路 ABCD	OUT	RAM 地址	查找结果
0000	0	0000	0
0001	0	0001	0
0010	0	0010	0
0011	0	0011	0
0100	0	0100	0
0101	0	0101	0
0110	0	0110	0
0111	0	0111	0
1000	0	1000	0
1001	0	1001	0
1010	0	1010	0
1011	0	1011	0
1100	0	1100	0
1101	0	1101	0
1110	0	1110	0
1111	1	1111	1

2.3 SRAM 型 FPGA 单粒子效应

FPGA 内部的布线资源主要是用来连接各个逻辑单元。空间环境中存在着大量的高能粒子，主要来源有地球周围的内外辐射带、太阳耀斑以及宇宙中的射线等。地球的内外辐射带主要是由于地球的磁场作用，在近地空间处俘获到大量高能质子、高能电子以及少量的重离子。内辐射带的高度大约在地球上空 1.5~2 个地球半径，外辐射带是在 4~5 个地球半径的距离。太阳耀斑会带来很多高能粒子，其中绝大多数为质子。表 2.2 列出了目前主要的空间辐射情况。

表 2.2 空间辐射情况

Table 2.2 Space radiation

类别	地球内外辐射带		太阳耀斑		宇宙射线	
	质子	电子	大事件	小事件	太阳活动高年	太阳活动低年
最大剂量 $/rad \cdot h^{-1}$	数十	几十	数十	约10^{-1}	约5×10^{-4}	约 15×10^{-4}
平均剂量率	数百 rad/d	数十 rad/d	（10^{-2}~10^{-3}） rad/事件	（1~10） rad/事件	约5×10^{-4}	约 15×10^{-4}
年剂量率	约10^5	约10^4	数千		5 左右	15 左右
变化情况	（0~10^6）rad/A		约（10^{-3}~10^{-4}）rad/A		恒定	
存在的空间环境	整个宇宙空间		赤道面几千 km 外，磁纬$> 50^{\circ}$		赤道面 1000~6000km 之间，磁纬$< 60^{\circ}$	

从上表中可以看出，空间环境中存在着大量的能量粒子，这些能量粒子会对电子器件带来威胁。高能粒子射线与物质之间产生相互作用后，会发生原子移位损伤和电离损伤。当集成电路器件暴露于辐射环境后，会导致器件出现数据扰动、器件参数漂移、甚至失效等情况。高能粒子辐射对器件带来的影响最常见的有两种，一是总剂量效应（Total Ionizing Dose，TID），二是单粒子效应（Single Event Effect，SEE）或称为单粒子现象（Single Event Phenomena，SEP）（邹三泳，2017）。当高能量的粒子入射至集成电路上的器件时，由于能量大于半导体材料的禁带宽度，会导致材料内部产生大量的电子空穴对，激发出的电子空穴对会随即发生复合，扩散和漂移。当产生的电子空穴对扩散或者漂移至

材料界面处时，由于界面处本身缺陷的俘获会导致界面陷阱电荷形成；当扩散至氧化层中，会形成氧化物陷阱电荷；陷阱电荷的存在会影响器件的阈值电压，造成集成电路的不稳定性，而当注射的高能粒子剂量超过阈值剂量时，会使得器件永久失效，电路功能异常。这种效应被称为总剂量效应（任小西，2007）。当空间中的高能粒子入射到器件内部时，由于高能量粒子引发的电离效应会使得材料内部产生电子空穴对，进而在注入路径上形成带点通道，留下了多余的电荷。由于多余电子以及空穴的产生，PN 结处的电场会被调制进而使得引入的多余电荷移动到器件某处，当该处的电荷超过临界电荷（Critical Charge，Q_c）时，会导致 PN 结电场完全扭曲，使得开关行为发生变化，不受控制，从而因此整个电路层面的逻辑行为发生改变，这种变化成为单粒子效应。本文重点讨论单粒子效应对 SRAM 型 FPGA 造成的影响，而单粒子效应也分为很多种情况，以下挑选出几种典型单粒子效应进行介绍。

（1）单粒子翻转（Single Event Upset, SEU）

单粒子翻转指的是当高能粒子穿过 FPGA 器件时，会发生电离，产生大量电荷（赵培雄，2020），如果电荷超过临界值，会造成该处的逻辑值发生从 0-1 或 1-0 的翻转。如果该翻转未被时序单元采集，则不会对电路造成影响，并且会在一段时间后自行恢复，但如果该翻转被采集到，那么至少会在时序单元中经历一个时钟周期，这会对电路功能造成严重影响。单粒子翻转只是对电路逻辑状态造成影响，并未对器件本身造成损伤，即这是一种软错误，所以可以通过对器件进行初始化配置来修复这种错误。

（2）单粒子瞬态效应（Single Event Transient, SET）

SET 也是一种软错误，它与 SEU 不同的是，SEU 主要发生在存储单元中，而 SET 多发生在模拟电路或组合电路中。SET 也是由于高能粒子入射到器件中后，发生电离会产生电子空穴对，致使该处的电流增加，并产生瞬态电流脉冲，就产生了单粒子瞬态效应，会在该处组合逻辑短暂的出现 0-1 翻转。单粒子瞬态效应表现形式通常为我们在电路中常见到的毛刺现象，多数情况下它可以自行回到正常状态。但如果电流脉冲传播到存储单元中，并且满足存储单元对建立时间、保持时间的要求，那么此时这个瞬时电流脉冲便会被存储单元锁住，

就变成了单粒子翻转现象。其示意图如图 2.3 所示。

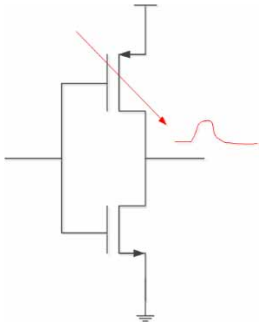


图 2.3 单粒子瞬态效应

Figure 2.3 Single Event Transient

（3）单粒子闩锁效应（Single Event Latchup, SEL）

闩锁效应主要存在于 CMOS 器件中，因为 CMOS 器件中会产生 PNPN 或 NPNP 结构，这种结构常常会寄生可控硅（石轩，2009）。通常情况下可控硅处于高阻态，VDD 和 VSS 之间为断开状态，但如果受到空间高能粒子入射，可能会使二者导通，发生短路，在一瞬间产生很大的电流，会将电路击穿。所以单粒子闩锁效应是一种硬错误，其对器件带来的损伤是不可恢复的（陆禹帆，2019）。

除了上述提出的三种常见的由高能粒子入射导致的器件错误外，科学家还发现了其他几种故障，总结之后，如表 2.3 所示（张曦文，2020）（李超，2017）（张越，2020）。

表 2.3 单粒子效应简介

Table 2.3 The introduce of the Single Event Effect

类型	描述	硬/软错误
单粒子翻转（Single Event Upset, SEU）	器件逻辑功能发生 0-1 翻转	软错误
单粒子瞬态（Single Event Transient, SET）	组合逻辑电路由于瞬时电流的冲击而导致的逻辑翻转	软错误
单粒子多位翻转（Multiple Bit Upset, MBU）	连续存储单元多位逻辑出现 0-1 翻转	软错误

单粒子功能中断（Single Event Functional Interrupt, SEFI）	一个单粒子导致控制部件故障	软错误
单粒子栅穿（Single Event Gate Rupture, SEGR）	大电流流过器件内栅介质，从而导致器件被击穿	硬错误
单粒子闩锁（Single Event Latchup, SEL）	CMOS 器件中寄生效应处发生短路，产生电流	硬错误
单粒子烧毁（Single Event Burnout, SEB）	电流过大，导致器件被烧毁	硬错误

2.4 ICAP 接口简介

2.4.1 ICAP 接口概述

ICAP 接口是 Xilinx 提供的一种内部接口，通过该接口可以对 FPGA 的配置存储单元内的数据进行回读。针对回读操作，包括两种，其一为读回验证，它可读取包含用户存储单元当前值的所有配置存储单元。还有一种回读形式为读回捕获，不仅可以读取配置数据，还可以读取所有输入输出单元以及基本可编程单元内寄存器的值，这点非常有利于在设计过程中进行调试。

在读取配置数据前，首先要向设备发送配置命令。这是为了启动 FPGA 对配置存储单元数据的读取。在发送之后，用户设备就会将配置数据转移至 ICAP 接口并被读出。用户设备可以选择的有 CPLD、微处理器或 FPGA。根据本课题研究的背景以及目的，在此选择的用户设备为 FPGA，并对其内部配置存储进行读取。

使用 ICAP 接口时，需要例化 Xilinx 提供的 ICAP 接口原语，具体操作如同调用其他子模块。本文采用的芯片类型是 xc7c100tfgg484-2，使用的是 ICAPE2 模板，而 ICAPE2 模块电路图如图 2.4 所示。将 ICAPE2 例化后，代码模板如图 2.5 所示。

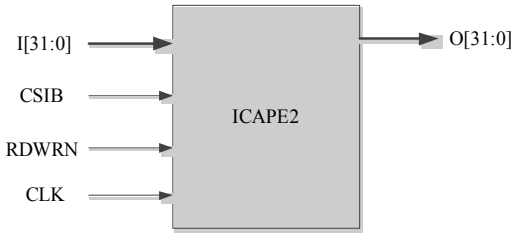


图 2.4 ICAPE2 模块电路图

Figure 2.4 Circuit diagram of icape2 module

```
// ICAPE2: Internal Configuration Access Port
//         Artix-7
// Xilinx HDL Language Template, version 2017.4

ICAPE2 #(
   .DEVICE_ID(0'h3651093),     // Specifies the pre-programmed Device ID value to be used for simulation
                               // purposes.
   .ICAP_WIDTH("X32"),         // Specifies the input and output data width.
   .SIM_CFG_FILE_NAME("None")  // Specifies the Raw Bitstream (RBT) file to be parsed by the simulation
                               // model.
)
ICAPE2_inst (
   .O(O),         // 32-bit output: Configuration data output bus
   .CLK(CLK),     // 1-bit input: Clock Input
   .CSIB(CSIB),   // 1-bit input: Active-Low ICAP Enable
   .I(I),         // 32-bit input: Configuration data input bus
   .RDWRB(RDWRB)  // 1-bit input: Read/Write Select input
);

// End of ICAPE2_inst instantiation
```

图 2.5 ICAPE2 例化模板

Figure 2.5 ICAPE2 instantiation template

2.4.2 ICAP 接口信号概述

ICAP 接口模块端口信号描述如表 2.4 所示（Xilinx，2018）。ICAP 的 I/O 端口位宽支持 8bit、16bit 以及 32bit，本文根据系统整体需求，采用的是 32bit 位宽。CSIB 信号控制 ICAP 的使能，为低时 ICAP 才可以进行数据读写操作。RDWRB 控制着读写模式，该信号高电平时 ICAP 接口进行读操作，为低电平时进行写操作。

表 2.4 ICAPE2 端口信号描述

Table 2.4 ICAPE2 port signal description

信号名	方向	位宽/bit	描述
CLK	输入	1	时钟

CSIB	输入	1	ICAP 使能信号，低电平有效
RDWRB	输入	1	读/写控制
I	输入	8/16/32	配置数据输入
O	输出	8/16/32	配置数据输出

ICAP 接口时序图如图 2.6 所示（Xilinx，2018），在读写配置数据过程中，必须严格遵循图 2.6 所示的时序关系。在读取配置内存时，因要首先需要向 ICAP 发送配置命令，所以需要先设置为写模式，即将 RDWRB 信号拉低。在发送完配置命令以及待回读的存储器帧地址后，将 RDWRB 拉高切换为读模式，开始读取配置数据。值得注意的是，在写完一次数据并切换到读模式后，从第 4 个周期开始，才会接收到回读的数据（王佳丽，2018）。RDWRB 在每次进行读写模式切换时，都需要保证 CSIB 信号维持在高电平。

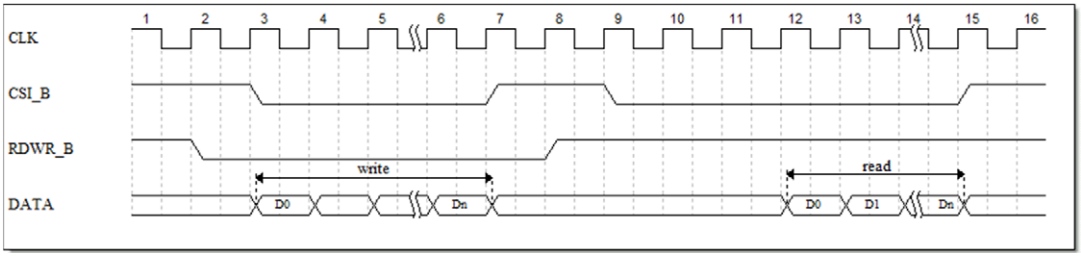


图 2.6 ICAP 时序图

Figure 2.6 ICAP Timing Diagram

2.4.3 ICAP 接口位交换原理

Xilinx 公司规定了在向 ICAP 数据接口发送读写命令或者配置数据时，或者读取到 ICAP 接口发送出来的数据，都需要再经过比特位流的转换，具体转换关系如图 2.7 所示。首先将数据从高位到低位每 8bit 划分为一组，ICAP 数据接口为 32bit，那么就可以划分为 4 组，假设每一组从高位到低位分别用从 D7 到 D0 表示，那么在位交换后，每组中的 D7 变成 D0，D0 变为 D7，D6 变成 D1，D5 变成 D2，依次类推，举例说明，交换前数据为 32`H89ABCDEF，那么在位交换后，就会变成 32`H91D5B3F7。

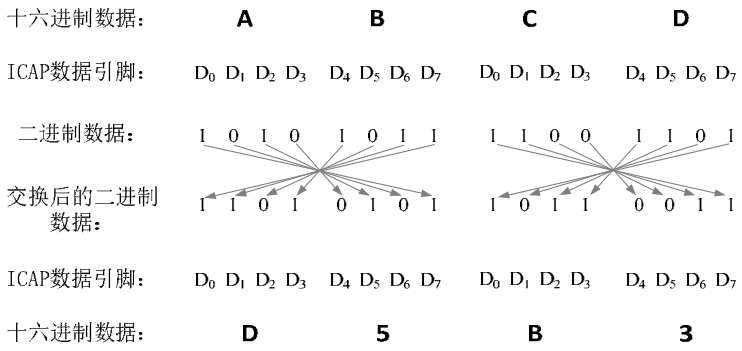


图 2.7 位交换原理

Figure 2.7 Bitswap principle

2.4.4 ICAP 接口输入命令序列

ICAP 接口在对 FPGA 配置存储器进行读写时，发送的命令序列都由两种数据包类型组成，分别命为 Type1 和 Type2，都为 32 位的数据包（Graham P，2016）。Type1 类型数据包主要用于对寄存器进行读写。数据包格式如表 2.5 所示，其中“Header Type”也就是高 3 位用来说明该数据包的数据类型，3’b001 为 Type1 类型；“Opcode”为操作码，可以通过它来切换读取写入操作，具体代表的含义如表 2.6 所示；“Register address”代表寄存器地址，其中 N 表示该位没有被使用，无实际意义，低五位表示待进行读写操作的寄存器；“Reserved”如前所述，为保留位，未被使用；“Word counter”为需要进行读写操作的数据总量。

表 2.5 Type1 数据包格式

Table 2.5 Type1 packet format

Header Type	Opcode	Register address	Reserved	Word count
[31:29]	[28:27]	[26:13]	[12:11]	[10:0]
001	xx	NNNNNNNNNxxxxx	NN	xxxxxxxxxxx

表 2.6 操作码格式

Table 2.6 Opcode format

OPCODE	Function
00	NOOP
01	Read

10	Write
11	Reserved

在 Type1 类型数据包之后，紧接着就为 Type2 类型数据包。Type2 类型用于写入长数据，所写入的地址，即为 Type1 中所提供的。Type2 类型的组成如表 2.7 所示，从中可以看出“Header Type”中 010 即表示该数据包为 Type2 类型，在“Header Type”以及“Opcode”之后，即为数据段。

表 2.7 Type2 数据包格式

Table 2.7 Type2 packet format

Header Type	Opcode	Word Count
[31:29]	[28:27]	[26:0]
010	xx	xxxxxxxxxxxxxxxxxxxxxxxxxxx

