MASTER THESIS XIDIAN UNIVERSITY

西安电子科技大学

# 硕士学位论文


## 高可靠星载JPEG-LS图像压缩硬件系统设计

作者姓名 赵亮亮

学校导师姓名、职称 王柯俨 副教授

企业导师姓名、职称 李郜伟 高工

申请学位类别 工程硕士

学校代码 10701　　学　号 1501120497
分 类 号 TN91　　密　级 公开

# 西安电子科技大学

## 硕士学位论文

# 高可靠星载 JPEG-LS 图像压缩硬件系统设计

**作者姓名：** 赵亮亮

**领　　域：** 电子与通信工程

**学位类别：** 工程硕士

**学校导师姓名、职称：** 王柯俨副教授

**企业导师姓名、职称：** 李郜伟 高工

**学　　院：** 通信工程学院

**提交日期：** 2018 年 4 月

# Highly Reliable Spaceborne JPEG-LS Image Compression Hardware System Design

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in Electronics and Communications Engineering

By

Zhao Liangliang

Supervisor: Wang Keyan Title: Associate Professor

Supervisor: Li Gaowei Title: Senior Engineer

April 2018

# 摘要

近年来，人类的深空探测活动越来越频繁，随着深空探测任务目标的明确性和多样性，星载系统在整个探测活动中扮演着更加重要的角色，一个高可靠性星载系统的应用，对探测项目来说显得至关重要。

一般来说，图像的采集和传输是获取信息采取的重要方法和途径。图像采集过程中产生的数据量和信息量有一定的相关性。为了从图像中获取更多信息，往往导致数据量的快速增长，这就造成深空探测有限的硬件资源和较大数据量之间的矛盾也更加显著。所以，采用高效率的图像压缩算法，就能有效地缓解当前这一矛盾。同时，在深空环境中，由于辐照效应的存在，电子系统容易遭受单粒子效应影响而产生单粒子翻转，严重影响了系统的正常功能甚至导致系统瘫痪。论文主要从以下两部分展开：

1．JPEG-LS 图像压缩算法复杂度低，同时可实现高效的图像压缩，凭借这两点该算法能很好的满足深空探测的需求，在高效压缩的同时，节省了硬件系统的资源开销。本文根据某深空探测卫星型号任务需求，以 JPEG-LS 图像压缩算法为基础，利用 Xilinx 公司的 Virtex-2 芯片平台设计了一个完整的高可靠性的图像与数据处理 FPGA 系统。在整个数字系统中，图像编码器决定着系统的整体工作速率，本设计在传统算法的基础上主要进行了以下优化：预测误差部分，行数据的缓存，Golomb 编码进行了优化，并且去除了一些不必要的参数。

2．针对单子效应问题，简单介绍了 Xilinx 公司 FPGA 的内部结构和工作原理，进而从元器件角度分析了产生单粒子效应的机理，研究了基于 SRAM 型 FPGA 的抗辐照和容错技术。本文设计了冗余保护方法（三模冗余设计）、单帧独立处理方法、图像源异常容错处理方法等可靠性措施。其中，三模冗余方法采用改进的 XTMR 实现方案，有效避免了传统 TMR 方案存在的缺陷；单帧独立技术是每帧图像处理结束后，逻辑电路产生复位信号，对系统和编码状态进行初始化技术，为了避免错误状态的累积；图像源异常处理是对输入系统的图像进行有效性和规格检测，避免无效图像的输入。

本设计的测试部分，除了对系统进行功能和时序测试之外，还包括可靠性设计有效性的验证，设计了系统的测试平台，通过功能仿真和时序仿真验证模拟并测试了系统对单粒子翻转的有效性和对图像源异常检测的有效性。测试结果表明本设计对需求中的功能、性能、资源占用以及可靠性等要求都能很好的满足。

**关 键 词**：JPEG-LS，单粒子效应，可靠性，三模冗余设计

# ABSTRACT

In recent years, deep space exploration activities of human beings have become more and more frequent. With the clarity and diversity of targets for deep space exploration missions, spaceborne systems have played an even more important role in the entire exploration activity. A highly reliable spaceborne system The application of the program is of crucial importance to the detection project.

In general, the acquisition and transmission of images is an important method and approach for obtaining information. There is a certain correlation between the amount of data generated during image acquisition and the amount of information. In order to obtain more information from the image, it often leads to a rapid increase in the amount of data, which results in a more significant contradiction between the limited hardware resources and the large amount of data in deep space exploration. Therefore, using an efficient image compression algorithm can effectively alleviate the current contradiction. At the same time, in the deep space environment, due to the existence of the irradiation effect, the electronic system is susceptible to the single-particle effect due to the single-particle effect, which seriously affects the normal function of the system and even leads to system defects. The paper is mainly developed from the following two parts.

1. The JPEG-LS image compression algorithm has low complexity, and can achieve high-efficiency image compression at the same time. With these two points, the algorithm can well meet the needs of deep-space detection. At the same time, it saves the hardware system's resource overhead. In this paper, based on the requirements of a deep space exploration satellite model task, based on JPEG-LS image compression algorithm, a complete high-reliability image and data processing FPGA system is designed using Xilinx Virtex-2 chip platform. In the entire digital system, the image encoder determines the overall operating speed of the system. The design is mainly based on the traditional algorithm, the following optimization: the prediction error part, the cache of the row data, Golomb encoding is optimized, and some are removed Unnecessary parameters.

2. For the monotonic effect problem, the internal structure and working principle of the Xilinx FPGA are simply introduced. Then the mechanism of the single-event effect is

analyzed from the perspective of the components. The radiation-resistance and fault-tolerance technologies based on the SRAM FPGA are studied. This paper designs redundancy measures (three-mode redundancy design), single-frame independent processing methods, image source exception fault-tolerant processing methods and other reliability measures. Among them, the three-mode redundancy method adopts an improved XTMR implementation scheme, which effectively avoids the defects of the conventional TMR scheme; a single-frame independent technology is a logic circuit that generates a reset signal after each frame image processing ends, and initializes the system and coding state , In order to avoid the accumulation of error conditions; Image source exception handling is the validity of the input system image and specification detection, to avoid invalid image input.

The test part of the design, in addition to the function and timing test of the system, also includes the validation of the validity of the reliability design, the design of the system test platform, simulation and test of the system through the functional simulation and timing simulation of the single-event flip The validity and effectiveness of the anomaly detection of image sources. The test results show that this design satisfies the requirement of functions, performance, resource occupation, and reliability.

**Keywords**: JPEG-LS, single particle effect, reliability, TMR

# 插图索引