2.5 RM 纠错码原理概述

2.5.1 RM 码编码原理分析

最常用的检错纠错手段是利用 Xilinx 提供的 FRAME_ECC。FRAME_ECC 是 Xilinx 官方提供检错纠错电路模块，可以在 FPGA 设计过程中直接例化使用。它放置在内部配置存储器中。它采用的编码方式是 13 位汉明码，在系统上电时，会计算初始的各个配置帧对应的 ECC 值，并对其进行存储。之后一旦发现与原始存储的 ECC 值不一致则判定为出现故障，便可及时对不一致处进行修复，解决单比特翻转。但是它只能做到对单个比特位翻转进行纠正，可以检测两比特位翻转并发出警告。而随着 FPGA 设计的越来越复杂，电路越来越集中，造成连续多比特翻转的现象越来越多。所以传统的 FRAME_ECC 纠错码已经不满足需求。Xilinx 公司在 virtex-6 以及 virtex-7 系列芯片中，利用交错内存技术，在物理上对多位翻转的比特位进行分离，并通过 SEM-IP 核进行控制，可以实现对 MBU 的纠错。但是该方法只可在特定器件上实现，并且需要 microblaze 处理器来控制，这会增加资源的消耗量，而且纠错检错处理周期相对较长，因此该方法不易在实际运用中使用。

在此提出一种可以实现多位纠错的 ECC 码，里德-穆勒码（Reed-Muller，RM）（赵春雨，2017）。RM 码里的编码方案是由 Muller 在 1954 年提出的，紧

接着在同一年间，Reed 又提出了相应的解码方案，由此构成了完整的 RM 码。RM 码由于其延时较低的特性，被广泛应用在星载计算机的信息传输上。

假设存在整数 m 和 r，使得在 $0 \le r \le m$ 时，存在一个 r 阶 RM 码，并且其对应的二进制码长 l 为 2^m，便将此时的 RM 码记为 RM(r,m)（钟敏，2021）（张永光，2020）。从中我们可以得到

$$l = 2^m \quad ...(2.1)$$

$$k(r,m) = \sum_{i=0}^{r} C_m^i \quad ...(2.2)$$

$$d_{\min} = 2^{m-r} \quad ...(2.3)$$

$$b = \frac{d_{\min} - 1}{2} \quad ...(2.4)$$

其中 l 为二进制码长，k 为阶数，$d_{\min}$ 为最小码距，b 为最大可以纠正的错误位数。取 $r=2$，$m=5$，则 $d_{\min}=8$，$b=3$。所以由此可以得出 RM(2,5)纠错码可以实现对 3 位错误比特位的纠正，这比常用的 FRAME_ECC 纠错码的纠错能力要强。

RM(2,5)码编码时，假设输入数据的位宽为 16bit，那么输出数据位宽是 32bit。由以上分析可知，RM 码中存在 5 个变量，将这 5 个变量设为 (a_1,a_2,a_3,a_4,a_5)，并定义编码因子 $Q=(1,Q_1,Q_2)$，其中

$$Q_1 = (a_1,a_2,...,a_5)^T \quad ...(2.5)$$

$$Q_2 = (a_1a_2,a_1a_3,...,a_2a_3,...,a_4a_5)^T \quad ...(2.6)$$

将编码因子与输入数据进行按位异或即可得到编码后的输出数据，所以 RM(2,5)码编码原理如图 2.8 所示。待编码的 data_in[15:0]输入编码模块后，生成 32 位的编码字 data_out[31:0]，其中 data_out[31:0]中的每一位如表 2.8 所示。

表 2.8 data_out 计算方式

Table 2.8 data_out calculation method

data_out[31]	data_in[15]
data_out[30]	data_in[15] + data_in[14]

data_out[29]	data_in[15] + data_in[13]
data_out[28]	data_in[15] + data_in[14] + data_in[13] + data_in[9]
data_out[27]	data_in[15] + data_in[12]
data_out[26]	data_in[15] + data_in[14] + data_in[12] + data_in[8]
data_out[25]	data_in[15] + data_in[13] + data_in[12] + data_in[5]
data_out[24]	data_in[15] + data_in[14] + data_in[13] + data_in[12] +data_in[9]+data_in[8] data_in[5]
data_out[23]	data_in[15] + data_in[11]
data_out[22]	data_in[15] + data_in[14] + data_in[11] + data_in[7]
data_out[21]	data_in[15] + data_in[13] + data_in[11] + data_in[4]
data_out[20]	data_in[15] + data_in[14] + data_in[13] + data_in[11] + data_in[9] + data_in[7] + data_in[4]
data_out[19]	data_in[15] + data_in[12] + data_in[11] + data_in[2]
data_out[18]	data_in[15] + data_in[14] + data_in[12] + data_in[11] + data_in[8] + data_in[2]
data_out[17]	data_in[15] + data_in[13] + data_in[12] + data_in[11] + data_in[5] + data_in[4] + data_in[2]
data_out[16]	data_in[15] + data_in[14] + data_in[13] + data_in[12] + data_in[11] + data_in[9] + data_in[8] + data_in[7] + data_in[5] + data_in[4] + data_in[2]
data_out[15]	data_in[15] + data_in[10]
data_out[14]	data_in[15] + data_in[14] + data_in[10] + data_in[6]
data_out[13]	data_in[15] + data_in[13] + data_in[10] + data_in[3]
data_out[12]	data_in[15] + data_in[14] + data_in[13] + data_in[10] + data_in[9] + data_in[7] + data_in[3]
data_out[11]	data_in[15] + data_in[12] + data_in[10] + data_in[1]
data_out[10]	data_in[15] + data_in[14]+ data_in[12+ data_in[10+ data_in[8+ data_in[6+ data_in[1
data_out[9]	data_in[15]+ data_in[13]+ data_in[12]+data_in[10]+ data_in[5+ data_in[3+ data_in[1
data_out[8]	data_in[15] + data_in[14] + data_in[13] + data_in[12] + data_in[10] + data_in[9] + data_in[8] + data_in[6] + data_in[5] + data_in[3] + data_in[1]
data_out[7]	data_in[15]+ data_in[11] + data_in[10] + data_in[0]
data_out[6]	data_in[15]+ data_in[14] + data_in[11]+ data_in[10] + data_in[7] + data_in[6]+ data_in[0]
data_out[5]	data_in[15]+ data_in[13]+ data_in[11]+ data_in[10]+ data_in[4]+ data_in[3]+ data_in[0]
data_out[4]	data_in[15] + data_in[14] + data_in[13]+ data_in[11]+ data_in[10] + data_in[9] + data_in[7] + data_in[6] + data_in[4]+ data_in[3]+ data_in[0]

data_out[3]	data_in[15]+ data_in[12] + data_in[11] + data_in[10] + data_in[2] + data_in[1] + data_in[0]
data_out[2]	data_in[15]+ data_in[14]+ data_in[12]+ data_in[11]+ data_in[10]+ data_in[8]+ data_in[7]+ data_in[6]+ data_in[2]+ data_in[1]+ data_in[0]
data_out[1]	data_in[15]+ data_in[13]+ data_in[12]+ data_in[11]+ data_in[10]+ data_in[5]+ data_in[4]+ data_in[3]+ data_in[2]+ data_in[1]+ data_in[0]
data_out[0]	data_in[15] + data_in[14] + data_in[13] + data_in[12] + data_in[11] + data_in[10] + data_in[9] + data_in[8] + data_in[7] + data_in[6] + data_in[5] + data_in[4] + data_in[3] + data_in[2] + data_in[1] + data_in[0]

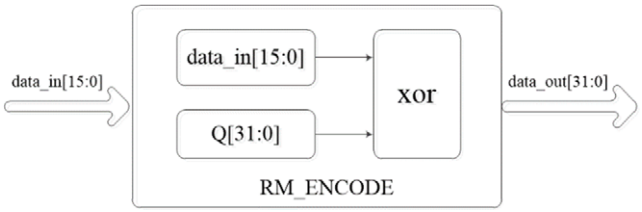


图 2.8 RM(2,5)码编码过程

Figure 2.8 the Encoding of RM(2,5)

2.5.2 RM 码译码原理

RM(2,5)码译码过程需要分为三个过程，首先计算译码后的低十位 data_decode[9:0]。低十位中的每一位需要通过 8 位的校验和确定。以 data_decode[0]为例，校验和如公式(2.7)所示。在得到 8 位校验和后，若其中 0 的个数大于 1 的个数，则 data_decode[0]=0，反之为 1，若二者相等，则说明发生了 4bit 以上翻转，该位出现错误。

$$C_{0,n} = \text{data\_out}[31-4n] + \text{data\_out}[30-4n] + \text{data\_out}[29-4n] + \text{data\_out}[28-4n](n = 0,1,...,7) \quad ...(2.7)$$

第二次计算 data_decode[14:10]，该部分由 16bit 校验和决定，以 data_decode[14]为例，计算公式如(2.8)和(2.9)所示，其中D_n为利用第一步计算结果得到的中间变量。

$$D_n = \text{data\_out}[n] - [\text{data\_decode}[9:0]]Q_2 \quad ...(2.8)$$

$$
\begin{aligned}
&C'_{14,15}=D_0+D_1 \quad C'_{14,14}=D_{16}+D_{17} \quad C'_{14,13}=D_8+D_9\\
&C'_{14,12}=D_{24}+D_{25} \quad C'_{14,11}=D_4+D_5 \quad C'_{14,10}=D_{20}+D_{21}\\
&C'_{14,9}=D_{12}+D_{13} \quad C'_{14,8}=D_{28}+D_{29} \quad C'_{14,7}=D_2+D_3\\
&C'_{14,6}=D_{18}+D_{19} \quad C'_{14,5}=D_{10}+D_{11} \quad C'_{14,4}=D_{26}+D_{27}\\
&C'_{14,3}=D_6+D_7 \quad C'_{14,2}=D_{23}+D_{22} \quad C'_{14,1}=D_{14}+D_{15}\\
&C'_{14,0}=D_{30}+D_{31}
\end{aligned}
\qquad ...(2.9)
$$

最后一次计算 data_decode[15]，通过公式(2.10)可得到位宽为 32 的值D'_n，判断该值中的 0 和 1 的数量，若全为 0 或全为 1，说明没有发生翻转，若 0 的数量多于 1，说明 0 所在的位置为翻转位，反之为 1。译码原理图如图 2.9 所示。

$$D'_n = D_n - [\text{data\_decode}[14:10]]Q_1 \qquad ...(2.10)$$

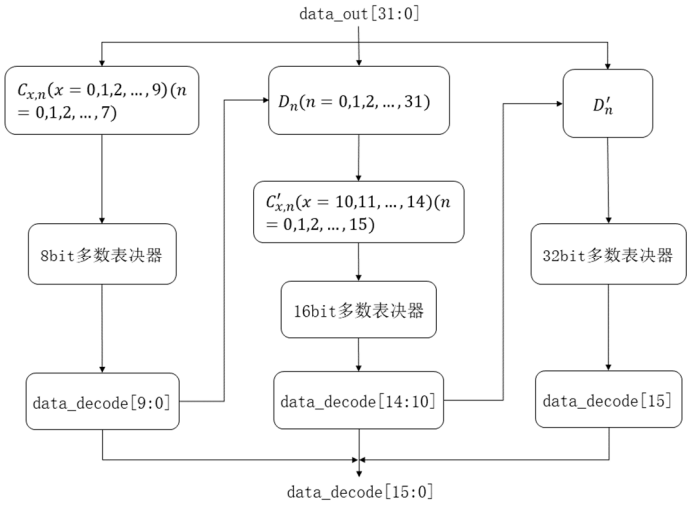


图 2.9 RM(2,5)码解码过程

Figure 2.9 the decoding of RM(2,5)

2.6 可靠性分析

没有做任何抗单粒子举措的 SRAM 型 FPGA，其可靠性 R 随时间 t 变化的曲线服从泊松分布（M. Kumar，2017），如式（2.11）所示。

$$R(t)=e^{-\lambda t} \qquad ...(2.11)$$

其中λ为单粒子翻转概率。TMR 是两个及两个以上模块正常运行，则系统正常，所以增加了 TMR 保护措施的 FPGA，其可靠性如公式（2.12）所示。

$$R(t)=3e^{-2\lambda t}-2e^{-3\lambda t} \quad ...(2.12)$$

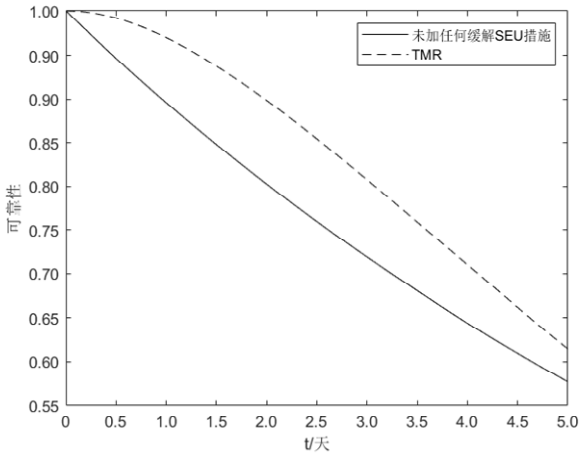


图 2.10 可靠性对比（未加任何缓解 SEU 措以及增加 TMR）

Figure 2.10 Reliability Comparison (no SEU mitigation measures vs TMR)

NASA 通过粒子束来模拟高能粒子，并对器件进行辐射，以此得到了器件单粒子翻转的概率λ大约为1.1×10^{-6}bit/day（Oki E，2002）。根据此值，通过 matlab 仿真，可以得到未加抗单粒子保护措施以及增加 TMR 后，对应的可靠性变化曲线，如图 2.11 所示。从中可以看出增加 TMR 后的可靠性明显增强。

设输入输出位宽为 n，则增加了 FRAME_ECC 纠错码的系统，其可靠性为

$$R(t)=e^{-\lambda nt}+C_n^1(1-e^{-\lambda t})e^{-\lambda(n-1)t} \quad ...(2.13)$$

而采用 RM(2,5)纠错码的系统可靠性为

$$R(t)=\sum_{i=0}^{3}C_n^i(1-e^{-\lambda t})^2e^{-\lambda(n-i)t} \quad ...(2.14)$$

图 2.11 显示了未加任何纠错码，使用 FRAME_ECC 纠错码以及 RM(2,5)码三者分别对应的可靠性曲线，从中可以看出适用了 RM(2,5)纠错码的可靠性最高，FRAME_ECC 码次之，未使用任何纠错码的可靠性最低。

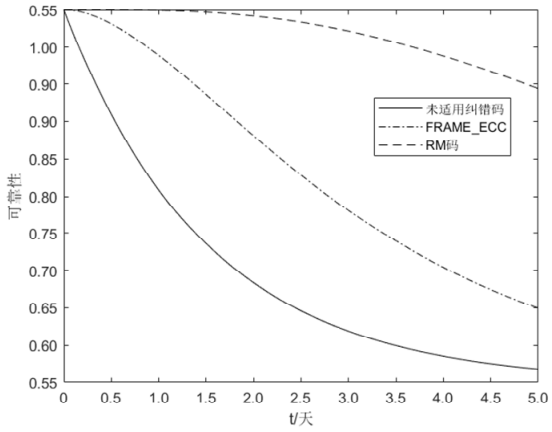


图 2.11 可靠性对比（未加任何纠错码、 FRAME_ECC 以及 RM 码）

Figure 2.11 Reliability comparison (no ECC vs FRAME_ECC vs RM code)

2.7 本章小结

本章节介绍了 SRAM 型 FPGA 的结构以及工作原理，SRAM 型 FPGA 在空间环境中所面临的单粒子现象，介绍了单粒子现象产生的原因以及会造成的影响。除此外还介绍了所使用的 ICAP 接口原理以及 RM(2,5)码的编译码原理，为后文系统设计提供了理论依据。最后就 SRAM 型 FPGA 可靠性呈泊松分布这一现象，对未加任何抗单粒子措施的 FPGA 系统以及增加三模冗余后的 FPGA 系统分别进行可靠性分析，发现增加了三模冗余后，系统可靠性大幅提高。除此外还分析了没有任何抗单粒子措施与使用 FRAME_ECC 纠错码、使用 RM 纠错码的三个 FPGA 系统间进行可靠性分析，结果发现使用 RM 纠错码的系统可靠性最高，FRAME_ECC 纠错码次之，未加任何保护措施的系统可靠性最低。由此可见，同时使用三模冗余以及 RM 纠错码后，系统可靠性将大幅增加。

第3章 抗 SEU 内部刷新系统设计

3.1 引言

结合目前国内外解决 SRAM 型 FPGA 抗单粒子翻转措施的优缺点，本文采取了基于内部 ICAP 接口的配置刷新方案，来解决 SRAM 型 FPGA 抗单粒子翻转问题，即设计了一种基于 ICAP 接口的抗单粒子翻转配置刷新系统。该配置刷新系统的详细设计方法将会在本章节中讲述。

3.2 基于 ICAP 的配置刷新系统设计

3.2.1 配置刷新系统整体结构描述

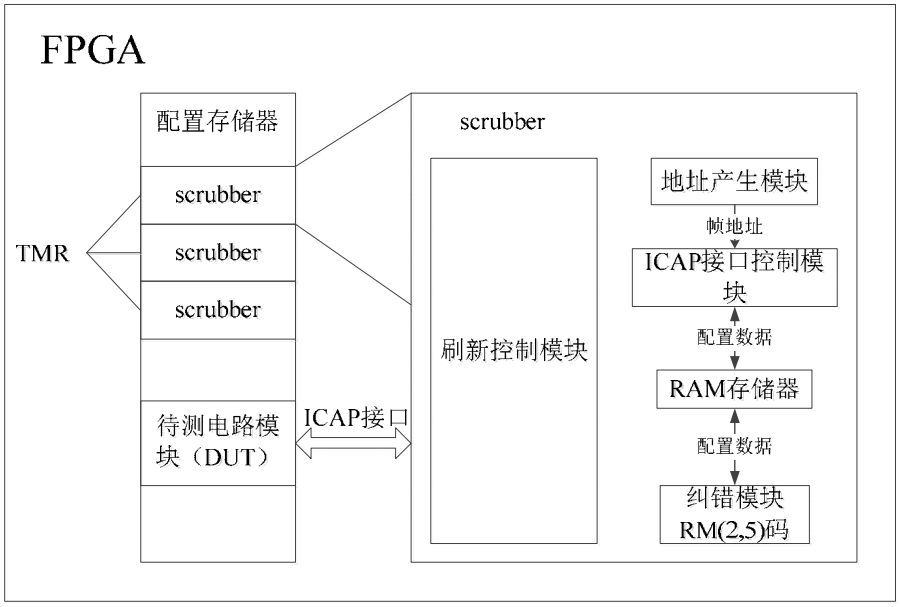


图 3.1 系统结构

Figure 3.1 the Structure of the System

本文采用内部刷新接口 ICAP 来实现对 SRAM 型 FPGA 的配置刷新，以此来提高 SRAM 型 FPGA 抗单粒子翻转能力。本文提出基于内部刷新的抗 SEU 系统框图如图 3.1 所示，主要包括刷新器模块（scrubber）、ICAP 接口，待测试电路模块（Design under test，DUT）组成（兰凤宇，2016），其中 DUT 即为实际中用户所设计的逻辑电路模块。刷新器 scrubber 为设计的配置刷新系统的核心

控制器。通过 scrubber 来实现对待测电路 DUT 模块的配置刷新，以达到抗单粒子翻转的目的。刷新器 scrubber 模块中主要包含刷新控制模块、地址产生模块、ICAP 接口控制模块、RAM 存储器以及纠错模块，纠错模块采用的是 RM(2,5)纠错码。同时因为 scrubber 自身所在电路同样可能被高能粒子辐射出现单粒子翻转问题，所以为了降低该部分电路出现故障的可能性，在此系统中特别提出对 scrubber 模块进行三模冗余加固，提高了 scrubber 容错率，同时也增加了系统的可靠性。具体设计方案如下所述。

3.2.2 地址产生模块

地址产生模块的主要功能是刷新回读提供所需要的帧地址。帧是 FPGA 配置存储器中最小的可寻址单元（闫健，2013）。无论是对 FPGA 进行初始化还是刷新操作，都需要以帧的形式回读配置数据，所以首先必须知道存储的配置数据所对应的帧地址。帧地址是由块（block）类型、顶部（top）/底部（bottom）地址、行（row）地址、列（column）地址和次（minor）地址组成（Xilinx，2018）。以 Xilinx Artix-7 xc7c100tfgg484 FPGA 为例，一共有 7 行，上半部 4 行，下半部 3 行。对于上半部分，每行有 90 列，而下半部分每行有 96 列。列包含一系列相同的资源，如 CLB Slice、I/O、CLK、BRAM 控制逻辑、DSP 等，具体情况如图 2 所示。如果一个逻辑电路被放置在一个 CLB slice 中，那么对应的配置数据就被放置在具有相同列地址的帧中。所以可以通过分析模块的位置来得到每个电路模块的帧地址。

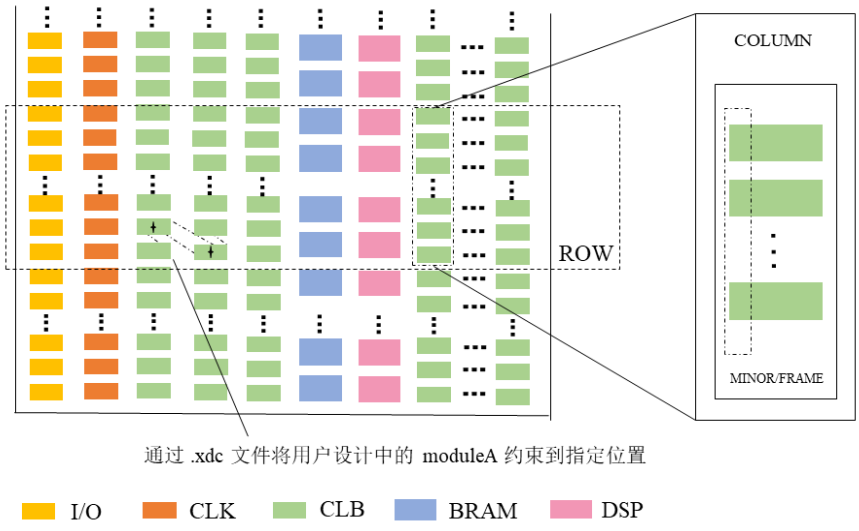


图 3.2 FPGA 可编程资源和配置帧

Figure 3.2 FPGA programmable resources and configuration frames

帧地址的确定可以分为四步。第一步查看综合后的电路，分析电路模块之间的连接方式，了解每个模块需要多少资源。第二步在原有的约束文件（.xdc）基础上，增加对电路各模块的布局布线约束，以此来将各模块固定至指定位置，约束完毕后重新进行综合和实现。示例如图 3.2 所示，该示例将模块（moduleA）限制在目标设备中的指定位置。第三，更新约束文件并重复第二步，直到成功实施。这可能需要一些时间和精力，这将是我们未来的优化工作。最后，查看 floorplan，确定各模块占用资源的位置，并根据帧地址排布可以确定模块所在位置的起始帧地址。但值得注意的是，由于帧地址不是连续排布的，所以在确定了起始帧地址后，不能简单的通过地址加 1 这种操作来去确定中间地址，所以需要根据帧地址排布规律以及帧地址寄存器（Frame Address Register, FAR）结构，来确定下一个有效帧地址。FAR 结构如表 3.1 所示。

表 3.1 FAR 结构

Table 3.1 FAR structure

地址类型	Bit 位	描述
Block Type	[25:23]	CLB、I/O、CLK (000)，BRAM 内容(001)，CFG_CLB (010)
Top/Bottom	22	上半部分为 0、下半部分为 1
Row Address	[21:17]	行地址从中心到顶部递增，然后清零再从中心到底部递增
Column Address	[16:7]	列地址从左侧为 0 开始，向右依次递增
Minor Address	[6:0]	在主列中选择一帧

根据以上对帧地址结构的分析，设计地址产生模块的端口信号如表 3.2 所示。Frame_addr_gen_start 信号是由刷新控制模块发送来的开始产生帧地址信号，该信号有效时，地址产生模块开始产生所需要的帧地址，initial_addr 作为初始帧地址，为下一帧地址的生成提供 top/bottom、行地址、帧地址等信息，并将生成的帧地址作为 next_addr 信号，发送至 ICAP 接口控制模块，去回读下一帧地址中的配置信息，与此同时拉高 frame_addr_gen_done 信号，是以此时帧地址

为有效帧地址，可以进行读取。finial_addr 信号为该部分帧地址对应的最终地址信息，若 next_addr 帧地址与 finial_addr 帧地址相等时，表明此时该部分帧地址已全部生成完毕。

表 3.2 地址产生模块端口信号

Table 3.2 Address generation module port signal

端口信号名	方向	位宽	说明
frame_addr_gen_start	输入	1	开始产生帧地址信号
initial_addr	输入	32	初始帧地址
next_addr	输出	32	生成的下一帧地址
finial_addr	输入	32	最终帧地址
frame_addr_gen_done	输出	1	帧地址有效

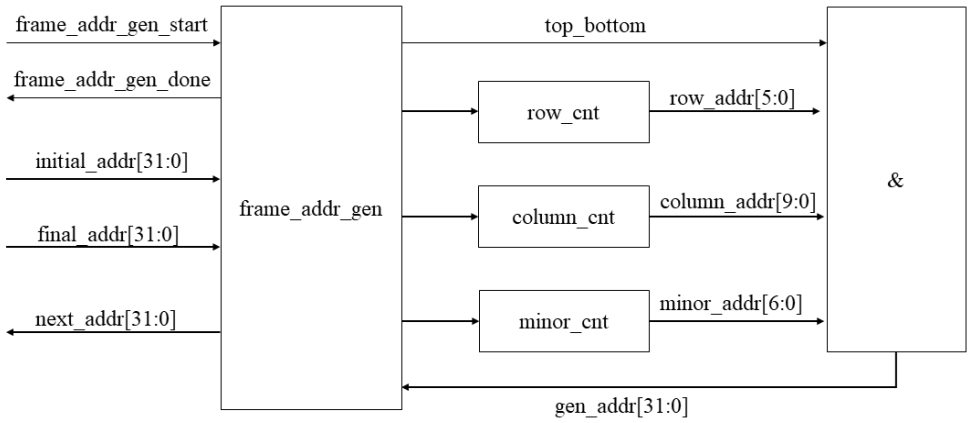


图 3.3 地址生成模块电路结构图

Figure 3.3 Circuit structure diagram of address generation module

地址生成模块电路结构图如图 3.3 所示。在开始产生帧地址（frame_addr_gen_start）信号有效时，从输入的初始地址（initial_addr）开始生成有效帧地址，并且根据表 1 还需要通过 top_bottom 信号以及行计数器

（row_cnt）、列计数器（major_cnt）、帧计数器（minor_cnt）三个计数器，来生成下一个地址，直至生成的地址为最终地址（final_addr）时，将帧地址产生完毕（frame_addr_gen_done）信号拉高，表明此时该模块的帧地址已经产生完毕。如果在此过程中，scrubber 中任何一个冗余模块出现单粒子翻转，地址产生模块将会把帧地址改为 scrubber 模块的初始地址，以此完成对 scrubber 模块的刷新操作。这样就可以避免由于错误累积导致多个冗余模块同时出现故障，引起整个电路出错。

3.2.3 ICAP 接口控制模块

如 2.4 章节所述，FPGA 可以通过 ICAP 接口对配置存储器内的配置数据进行读写。因此 ICAP 接口控制模块主要的功能便是去控制 ICAP 接口，从而驱动 ICAP 去读写配置数据。ICAP 接口所有进行的配置刷新操作都是通过控制 FPGA 内部相关配置寄存器来实现的。

3.2.3.1 FPGA 配置寄存器

FPGA 内部有许多个寄存器，其中发挥较为重要作用的为配置寄存器，因为只有通过对配置寄存器进行读写才可以对比特流进行操作。本文设计的配置刷新系统，正是利用 ICAP 接口对配置寄存器实现读写操作。在此介绍几个在配置刷新过程中，会使用到的一些配置寄存器（李麒懿，2017），相关介绍如表 3.3 所示。

表 3.3 配置寄存器介绍

Table 3.3 Introduction to configuration register

寄存器名称	写/读	地址	描述
IDCODE （设备 ID 寄存器）	写/读	01100	用于表示该芯片对应的 ID
CRC （CRC 校验码寄存器）	写/读	00000	在设备启动时，将比特流文件的 CRC 校验结果写入至此寄存器中，若当前校验值与写入数据不等，拉高 CRC_ERROR，以暂停设备的启动

FAR （帧地址寄存器）	写/读	00001	在通过 ICAP 接口读写配置数据前，需要向该寄存器写入对应帧地址
FDRI （输入帧数据寄存器）	写	00010	存储需要写入配置存储器中的数据
FDRO （输出帧地址寄存器）	读	00011	存储从配置存储器中回读到的配置数据
CMD （命令寄存器）	写/读	00100	存储需要写入配置存储器中的数据

从表 3.3 可以看出，FPGA 中定义了很多不同类型的配置寄存器，通过控制这些寄存器以实现相应的操作。比较特殊的为 CMD 寄存器，在每次向 FAR 寄存器中写入新的地址时，都需要执行一次 CMD 寄存器中的操作命令，具体的命令如表 3.4 所示。

表 3.4 CMD 寄存器操作命令介绍

Table 3.4 Introduction to CMD register operation command

Command	Code	Description
NOOP	00000	空命令
START	00101	开始启动序列命令
DESYNC	01101	取消设备同步命令
RCRC	00111	重置 CRC 寄存器命令
RCFG	00100	读取配置数据命令
WCFG	00001	写入配置数据命令

3.2.3.2 ICAP 接口读写 FPGA 配置存储器

对 FPGA 配置寄存器与配置存储器读取的主要区别在于，配置寄存器的读取内容只包含一个字，而配置存储器内最小的读取单元为一帧。读写配置寄存器中的数据比较关键的寄存器有 FDRI 以及 FDRO 寄存器，它们是配置数据输入输出的接口。但是需要注意的是，配置存储器与 FDRI、FDRO 寄存器间存在帧缓冲区，如图 3.4 所示。所以在对配置存储器读写时，每一个配置帧都需要借助一帧伪帧来实现，伪帧负责对帧缓冲区进行刷新。伪帧的大小与一帧配置

数据大小相同，其内部数据全为 0。在对配置存储器写入配置数据时，首先写入一帧有效配置数据，紧接着再写入一帧伪帧，如此才能将有效配置数据从帧缓冲区压入配置存储器中。所以如果要向配置存储器写入一帧数据，实际上需要配置两帧数据，即一帧有效数据再加一帧伪帧。而从配置存储器回读数据时，首先读出的第一帧数据其实是伪帧数据，而紧接着的第二帧数据才为实际要读取的有效帧数据。Aritex-7 中每帧数据中包含 101 个字，所以如果要读写一帧数据，那么实际上必须操作两帧数据，即 202 个字。

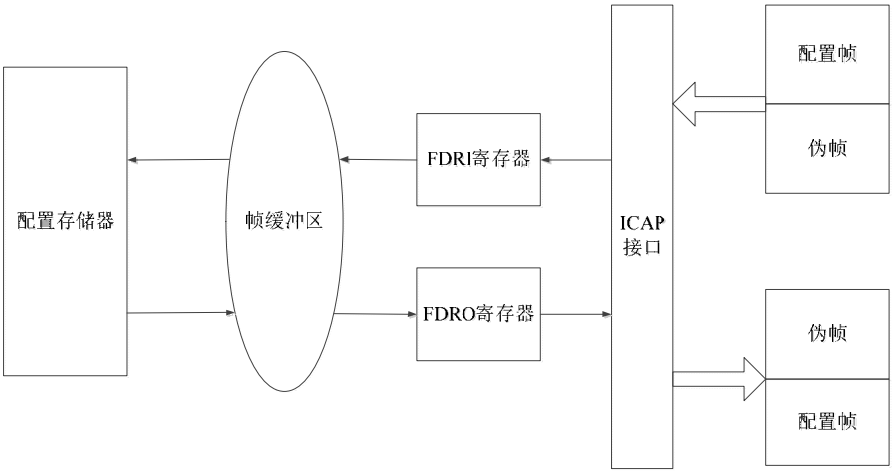


图 3.4 帧缓冲区

Figure 3.4 Frame buffer

通过 ICAP 接口读写配置存储器内容前，要发送一系列配置命令，以告知 FPGA 要进行读写配置存储器操作。通过 ICAP 来读取 FPGA 配置存储器的命令序列如表 3.5 所示。

表 3.5 回读配置存储器的命令序列

Table 3.5 Read back the command sequence of configuration memory

步骤	RDWRN	配置数据	描述
		FFFFFFFF	虚拟字（Dummy word）
		000000BB	总线宽度同步字（Sync word）
1	低电平	11220044	总线宽度检测（Detect）
		FFFFFFFF	虚拟字（Dummy word）
		AA995566	同步字（Sync word）

		20000000	Type1 空字（NOOP）
2	低电平	30008001 0000000B 20000000	Type1 类型，向 CMD 写 1 Word SHUTDOWN CMD Type1 空字（NOOP）
3	低电平	30008001 00000007 20000000	Type1 类型，向 CMD 写 1 Word RCRC 命令 Type1 空字（NOOP）
4	低电平	20000000 20000000 20000000 20000000 20000000	Type1 空字（NOOP） Type1 空字（NOOP） Type1 空字（NOOP） Type1 空字（NOOP） Type1 空字（NOOP）
5	低电平	30008001 00000004 20000000	Type1 类型，向 CMD 写 1 Word RCFG CMD Type1 空字（NOOP）
6	低电平	30002001 00000000	Type1，向 FDRO 写 1 Word FAR 地址 00000000
7	低电平	28006000 xxxxxxxx	Type1 类型，从 FDRO 读 0 个字 Type2 类型，从 FDRO 读 xxxxxxxx 个字
8	低电平	20000000 …	Type1 空字（NOOP） （共 31 个）
9	高电平	00000000 … 00000000	读回第一帧数据 … 读回第 xxxxxxxx 帧数据
10	低电平	20000000	Type1 空字（NOOP）
11	低电平	30008001 00000005 20000000	Type1 类型，向 CMD 写 1 Word START CMD Type1 空字（NOOP）
12	低电平	30008001 00000007 20000000	Type1 类型，向 CMD 写 1 Word RCRC CMD Type1 空字（NOOP）
13	低电平	30008001 0000000D	Type1 类型，向 CMD 写 1 Word DESYNC CMD

14	低电平	20000000	Type1 空字（NOOP）
		20000000	Type1 空字（NOOP）

更进一步来讲，回读配置存储器的步骤如下：

（1） 以虚拟字“FFFFFFFF”开始，之后对总线宽度进行检测，并以“AA995566”同步字为第一阶段结束，在此后需要跟一节“20000000”，视为空字，即此时无需执行任何动作；

（2） 对 CMD 命令进行设置，以激活 SHUTDOWN 命令；

（3） 对 CMD 命令进行设置，以激活 RCRC 命令，对 CRC 寄存器进行重置；

（4） 对 CMD 命令进行设置，以激活 RCFG 命令，提示设备将要进行对配置存储器的读取操作；

（5） 对 FAR 寄存器进行设置，设定将要回读的帧地址；

（6） 对 FDRO 寄存器进行设置，设定待回读的帧数量，最少回读一帧数据；

（7） 接收回读到的帧数据；

（8） 对 CMD 命令进行设置，实现去同步，以结束对帧数据进行回读。

通过 ICAP 接口来回写至 FPGA 配置存储器步骤与回读类似，具体如表 3.6 所示。与回读配置存储器不同的是，回写配置寄存器需要先向 IDCODE 寄存器中，写入器件的 ID。

表 3.6 回写配置存储器的命令序列

Table 3.6 Write the command sequence of configuration memory

步骤	RDWRN	配置数据	描述
1	低电平	FFFFFFFF	虚拟字（Dummy word）
		000000BB	总线宽度同步字（Sync word）
		11002244	总线宽度检测（Detect）
		FFFFFFFF	虚拟字（Dummy word）
		AA995566	同步字（Sync word）
		20000000	Type1 空字（NOOP）
2	低电平	30008001	Type1 类型，向 CMD 写 1 Word
		0000000B	SHUTDOWN CMD

		20000000	Type1 空字（NOOP）
3	低电平	30008001	Type1 类型，向 CMD 写 1 Word
		00000007	RCRC CMD
		20000000	Type1 空字（NOOP）
4	低电平	20000000	Type1 空字（NOOP）
		20000000	Type1 空字（NOOP）
		20000000	Type1 空字（NOOP）
		20000000	Type1 空字（NOOP）
		20000000	Type1 空字（NOOP）
5	低电平	30018001	写一个字到 IDCODE
		03628093	器件 ID
6	低电平	30008001	Type1 类型，向 CMD 写 1 Word
		00000001	WCFG CMD
		20000000	Type1 空字（NOOP）
7	低电平	30002001	Type1 类型，写一个字到 FDRI
		00000000	FAR 地址 00000000
8	低电平	30004065	写 101 个字至 FDRI
		Frame_data	待写入的有效帧数据
9	低电平	20000000	Type1 空字（NOOP）
		…	（共 31 个）
10	低电平	30008001	Type1 类型，向 CMD 写 1 Word
		00000005	START CMD
		20000000	Type1 空字（NOOP）
11	低电平	30008001	Type1 类型，向 CMD 写 1 Word
		00000007	RCRC CMD
		20000000	Type1 空字（NOOP）
12	低电平	30008001	Type1 类型，向 CMD 写 1 Word
		0000000D	DESYNC CMD
13	低电平	20000000	Type1 空字（NOOP）
		20000000	Type1 空字（NOOP）

3.2.3.3 ICAP 接口控制模块实现

该模块的主要功能为通过 ICAP 接口对 FPGA 配置存储器进行读写操作（佟昕，2018）。通过之前对 ICAP 接口以及读写配置存储器的介绍，设计了 ICAP 接口控制模块。ICAP 接口控制模块电路图如图 3.5 所示。

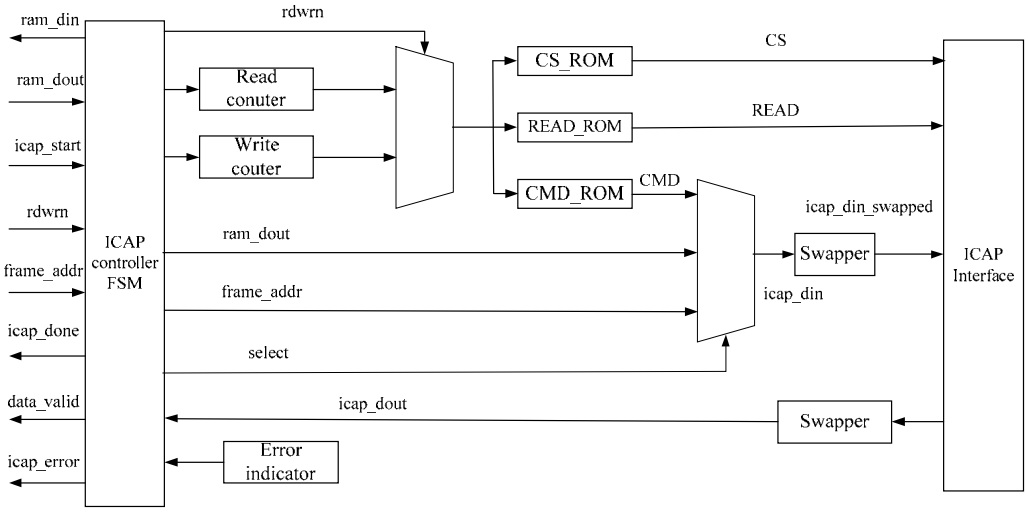


图 3.5 ICAP 接口控制模块电路图

Figure 3.5 Circuit diagram of ICAP interface control module

ICAP 接口控制模块所需要的端口如表 3.7 所示。icap_start 信号为开始读写配置数据信号，若该信号有效，则表示将开始对配置数据进行读写，icap_start 信号有效时，CSIB 使能信号拉低，使 ICAP 接口使能。icap_done 信号则表示读写配置数据已完成。frame_addr 为待读写的配置帧地址。因为 ICAP 接口存在着帧数据缓冲器，所以需要 data_valid 信号来标识是否为有效数据。ICAP 数据接口 data 信号主要的来源有三处来。一是根据 Xilinx 提供的配置手册，在通过 ICAP 读写之前都需要先发送配置命令；二是由地址产生模块提供的帧地址；三是 RM 检错纠错模块检查到回读到的数据发生了 SEU 后，对数据进行纠正，纠正后的数据需要重新通过 ICAP 写入原始帧。

表 3.7 ICAP 接口控制模块端口信号

Table 3.7 ICAP interface control module port signal

端口信号名	方向	位宽	说明
clk	输入	1	系统总时钟
rst_n	输入	1	系统复位信号

icap_start	输入	1	开始读写配置数据信号
icap_done	输出	1	读写配置数据结束信号
rdwrn	输入	1	来自刷新控制模块的读写模式切换信号
RDWRN	输出	1	ICAP 读/写模式切换信号，高电平为读模式，低电平为写模式
CSIB	输出	1	ICAP 使能信号，低有效
frame_addr	输入	32	帧地址
data_valid	输出	1	数据是否有效信号
ram_addr	输出	7	RAM 地址
ram_din	输出	32	输入至 RAM 的数据
ram_dout	输入	32	RAM 输出数据

ICAP 接口控制模块通过状态机的跳转来控制完成对配置存储器的读写。其状态机根据读写模式，分为两个支路，如图 3.6 所示。ICAP 接口控制模块在上电复位后，处于 IDLE 初始状态。待收到从刷新控制模块发来的 icap_start 信号后，状态机进入到 INITIAL 状态。该状态下，要对 ICAP 接口进行初始化，并将 ICAP 的读写模式切换至写模式。完成后，进入到 SYNC 状态，向 ICAP 接口发送同步命令，进行设备同步。设备同步之后，切换至 RCRC 状态，实现对 CRC 寄存器的配置。之后确定刷新控制模块给出的读写命令。

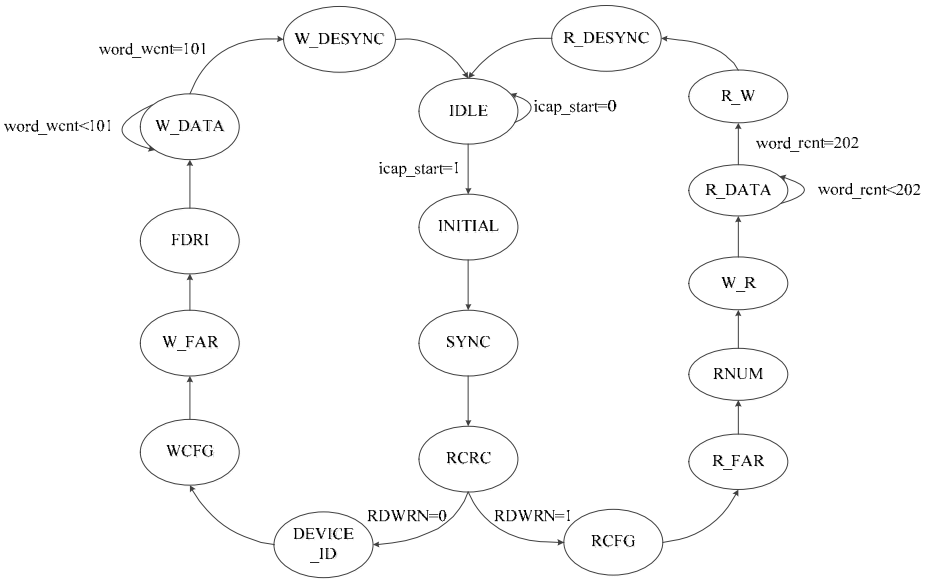


图 3.6 ICAP 接口控制模块状态机

Figure 3.6 The FSM of the ICAP interface control module

若为读命令，即来自刷新控制模块的 RDWRN 为 1，那么接下来状态机进入读模式支路，即将状态机跳转到 RCFG 状态，向 CMD 寄存器发送 RCFG 命令。之后进入 FAR 状态，将待回读的帧地址写入 FAR 寄存器。写完后进入 RNUM 状态，写入待读取的字个数。然后需要进入 W_R 状态，将 ICAP 由原先的写模式切换为读模式。切换完成后才能进入 R_DATA 状态，开始对配置数据进行回读。因为读取 1 帧数据同时需要读取一帧伪帧数据，所以待回读字计数器 frame_rcnt 值达到 202 时，才可以将状态机切换至 R_W 状态，否则仍处于 R_DATA 回读数据状态。R_W 状态下，将 ICAP 读模式切换至写模式，切换完成后，进入到 DESYNC 去同步状态，到此完成一次回读配置数据操作。

若为写模式，即 RDWRN 为 0，则状态机从 RCRC 状态进入到 DEVICE_ID 状态，对设备 ID 进行确认。确认无误后进入到 WCFG 状态，向 CMD 寄存器发送 WCFG 命令。之后进入 W_FAR 状态，向 FAR 寄存器写入待配置的帧地址。写完帧地址后，进入到 FDRI 状态，将待写入的数据发送至 FDRI 寄存器中。写完后进入到 W_DATA 状态，实现对数据的写入操作，待写入完成后，即 frame_wcnt 值等于 101，一帧数据写入完毕，状态机进入到 W_DESYNC 状态，实现设备去同步化，至此一帧数据写入完毕。

ICAP 接口控制模块具体的算法如图 3.7 所示。首先在判断到 icap_start 信号有效后，ICAP 接口控制模块开始工作，然后判断当前为读模式还是写模式。若为读模式则向 ICAP 数据接口发送读命令序列，并将回读到的配置数据存储至 ram 存储器中，如果为写模式，则发送写命令序列，并将 ram 存储器中的配置数据发送至 ICAP 数据接口。

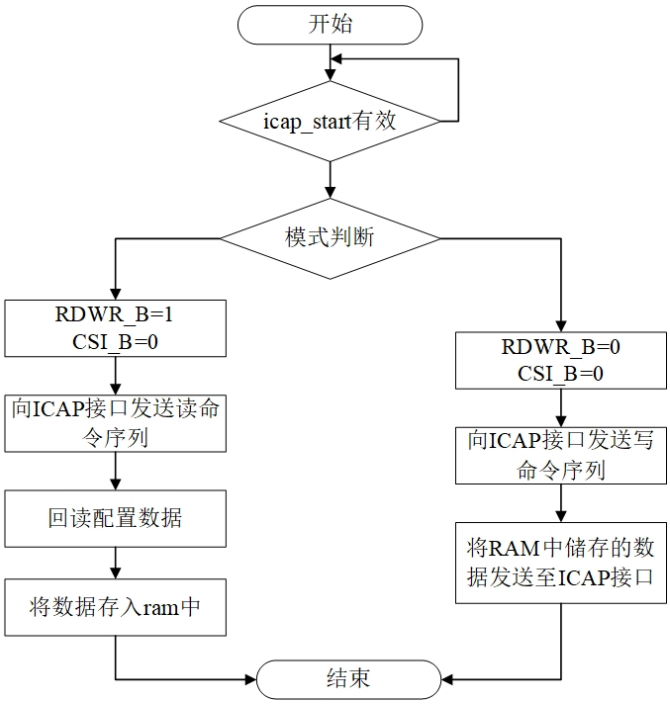


图 3.7 ICAP 接口控制模块算法

Figure 3.7 The algorithm of ICAP interface control module

3.2.4 RM 检错纠错模块

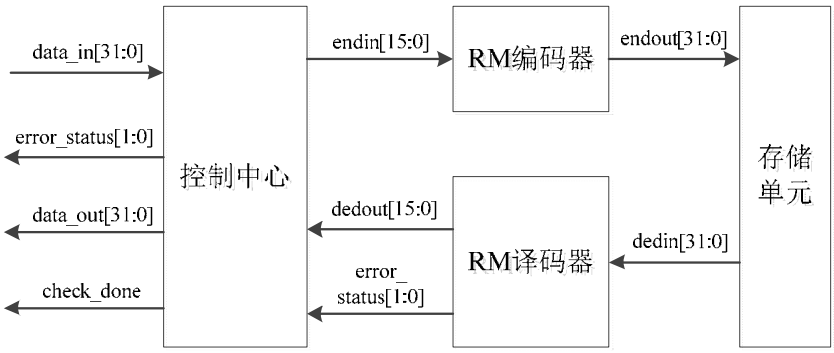


图 3.8 RM 检错纠错模块电路结构

Figure 3.8 Circuit structure of RM error detection and correction module

根据第 2 章中 2.5 节介绍的 RM(2,5)码编码译码原理，设计出 RM 检错纠错模块。RM 码检错纠错模块电路结构图如图 3.8 所示，并且其中的部分端口信号如表 3.8 所示。

表 3.8 RM 检错纠错模块端口信号

Table 3.8 RM error detection and correction module port signal

端口信号名	方向	位宽	说明
data_in	输入	31	待编码的数据
data_out	输出	31	译码后的数据
error_status	输出	2	输入数据错误状态指示（00：无错误出现；01：出现了 3 位以下的错误并进行了修复；11：出现了 3 位以上错误，无法修复）
check_done	输出	1	检错纠错完毕

纠错模块主要功能是对回读的数据进行检查并判断是否存在单粒子翻转，若翻转则进行纠正。在此采用的纠错码为 RM(2,5)码，最多对连续 3 位单粒子翻转进行纠正，可检测连续 4 位翻转但无法修复。回读的数据经过纠错模块内的编码单元后，对数据进行编码，并将编码后的数据存储到 BRAM 中，之后再从 BRAM 中读出并经过译码单元进行译码处理，在对数据译码过程中，判断数据是否出错，若出错位数低于 3 位，译码单元在译码过程中可直接进行纠正，将数据进行回复，但若超过 3 位错误，译码单元便无法进行修复，此时会给出错误信号 error_status[1:0]，00 表示无错误出现，01 出现了 3 位以下错误且进行了纠正，10 表示出现 4 位翻转，11 表示 4 位以上的翻转。10 以及 11 状态都无法对翻转直接进行纠正。

3.2.5 刷新控制模块

刷新控制模块是整个刷新器逻辑控制中心，负责向其他各个模块下发各类动作指令。在此为了对设计的内部刷新系统进行验证，增加了故障注入功能，因此整个刷新系统变成了如图 3.9 所示，图中虚线框中即为增加的故障注入模块部分，而用户在实际使用刷新系统时，无需使用该故障注入部分功能。从图中可以看出上位机在进行故障注入时采用的是 UART 串口。

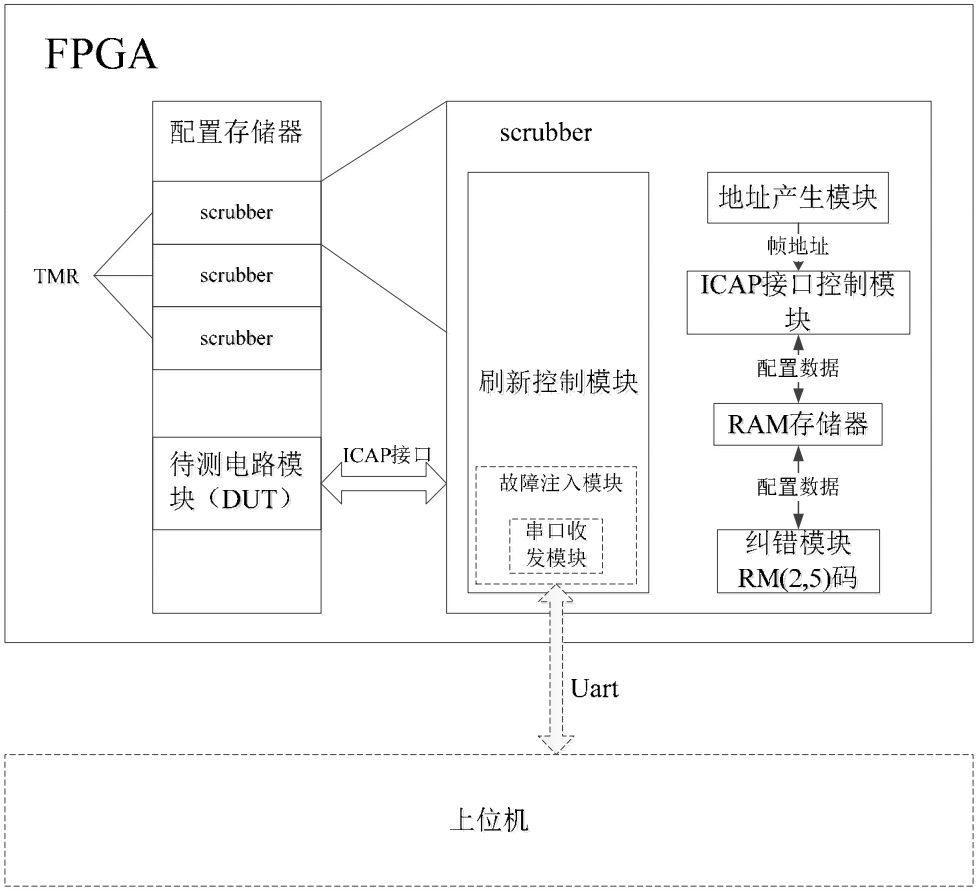


图 3.9 带故障注入的系统结构图

Figure 3.9 System structure diagram with fault injection

刷新控制模块主要包含两个功能，一是系统模式的切换，二是操作指令的发送。在这里系统的模式主要分为两种，一是故障注入模式，二是指令发送模式，其中故障注入模式的优先级要高于其他。刷新控制模块通过内部状态机来控制模式的切换和指令的发送。刷新控制模块状态机共有五个状态，分别为空闲态（idle）、故障注入状态（error_inject）、观测状态（observation）、纠错状态（correction）、错误状态（wrong）。

空闲态为系统初始状态，即 FPGA 配置完成后系统所处的状态。之后，若上位机向 FPGA 发送故障注入请求命令后，刷新控制模块将会从空闲态跳转到故障注入状态。故障注入状态下，刷新控制模块会根据来自上位机的故障注入帧地址完成故障注入操作。待故障注入完成后，若此时收到来自外部的配置刷新使能命令，则状态机进入到观测状态，若没有收到则回到空闲态。再空闲态

时，若收到刷新使能命令，同样可跳转到观测状态。观测状态下 scrubber 会根据地址产生模块生成的帧地址对配置数据进行回读，并检测配置数据是否有出现单粒子翻转，如果没有出现单粒子翻转，则状态机仍处于观测状态。在观测状态下，如果上位机发出故障注入命令，则状态机将会切回故障注入状态。如果出现单粒子翻转，则状态机从观测状态进入到纠错状态。纠错状态下，若翻转位数在 3 位以上，那么超出系统可以纠错范围，状态机跳转至错误状态，表明此时系统无法自己进行修复，需要重新对其配置才可以。如果翻转位数在 3 位以下包括 3 位，则可通过 RM 纠错检错模块对翻转位进行修复，待修复完成后状态机重新进入观测状态。具体的状态机跳转如图 3.10 所示，其中的信号如表 3.9 所示。

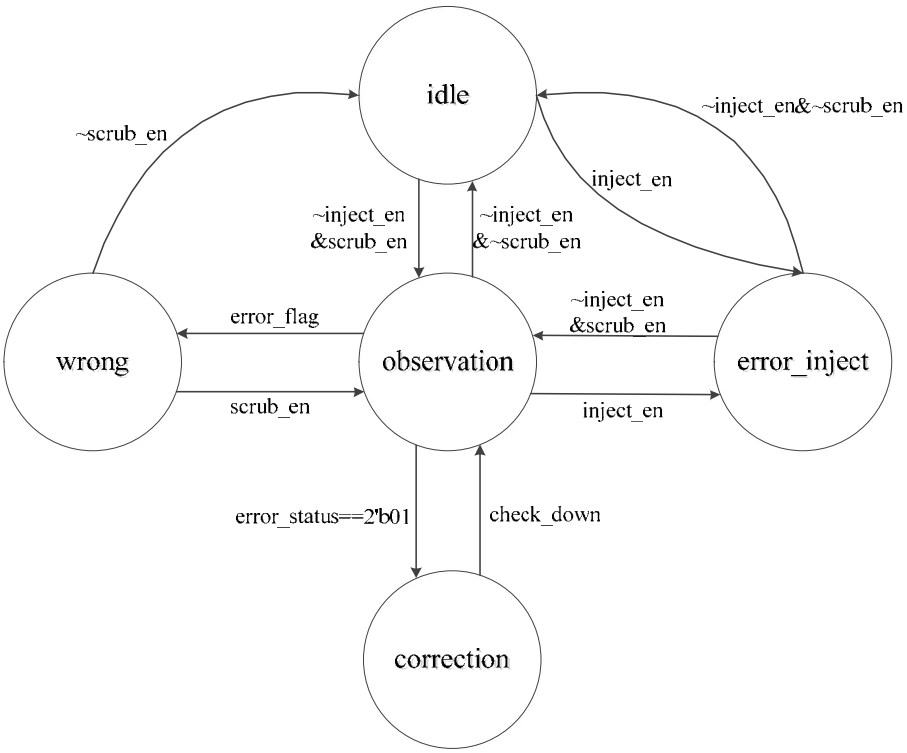


图 3.10 刷新控制模块状态机示意图

Figure 3.10 Schematic diagram of scrubbing control module state machine

故障注入过程中需要进行 1-4 位不等的翻转，以达到验证系统功能的目的，如图 3.11 所示，为进行多位故障注入时的流程。

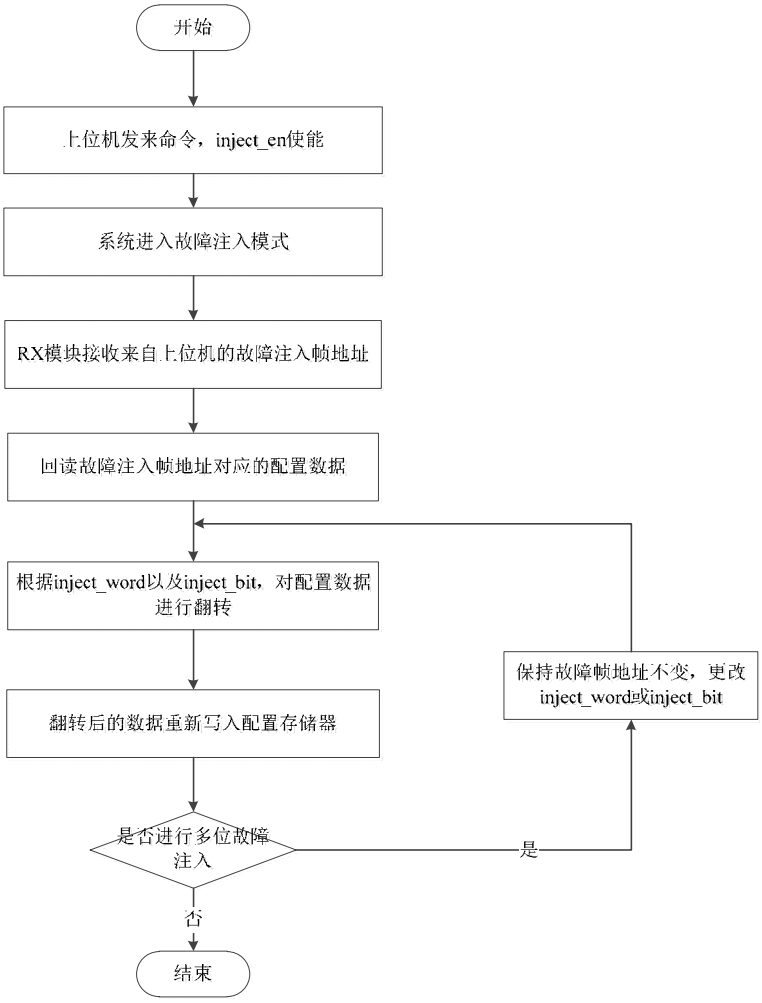


图 3.11 故障注入流程图

Figure 3.11 Fault injection flow chart

刷新控制模块端口信息如表 3.9 所示。scrub_en 为外部传来的刷新使能命令，frame_addr_gen_start 是给地址产生模块的开始产生帧地址命令，同时会将 initial_frame_addr 初始帧地址信号也给到地址产生模块，error_flag 是给到顶层模块的 3 位以上错误指示信号，方便顶层模块根据这一指示做出相应动作反馈。表格中阴影所在区域为模拟单粒子翻转，所添加的故障注入模式所需信号，inject_en 为来自上位机的故障注入开始使能标志，标志此时开始进行故障注入，而 injecct_frame_addr 即为故障注入的帧地址。check_down 为来自 RM 检错纠错模块的检错纠错完毕信号。

表 3.9 刷新控制模块端口信号

Table 3.9 scrubbing control module port signal

端口信号名	方向	位宽	描述
clk	输入	1	系统时钟
rst_n	输入	1	系统复位
scrub_en	输入	1	刷新使能标志
inject_en	输入	1	故障注入使能标志
inject_frame_addr	输入	32	故障注入帧地址
frame_addr_gen_start	输出	1	开始产生帧地址命令
initial_frame_addr	输出	32	初始帧地址
error_flag	输出	1	3 位以上错误指示
check_down	输入	1	检错纠错完毕信号

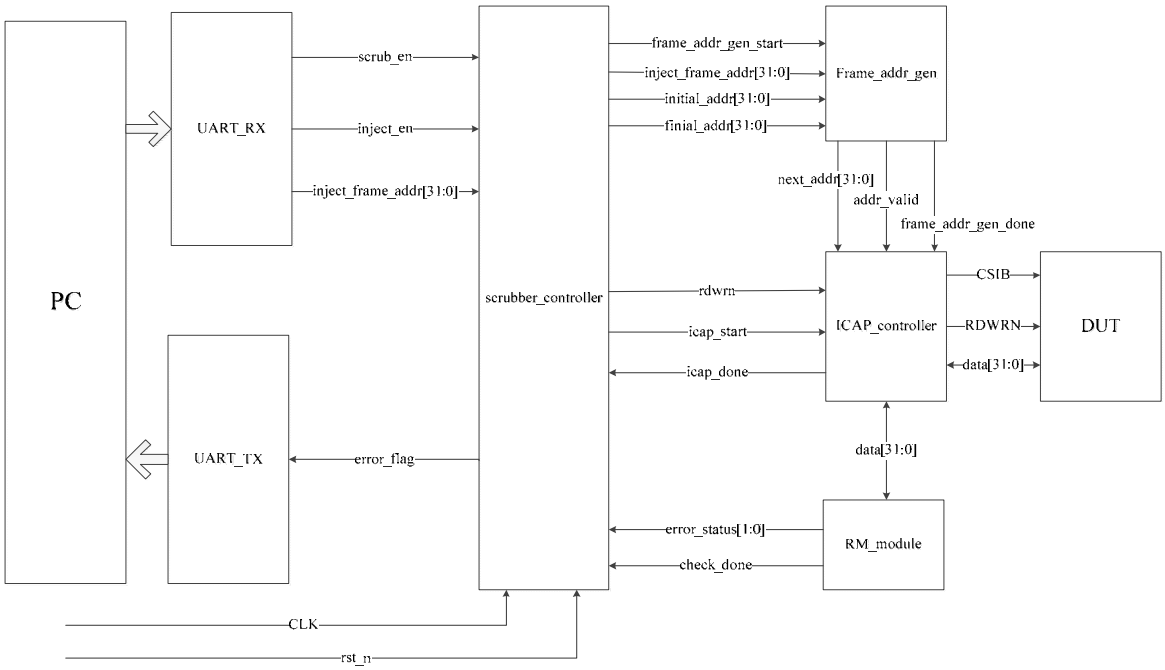


图 3.12 scrubber 电路结构图

Figure 3.12 The structure of the scrubber circuit

刷新控制模块作为整个 Scrubber 的核心模块，通过发送各种指令信息控制着其他子模块的运行。主要的指令包括开始生成帧地址指令、开始读写配置数据指令等。刷新控制模块在收到由外部发来的刷新开始命令后(scrub_en)，首先向地址产生模块发送帧地址生成命令(frame_addr_gen_start)，待纠错模块对回读的数据检错纠错完毕后，刷新控制模块会重新控制地址产生模块生成下一帧地址。如果纠错模块是 4 位以上的翻转，系统无法修复，刷新控制模块会关闭所

有使能来暂停系统工作。以上整个过程中如果 scrubber 冗余模块存在错误，会立刻关闭所有操作，并将地址产生模块的初始帧地址切换为 scrubber，重新开始对 scrubber 进行刷新。带故障注入功能的整个配置刷新系统电路结构图如图 3.12 所示。

3.3 改进的三模冗余结构

如前文所提到的，scrubber 所在电路同样会受到高能粒子辐射，发生单粒子翻转。而传统的内部刷新电路并没有对 scrubber 进行加固措施。为了解决这个问题，本文提出对 scrubber 进行三模冗余加固措施。但冗余模块仍会出现单粒子翻转，若翻转后不进行维护，随着错误累积，可能会出现多个冗余模块同时出现故障，而此时多数表决器误将多个出错冗余块产生的输出作为正确结果，进行表决，这会引起 scrubber 模块输出错误，从而导致系统出错。鉴于此，本文对每个三模冗余部分增加错误判断信号，若该信号有效，则说明冗余模块其一出现了故障，那么地址产生模块会将刷新地址改为 scrubber 初始地址，对整个 scrubber 模块进行刷新，以此达到避免错误积累的目的。经过三模冗余加固后的 scrubber 如图 3.13 所示。对其中的错误判决器输入输出真值表经过化简后，可以得到其电路图如图 3.14 所示。

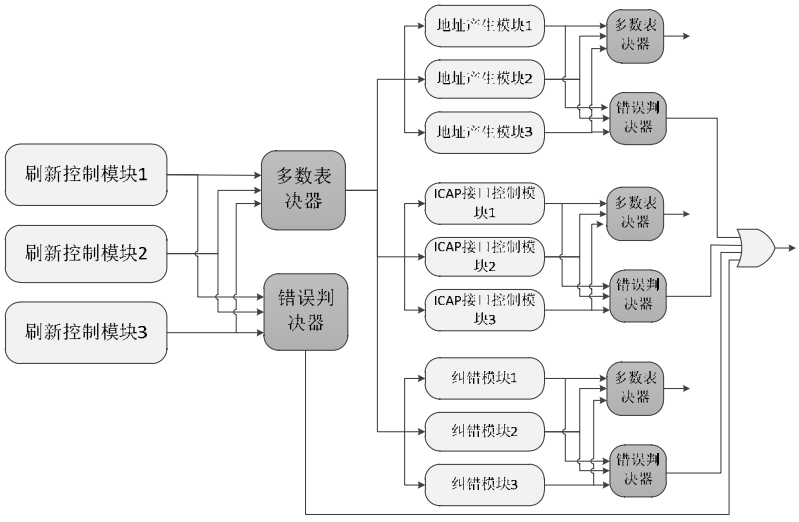


图 3.13 三模冗余结构

Figure 3.13 The Structure of the TMR

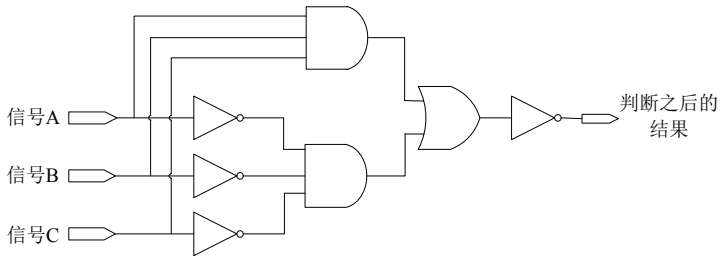
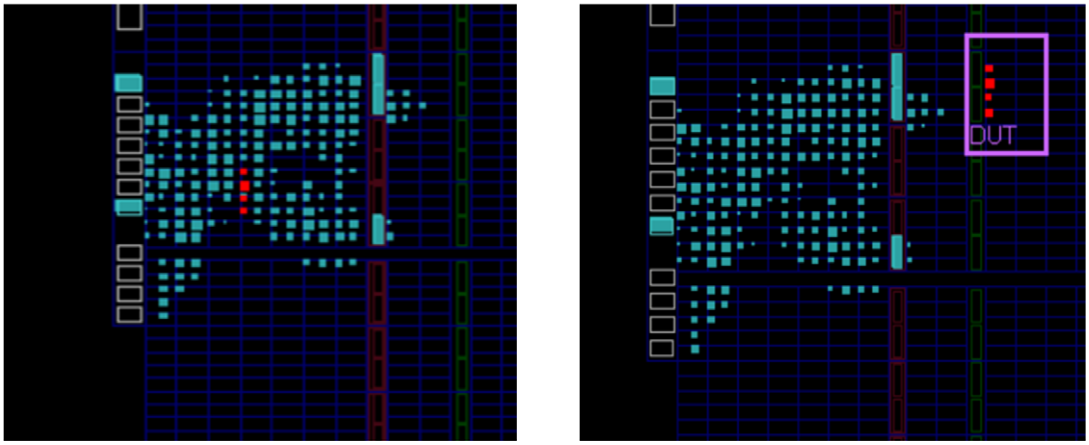


图 3.14 错误判决器电路图

Figure 3.14 Circuit diagram of error judger

3.4 分布式布局

FPGA 在布局布线时，开发工具往往会倾向于将各个模块集中分布，这不利于我们确定 DUT 以及 scrubber 分别对应的起始帧地址。所以本文提出利用 PlanAhead 中的 Pblock 工具对两个模块的分布位置进行调整。在进行布局布线时，对 scrubber 与 DUT 分布的位置进行手动的调整与约束，以此来将两个模块布局布线分隔开。这样在调整之后，可以根据 FPGA 帧地址排布规律，就可以得到 scrubber 以及 DUT 分别对应的起始帧地址，之后可根据需要，将刷新的帧地址切换到 DUT 或者 scrubber 模块。



(a) (b)

图 3.15 布局布线（a: vivado 自动布局；b:使用 pblock 调整后）

Figure 3.15 floorplan(a: by vivado; b: by pblock)

以 4 输入加法器作为 DUT 模块为例，进行逻辑综合，图 3.15 中(a)为由 vivado 工具自动布局布线后的结果，图中红色部分是 DUT 模块布局布线所在区

域，从图中可以看出 DUT 与 scrubber 模块没有很明显地被分割开。(b)为使用 pblock 调整后的布局布线图，从中可以看出经过调整布局后，可以将 DUT 与其他模块分布位置分割开。同时我们可以得到 DUT 分布是在 FPGA 的下半部分，第 2 行 30 列的第 0 帧到第 42 帧。那么根据 FPGA 帧地址排列规则以及 DUT 所在的行和列位置信息，可以得到 DUT 的有效帧地址是 0x110f00-0x110f2a。由此可以很方便地对 DUT 帧的起始位置进行定位。

3.5 配置刷新系统工作流程

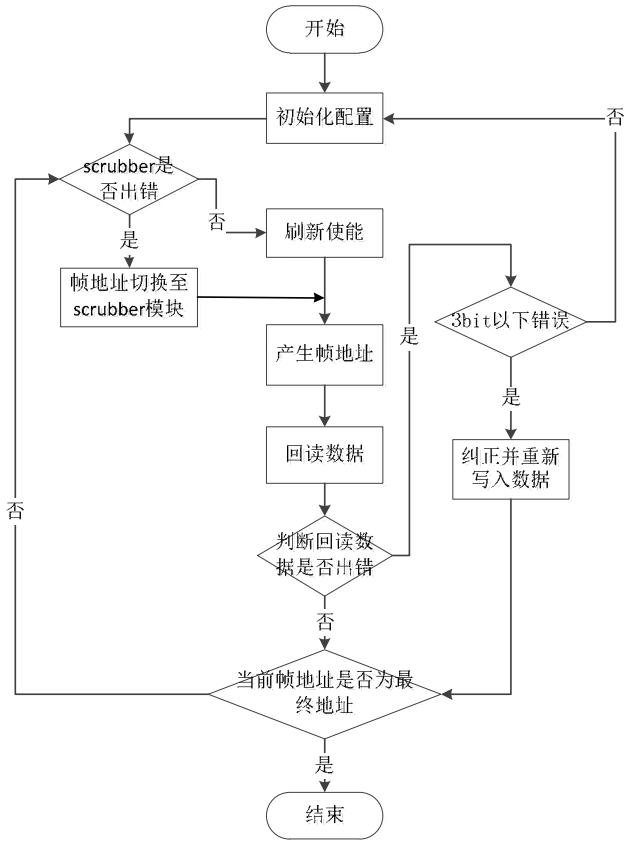


图 3.16 系统工作流程

Figure 3.16 The flow of the system

综上所述，整个内部刷新系统的工作流程如图 3.16 所示。首先对系统进行初始化配置，之后将刷新使能信号拉高，进入刷新状态，然后地址产生模块开始产生帧地址。ICAP 接口控制模块根据帧地址回读对应的配置数据，将帧地址以及读写命令序列发送至 ICAP 接口。纠错模块对回读到的配置数据进行检查，

若正确则继续判断 scrubber 是否出错，继而继续产生下一帧地址；若 3 位以下发生翻转，纠错模块纠错后再写入对应的帧地址，然后开始下一帧读写。如果发生 3 位以上的翻转，纠错模块无法恢复，那么系统要停止工作，重新对系统进行初始化。在以上过程中，若 scrubber 冗余模块发生错误，则将帧地址切换到 scrubber 的初始地址，系统进入到对 scrubber 进行刷新的状态。

3.6 本章小结

本章节详细介绍了所设计的内部刷新系统，介绍了刷新器 scrubber 的组成，以及其内部 ICAP 接口控制模块、地址产生模块、RM 检错纠错模块以及刷新控制模块这四个模块的详细设计方式、信号的组成等。除此外还介绍了本文所设计的带有错误判别能力的三模冗余结构。最后介绍了系统设计所采用的分布式布局方式。

第4章 抗 SEU 内部刷新系统功能验证与性能评估

4.1 引言

本文在 xilinx 公司推出的 Artix-7 系列芯片 xc7c100tfgg484-2 上，对上述提出的系统进行实验验证，具体实验平台如图 4.1 所示。包括 ICAP 接口控制模块、地址产生模块、刷新控制模块等各个模块的功能验证。在将比特流文件从上位机通过 JTAG 线下载至 FPGA 芯片后，可以通过 Xilinx 公司开发的调试工具 ChipScope 对各个模块内部信号进行抓取，通过观察抓取后的信号波形，来验证模块功能是否正确。除了抓取信号来判断系统功能是否正确外，还设计了串口回读方式，读取 FPGA 内部的配置数据，发送至上位机，通过上位机收到的配置数据以此来直观的判断故障注入以及纠错检错功能是否正确。除了对系统功能进行验证外，还对系统的容错能力进行了实验设计分析。

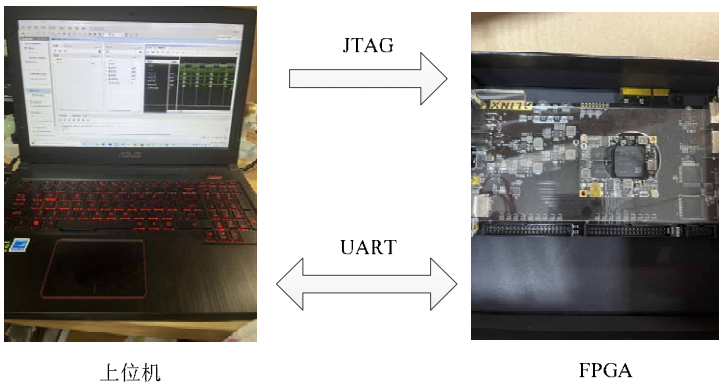


图 4.1 实验平台

Figure 4.1 Experimental platform

4.2 系统功能验证

配置刷新系统在 vivado 中进行逻辑综合后，电路图如图 4.2 所示，之后将对配置刷新系统内部各个模块的功能进行验证。

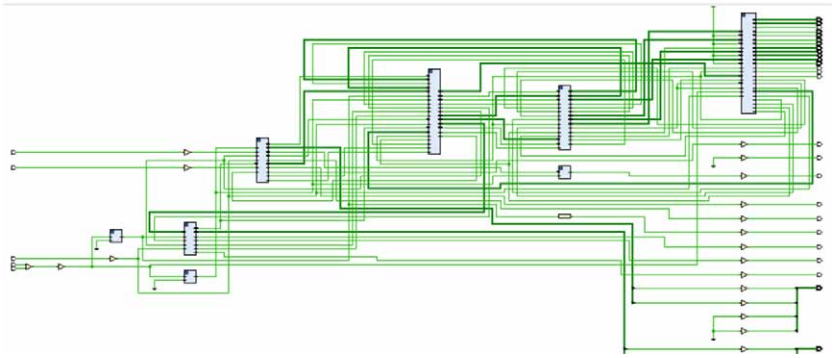


图 4.2 配置刷新系统电路图

Figure 4.2 Configuration scrubbing system circuit diagram

4.2.1 ICAP 接口控制模块

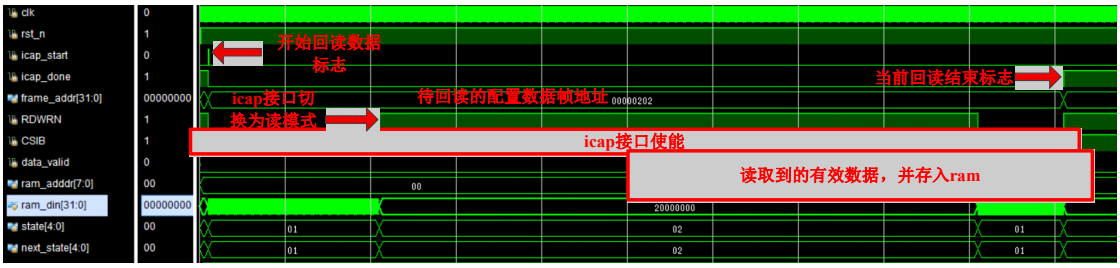


图 4.3 ICAP 回读配置数据波形

Figure 4.3 The waveform of the ICAP read back configuration data

ICAP 接口控制模块主要是用来实现对 FPGA 内部配置寄存器进行读写，在此抓取了相关关键信号，图 4.3 中显示了 ICAP 接口回读配置数据时，一些关键信号波形情况。当刷新控制模块检测到 icap_done 信号为高电平时，即此时 ICAP 接口控制模块并没有进行回读或写入任何操作，处于空闲状态，此时刷新控制模块便可通过拉高 icap_start 信号，使 ICAP 接口控制模块从空闲态进入工作模式，这一点从状态机状态表示信号 state[4:0]或者 next_state[4:0]中皆可看出。在 icap_start 信号拉高后，将 ICAP 接口端的 CSIB 使能信号拉低，以使能 ICAP 接口，之后先保持 RDWRN 为低电平，即将 ICAP 接口维持在写模式下，此目的是为了写入配置命令序列。再配置命令序列写完后，根据 ICAP 接口时序规则，等待 4 个时钟周期后，再将 RDWRN 信号拉高，使 ICAP 接口进入读模式，如图 4.4 所示。与此同时，地址产生模块也会将生成的地址传入到 ICAP 接口控

制模块中，即图中的 frame_addr[31:0]，ICAP 接口根据提供的帧地址，对配置数据进行回读，但回读到的数据因存在缓冲区，所以缓冲区回读到的数据皆为非有效数据，待缓冲区的数据读取完毕后，将 data_valid 信号拉高，是以之后回读到的数据皆为有效数据，ram 存储器在收到 data_valid 数据有效信号后，将回读到的数据存储至 ram 存储器中。

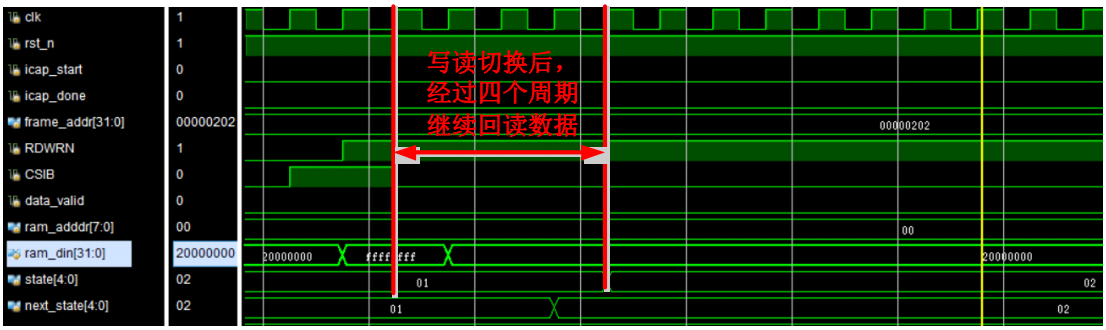


图 4.4 ICAP 回读配置数据波形细节

Figure 4.4 The waveform details of the ICAP read back configuration data

图 4.5 中显示了 ICAP 接口写入配置命令序列时，一些关键信号波形情况。ICAP 接口控制模块从储存着配置命令序列的 rom 中，读取处配置命令，并将 CSIB 信号以及 RDWRN 信号保持低电平，以使能 ICAP 接口信号并保持 ICAP 为写入状态，将从 rom 中读取的配置命令通过 ICAP 接口写入配置寄存器中。因为 ICAP 数据端存在着比特位翻转现象，所以需要把原配置命令，经过位翻转后再发送至 ICAP 数据接收端，icap_din 为位翻转后的值。

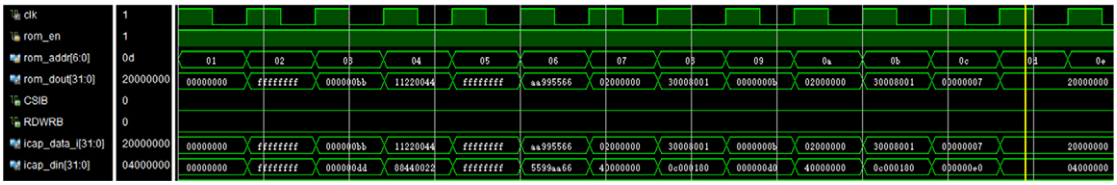


图 4.5 ICAP 接口写入配置命令序列波形

Figure 4.5 ICAP interface writes configuration command sequence waveform

4.2.2 地址产生模块

对地址产生模块的信号进行抓取，结果如图 4.6 所示。从图中可以看出，初始地址 initial_addr [31:0]=32’h00400f00，从初始地址起，每当

frame_addr_gen_start 信号有效时，开始根据行计数器 row_addr[31:0]、列计算器 column_addr[9:0]、minor_addr[6:0]计数器以及 top_bottom 信号来生成下一个帧地址，待生成完毕后，将 frame_addr_gen_done 信号拉高，目的是为了告知 ICAP 接口控制模块，可以对此时生成的地址进行采集。从中可以看出，列地址在从 01e 向 01f 转换时，此时的前后帧地址是不连续的。

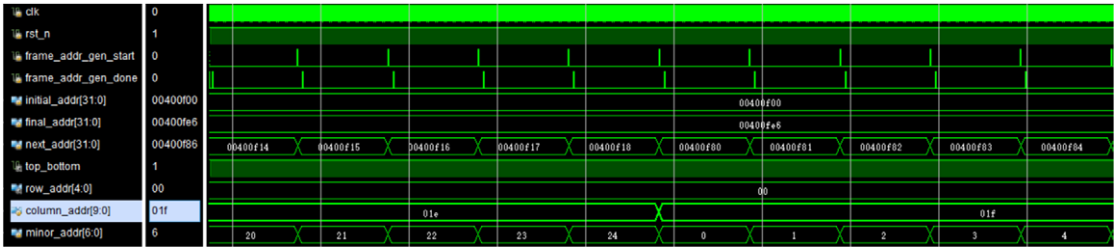


图 4.6 地址产生模块波形

Figure 4.6 The waveform of address generation module

4.2.3 刷新控制模块

图 4.7 故障注入模式波形

Figure 4.7 The waveform of Fault injection mode

如 3.2.5 章节所述，刷新控制模块主要有两种模式，一是故障注入，二是控制配置刷新。首先对故障注入模式进行验证，如图 4.7 所示。从图中可以看出，故障注入使能信号 inject_en 已被拉高，表明刷新控制模块正处在故障注入模式，而且可以看出在故障注入期间，刷新使能信号 scrub_en 一直处在低电平状态。

从图中也可以观察到此时故障注入的帧地址 inject_frame_addr 为 00000102（十六进制），具体的故障注入位置该地址下读取的配置数据的第 3 字节，第 4 比特位。在故障注入过程中，首先将 RDWRN 信号拉高，对故障注入提供的地址，通过 ICAP 接口进行回读，并将回读到的数据储存至 ram 存储中。回读完毕后再对需要进行故障注入的位置数据进行修改，修改完成后再通过 ICAP 接口，将修改完的数据重新写入配置寄存器中。

如图 4.8 所示，是根据故障注入位置，进行故障注入后通过 ICAP 接口写入的配置数据，从图中可以看到写入的配置数据 icap_din 从原来的 00000000（十六进制）修改为 00000010（十六进制），这表明了该系统成功实现了故障注入，这也从侧面验证了刷新控制模块的配置刷新功能不存在问题，因此关于配置刷新功能信号的抓取在此不再进行赘述。上述是进行 1 位故障注入的过程，如果需要进行多位故障注入，只需要保持故障注入地址 inject_frame_addr 以及故障注入字节 inject_word 不变，只需修改故障注入的比特位 inject_bit，就可以实现多位故障注入。

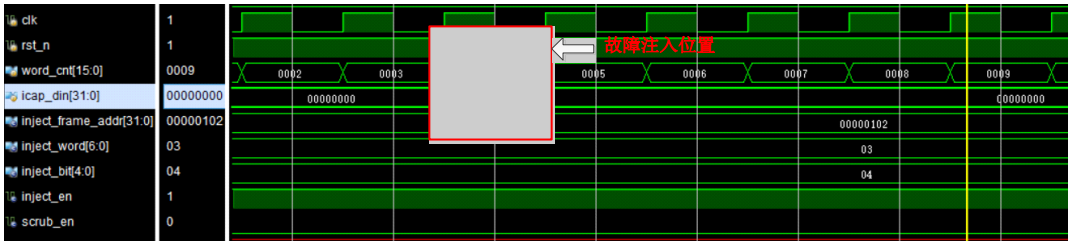


图 4.8 故障注入模式波形细节

Figure 4.8 The waveform details of Fault injection mode

4.2.4 三模冗余功能验证

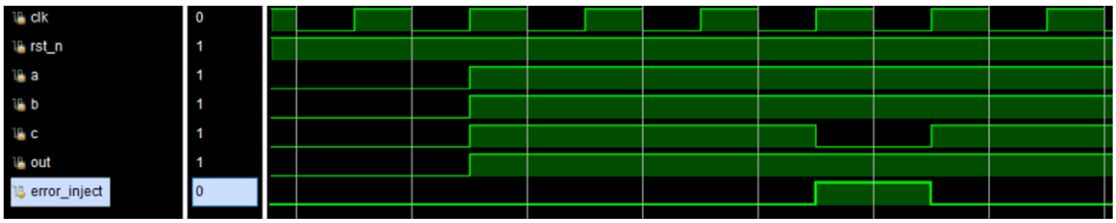


图 4.9 1 位寄存器三模冗余波形

Figure 4.9 The waveform of 1 bit register three mode redundancy waveform

根据 3.3 章节中所述，改进的三模冗余可以承担两个作用，一是当冗余模块其中一个出现错误时，能够保证输出结果仍然正确；其二在冗余模块其中一个出现错误时，及时发出警告。图 4.9 显示了 1 位寄存器三模冗余后的结果。从图中可以看出，在 a、b、c 寄存器都不存在问题时，out 保持正常输出。之后当 c 寄存器出现单粒子翻转后，error_inject 信号立刻拉高，表明此时三模冗余模块检测到内部存在故障，但是观察 out 信号也可以看出，因此时只有一个冗余模块出现问题，所以 out 信号仍然保证正常输出。之后待 c 寄存器恢复正常后，error_inject 信号也随之拉低，恢复至默认值。

图 4.10 为多位寄存器下，三模冗余波形图。从图中可以看出，即便是多位寄存器，三模冗余模块仍能保持正常结果输出。同样的在 c 寄存器由 16'haa55 翻转为 16'haa15 时，三模冗余仍能迅速判断出故障，并且将 error_inject 拉高。

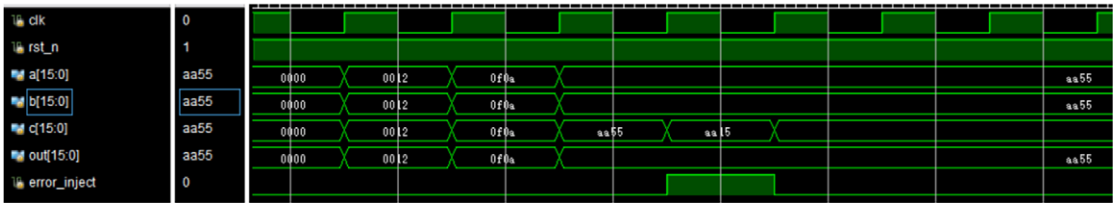


图 4.10 多位寄存器三模冗余波形

Figure 4.10 The waveform of multi bit register three mode redundancy waveform

4.2.5 检错纠错功能验证

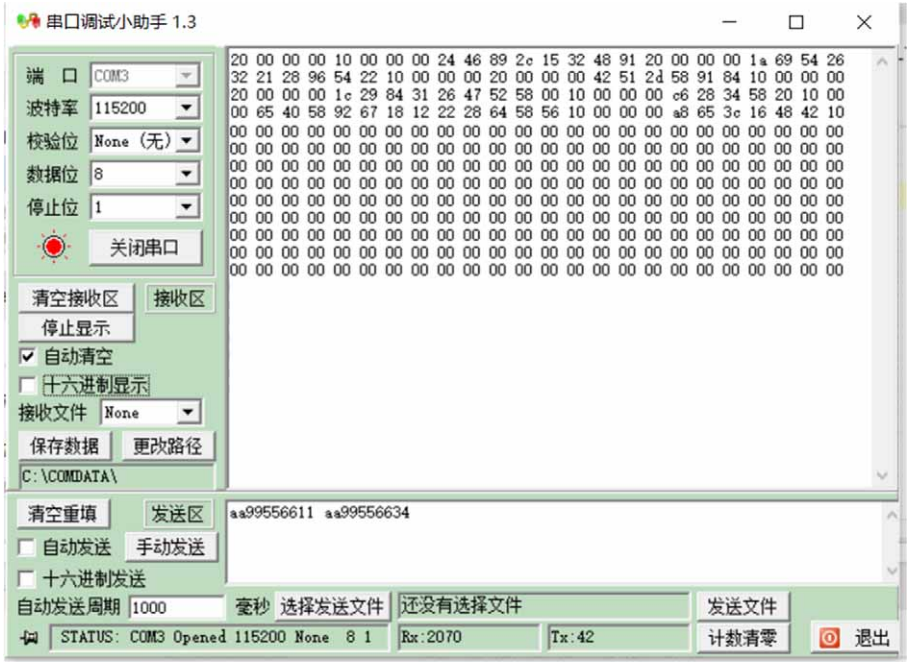


图 4.11 串口调试助手数据显示

Figure 4.11 Serial port debugging assistant data display

为了更加直观的验证系统纠错检错功能是否正确，在此通过添加的串口发送模块，将回读到的配置数据，通过 uart 串口发送至上位机中，并在上位机中通过串口调试助手，对接收到的数据进行显示，如图 4.11 中所示。从图中可以看出“aa99556611”为开始配置刷新命令，此时配置刷新系统进行正常的刷新功能，并将刷新过程回读到的配置数据发送至上位机中，收到的数据如图 4.11 中串口调试助手的接收区所示。同理要进入故障注入模式，只需要上位机发送提前约定好的命令“aa99556634”，那么配置刷新系统就将切入到故障注入模式中。

将配置刷新、故障注入以及检错纠错后配置存储器的数据回读发送至上位机，得到的部分数据结果如图 4.12 所示。首先对原始配置数据进行收集，如图 4.12 中（a）所示，便是系统的初始配置数据，而图（b）则是收集到的进行故障注入后的数据，图中红色框为进行了 1 位单粒子翻转，蓝色框为 2 位，黄色框里的为 3 位单粒子翻转。图（c）是系统经过纠错检错后收集到的配置数据，与图（a）相对比，可以清楚地看出系统成功完成了对翻转位的纠错检错。

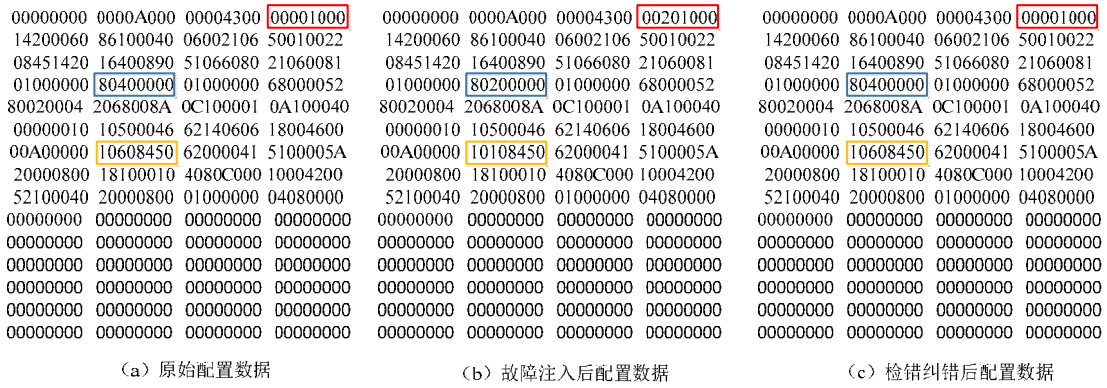


图 4.12 回读数据

Figure 4.12 Readback data

4.3 系统性能评估

为了评估系统的抗单粒子翻转性能，最好的测试环境是将 FPGA 系统放置在模拟空间环境下，使其接受高能粒子辐射，并实时观测 FPGA 系统的运行情况，最终来判断该 FPGA 系统的抗单粒子能力。但是该测试方法十分复杂，所需测试环境要求较高，不易实施，而且该测试方法需要耗费大量时间。因此本文采用上文提出的故障注入的方式来模拟系统所遭受的单粒子翻转情况。

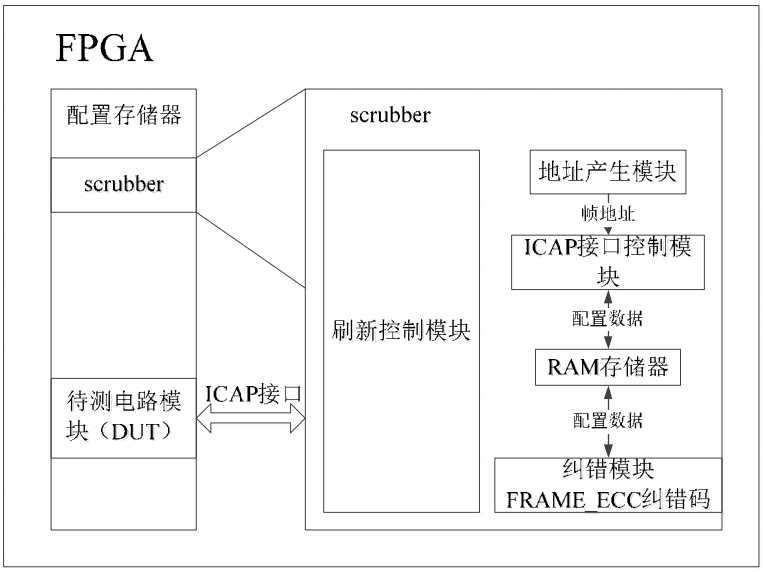


图 4.13 传统内部刷新系统

Figure 4.13 Traditional internal scrubbing system

传统的内部刷新系统如图 4.13 所示，它是由刷新器（scrubber）、ICAP 接口、待测试电路模块（Design under test，DUT）组成，其中 DUT 即为实际中用户所设计的逻辑电路模块。从图中可以看出刷新器所在区域不具有任何抗辐射措施，所以该部分易发生单粒子翻转。若刷新器发生故障，则会导致刷新系统崩溃，使整个电路系统无法正常工作。除此外，该传统内部刷新系统还存在的主要问题是其回读检测采用的是 FRAME_ECC 纠错码，该纠错码只可以对 1 位翻转位进行修改，对连续 2 位翻转位进行检错，检错纠错能力较弱。为了能更好的说明本文所设计的刷新系统具有更好的抗单粒子翻转性能，本文设计了 3 种刷新系统方案来综合比对，来说明改进后刷新系统抗辐射性能。

DUT 模块首先选择为常用的 4 位加法器，除此外，为了更贴合实际情况，

还从 IWSL2005 基准电路中选取 3 种不同的电路作为 DUT 模块，这 3 个电路分别代表着不同的应用领域，其中一个为组合逻辑较多的算术电路（FPU），另一个为时序逻辑较多的控制电路（MEM_CTRL），最后一个既包含算术电路又包括控制电路（USB_FUNC）。

对以上 4 种 DUT 电路设置为 4 个不同的工程，分别进行逻辑综合。表 4.1 显示了 4 种 DUT 自身所消耗的 FPGA 内部资源，以及在这 4 种 DUT 电路下，传统内部刷新系统整体的资源消耗情况和本文所提出的改进后的内部刷新系统所消耗的资源情况。从表中可以看出，改进后的系统，由于对 scrubber 增加三模冗余，故而导致资源占用率相对传统内部刷新有所增加，资源消耗量大约是其 3 倍左右。即便如此，改进后的内部刷新系统所占用的资源仍然很低。在系统频率为 100 MHz 时，本文所设计的内部刷新系统，回读一帧需要 114 个时钟周期，也就是需要 1.14 μs，配置一帧数据需要 110 个时钟周期，即 1.1 μs。

表 4.1 资源消耗统计

Table 4.1 Resource consumption statistics

	DUT 电路		系统整体				总资源	
			传统		改进后			
	LUT	FF	LUT	FF	LUT	FF	LUT	FF
4 位加法器	8	11	732	357	2013	1054	63400	126800
FPU	2574	1627	3304	1984	4586	2681		
MEM_CTRL	429	1032	1178	1378	4694	2142		
USB_FUNC	372	343	1015	712	2453	1325		

本文采用故障注入的方式来模拟单粒子效应，以此来验证系统的抗辐射性能。针对以上 4 组 DUT 电路，将每组 DUT 所在的系统例化两次，其中一份作为对照组，另一份为实验组，并同时施加相同的激励。采用线性反馈移位寄存器 LFSR（linear feedback shift register，LFSR）生成随机故障注入帧地址，对实验组进行故障注入。在故障注入过程种，如果观察到实验组 DUT 的输出与对照

组不一致，那么此时已经注入的故障数目即为该系统最大可以接受的故障注入数目。

为了能更好的说明本文所设计的刷新系统具有更好的抗单粒子翻转性能，本文设计了 3 种刷新系统方案来综合比对，来说明改进后刷新系统抗辐射性能。

方案（1）：对传统内部刷新系统进行故障注入，以得到其最大故障注入数目；

方案（2）：将传统内部刷新系统的刷新电路部分增加三模冗余处理后，对该系统进行故障注入，得到其最大故障注入数目；

方案（3）：对本文所设计的内部刷新系统进行故障注入，得到最大故障注入数目。

为了使实验数据更准确，减少偶发性造成的影响，重复对每种方案进行 100 次故障注入实验。100 次故障注入实验后，得到的各系统最大故障注入数目如表 4.2 所示。从表中可以得到，在 4 种 DUT 电路平均下，方案（2）最大故障注入数目约为方案（1）的 1.93 倍，这说明了增加三模冗余防护，可在一定程度上提高系统的容错率。方案（3）也就是本文所提出的系统最大故障注入数目是方案（1）的 2.56 倍，是方案（2）的 1.34 倍，这表明采用纠错能力更强的纠错码可以进一步提升系统的容错能力。图 4.14 为表 2 对应的最大故障注入数目统计图，从图中也可以直观地看出方案（3）的容错率最高，方案（1）最低。综上，本文所提出的内部刷新系统方案相较于之前的传统内部刷新系统，其容错率得到了一定的提升。

表 4.2 最大故障注入数目统计

Table 4.2 Statistics of the maximum number of fault injections

DUT	方案（1）	方案（2）	方案（3）
4 位加法器	37.42(1.00x)	71.12(1.90x)	94.67(2.53x)
FPU	89.31(1.00x)	174.56(1.95x)	215.24(2.41x)
MEM_CTRL	57.20(1.00x)	109.84(1.92x)	153.30(2.68x)
USB_FUNC	54.43(1.00x)	105.25(1.93x)	142.61(2.62x)

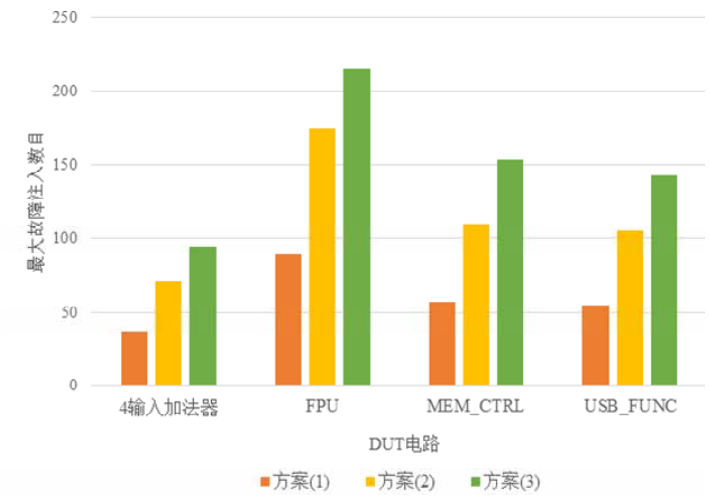


图 4.14 最大故障注入数目统计图

Figure 4.14 Statistics figure of the maximum number of fault injections

由以上故障注入实验结果可知，传统的内部刷新方式在抗单粒子翻转方面可靠性最低，而对刷新控制器增加三模冗余后的内部刷新方式可靠性有一定程度地提高。本文所设计的内部刷新系统相较于前两者，在抗单粒子翻转方面可靠性最高。由此可见，可以将本文所设计的抗单粒子翻转系统应用于航天器内的 SRAM 型 FPGA 中，以此来提高 SRAM 型 FPGA 的抗单粒子翻转能力。

4.4 本章小结

本章首先对系统功能进行验证，将所设计的系统下载配置到 Artix-7 xc7c100tfgg484-2 上，通过 chipscope 工具对系统内各个模块信号进行抓取，以此来验证了系统功能的正确性。设计了 uart 串口配置数据回读模块，以此可以直观地验证检错纠错功能地正确性。而针对于系统性能的评估，采用故障注入的方法来模拟单粒子翻转，并设计了三种不同类型的电路，包括全组合逻辑电路、时序电路以及二者电路的结合，以此来保证可以模拟到各类真实的电路情况。通过对三种电路进行故障注入，统计其可接受的最大故障注入数目，以此来说明系统的容错率。重复进行 100 次实验后，对实验结果进行统计，结果证实本文所设计的抗单粒子翻转系统容错率大幅提高。

第5章 总结与展望

5.1 总结

SRAM 型 FPGA 在空间环境下面临着严重的单粒子翻转问题。本文在经过对 SRAM 型 FPGA 抗单粒子翻转问题研究后，并结合国内外单粒子翻转问题所的进展，提出了基于 ICAP 以及 RM 纠错码的内部刷新系统。本文所做出的工作具体如下：

（1）利用 FPGA 内部 ICAP 接口，提出基于 ICAP 的内部刷新系统，该系统通过内部 ICAP 接口对配置数据进行回读与重写，以此来实现对 FPGA 的配置刷新。

（2）使用了纠错能力更强的 RM(2,5)纠错码，其可以对 3 位及其以下的翻转位进行纠错，可对 4 位比特位翻转进行检查。通过 RM(2,5)纠错码进一步提高了系统的抗单粒子翻转能力。

（3）对系统刷新器 scrubber 所在区域进行三模冗余加固，提高刷新模块的容错率。并且对三模冗余进行改进，采用了可以对三模冗余模块进行错误判断的冗余方式，如果任一冗余模块发生单粒子翻转，则会立刻对刷新模块进行刷新，以此可以避免三模冗余错误累积而导致系统出现故障。

（4）提出将刷新模块与 DUT 进行分布式布局，以此可以更加方便确定两个模块对应的帧地址。

（5）通过 matlab 对所设计的刷新系统可靠性进行分析，可以得出改进后的系统可靠性大幅增加。最后通过故障注入实验，证实了改进后的系统对故障的容错率约是传统内部刷新系统的 2.56 倍。

综上，本文所提出的基于 ICAP 的内部刷新系统可靠性以及容错率都大幅增加，为卫星在空间中存在的单粒子翻转问题的解决提供了参考设计。但鉴于设计方案不够成熟，还没有在实际运用中得到验证，并且其可靠性没有外部刷新方式高，但鉴于其设计相对简单，所以可以将其应用与低成本卫星当中。

5.2 研究内容不足

在对系统性能进行评估时，由于需要控制时间成本，导致实验总次数受到限制，可能会导致对系统性能评估不够精确，后续可以继续增大故障注入的剂量和实验次数。本文所提出的刷新系统是对配置数据都进行刷新，这种刷新方式虽然可靠，但是每次刷新也会使用大量时间。而很多学者也提出，只需对实际电路的配置数据进行刷新即可，即关键位刷新。所以后续可以系统进一步改进，对配置文件的关键位进行提取，只对关键位进行刷新即可。另外，本文提出的 RM(2, 5)码虽然提升了纠错能力，但是其需要消耗很大存储空间来存储原始的 ECC 码，编译码效率有所下降，之后可以进一步对算法进行改进。总之，本文所设计的系统还存在一定的不足之处，后续还需进一步完善。

参考文献

卜雷雷.基于 FPGA 的星载 RAM 抗 SEU 的研究与设计[D].成都：电子科技大学，2010.

邓双敏. 基于可重构芯片的故障智能自检测与演化修复[D].大连，大连理工大学，2021.

冯兴.基于商用芯片星载信号处理平台抗 SEU 技术研究与实现[D].解放军信息工程大学，2016.

何宾.EDA 原理及 VHDL 实现[M].北京：清华大学出版社，2011.

黄家提. 基于 65nm 体硅 CMOS 工艺下抗辐射 SRAM 单元设计加固方法的研究[D].安徽，安徽大学，2019.

韩涛.基于 SRAM 型 FPGA 的抗辐照加固技术研究[D].大连：大连理工大学，2021.

李超. FinFET 器件单粒子效应仿真研究[D].西安，西安电子科技大学，2017.

兰风宇. Xilinx Virtex-7 FPGA 软错误减缓技术研究[D].哈尔滨，哈尔滨工业大学，2016.

闫健. 支持可重构的流接口的设计与实现[D].复旦大学,2013.

雷利华，耿立红，金声震等.空间太阳望远镜系统的设计[J].宇航学报，2005，18(1)：38-41.

厉明坤. Zynq 配置存储器单粒子翻转检测与修复技术研究[D].哈尔滨：哈尔滨工业大学,2017.

陆启帅，陆彦婷，王地. Xilinx Zynq SoC 与嵌入式 Linux 设计实战指南—兼容 ARM Cortex-A9 的设计方法[M]. 北京：清华大学出版社，2014.

李麟懿. SRAM 型 FPGA 容错方案的设计与初步实验研究[D].武汉，华中科技大学，2017.

李昕.纳米 CMOS 集成电路抗辐射加固锁存器设计研究[D].合肥：合肥工业大学，2017.

李亚军.基于三模冗余的 ASIC 抗辐照设计方法学研究[D].大连：大连理工大学，2021.

庞波,郝维宁,张文峰等.一种 SRAM-FPGA 在轨重构的工程实现方案[J].航天器工程, 2017, 26(05): 51-56.

任小西. 基于可重构计算的高可靠星载计算机体系结构研究[D].长沙，湖南大学，2007.

舒德刚. SRAM 型 FPGA 单粒子翻转定向故障注入技术研究[D].哈尔滨，哈尔滨工业大学，2019.

孙洁朋,陈波寅,晏慧强,丛红艳,何小飞.基于 FPGA 定时刷新控制单元的应用技术研究[J].电子

与封装，2021，21(01)：65-70.

斯涛. 面向 SRAM 型 FPGA 单粒子软错误的多频度刷新技术研究[D].上海，上海交通大学，2019.

宋克非.FPGA 在航天遥感器中的应用[J].光机电信息，2010，27(12)：49-55.

石轩. 新型抗 SEU 存储器读写结构及 ECC 编码方法研究[D].西安，西安电子科技大学，2009.

佟昕,于勇,赵宝珍,张振华.基于 DMR-CED 容错方法的多相结构数字下变频 SEU 防护设计[J].遥测遥控，2018，39(04)：54-59.

田原.可见光通信系统基带传输关键技术研究[D].南京，东南大学，2019.

王喆,唐麒,王玲,魏急波.一种基于模拟退火的动态部分可重构系统划分-调度联合优化算法[J].计算机科学,2020，47(08)：26-31.

吴楠,陈强,官友廉,张万玉.FPGA 自主刷新技术研究[J].无线电工程，2019，49(12)：1094-1098.

徐文波, 田耘. Xilinx FPGA 开发使用教程（第二版）[M].北京：清华大学出版社, 2012: 7-13.

薛晓良. SRAM 型 FPGA 在辐照环境下的容错技术研究[D].中国科学院大学(中国科学院光电技术研究所)，2019.

杨海钢,孙嘉斌,王慰.FPGA 器件设计技术发展综述[J].电子与信息学报，2010，32(03)：714-727.

赵春雨. 星载计算机 SRAM 加固可靠性的研究与设计[D].哈尔滨，哈尔滨工业大学,2017.

钟敏. SRAM 型 FPGA 的 SEU 容错技术研究[D].中国科学院大学(中国科学院光电技术研究所)，2021.

赵磊，王祖林，周丽娜等. 星载 SRAM 型 FPGA 可靠性快速评估技术[J].北京航空航天大学学报，2013，39(7)：863-868.

赵培雄. SRAM 和 MRAM 器件的单粒子效应研究[D].中国科学院大学(中国科学院近代物理研究所)，2020.

张荣生.SRAM 型 FPGA 故障注入及刷新技术研究[D].哈尔滨：哈尔滨工业大学，2019.

邹三泳. 抗辐射 SRAM 的研究与设计[D].长沙，国防科技大学，2017.

张文龙. 航天应用 FPGA 可靠性设计[D].西安，西安电子科技大学，2014.

张曦文. 基于 SMIC 0.13μm 商用工艺线的 1KB 抗辐照加固 SRAM 设计[D].成都，电子科技大学，2020.

张永光.RM 码特征分析[J].微电子学与计算机，2020，37(05)：39-42.

张越,章宇兵,马天琦,李斌,石叶楠.基于 COTS 产品的高效能空间计算技术研究[J].天地一体化信息网络，2020,1(01)：54-60.

周宇澄.集成电路容软错误加固锁存器方案研究与设计[D].合肥：合肥工业大学，2017.

Balasubramanian A, Bhuva B L, Black J D, et al. RHBD techniques for mitigating effects of single-event hits using guard-gates[J]. IEEE Transactions on Nuclear Science, 2006, 52(6): 2531-2535.

Bolchini C, Miele A, Santambrogio M D. TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs[C]. IEEE International Symposium on Defect & Fault-tolerance in Vlsi Systems, 2007, pp. 87-95.

"Correcting single-event upsets through Virtex partial configuration" 2000.

Giane Ulloa, Vinícius Lucena, Cristina Meinhardt. Comparing 32nm Full Adder TMR and DTMR Architectures[C]. 2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS). 2017, 294-297.

Giordano R，Perrella S，Izzo V，et al. Redundant-Configuration Scrubbing of SRAM-based FPGAs[J]. IEEE Transactions on Nuclear Science，2017，64(9)：2497–2504.

Li J，Choutko V，Xiao L. Single Event Upset Analysis: On-orbit performance of the Alpha Magnetic Spectrometer Digital Signal Processor Memory aboard the International Space Station [J]. Nuclear Instruments and Methods in Physics Research Section A:Accelerators,Spectrometers,Detectors and Associated Equipment，2017，885：98–104.

M. Kumar, D. Digdarsini, N. Misra and T. V. S. Ram, SEU mitigation of Rad-Tolerant Xilinx FPGA using external scrubbing for geostationary mission [C]. 2017 4th International Conference on Signal Processing and Integrated Networks (SPIN), 2017, 414-418.

L. Palkuti，M. Alles，H. Hughes. The role of radiation effects in SOI technology development[C]. 2014 SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), Millbrae, CA, 2014, 1-2.

M. Kumar, D. Digdarsini, N. Misra and T. V. S. Ram, SEU mitigation of Rad-Tolerant Xilinx FPGA using external scrubbing for geostationary mission, 2017 4th International Conference on Signal Processing and Integrated Networks (SPIN), 2017, 414-418.

Mohr K C, Clark L T, Holbert K E. A 130-nm RHBD SRAM with high speed SET and area efficient TID mitigation[J]. IEEE Transactions on Nuclear Science, 2007, 54(6): 2092-2099.

P. K. Samudrala J. R. Ramos, and S. Katkoori. Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs [J]. IEEE Transactions on Nuclear Science, 2004, 51(5)：2957-2969.

Oki E, Zhigang J, Rojas-Cessa R, et al., Concurrent round-robin-based dispatching schemes for Clos-network switches[J]. IEEE/ACM Transactions on Networking，2002，10(6)：830-844.

Graham P, Caffrey M, Johnson E, et al. SEU mitigation for half-latches in Xilinx Virtex FPGAs[J]. IEEE Transactions on Nuclear Science, 2016, 50(6): 2139-2146.

Rahul Parhi, Chris H. Kim, Keshab K. Parhi. Fault-Tolerant Ripple-Carry Binary Adder using Partial Triple Modular Redundancy (PTMR)[C]. 2015 IEEE International Symposium on Circuits and Systems (ISCAS). 2015, 41-44.

U. Legat A. Biasizzo and F. Novak. SEU recovery mechanism for SRAM-based FPGAs. IEEE Transactions on Nuclear Science. 2012, 59(5): 2562-2571.

"Virtex-6 FPGA configuration user guide" 2014.

"Virtex series configuration architecture user guide" 2000.

Xilinx. Space-Grade Virtex-4QV Family Overview. 2014.

Xilinx. Radiation-Hardened, Space-Grade Virtex-5QV Family Data Sheet: Overview. 2018.

Xilinx. Xilinx TMRTool User Guide. 2017.

Xilinx Inc.7 Series FPGAs Configuration,User Guide UG470 (v1.13.1), 2018.

Zheng. S, A Rapid Scrubbing Technique for SEU Mitigation on SRAM-Based FPGAs [C], 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019, 1-5.

A. Stoddard, A. Gruwell, P. Zabriskie et al., A Hybrid Approach to FPGA Configuration Scrubbing [J]. IEEE Transactions on Nuclear Science, 2017，64（1）：497-503.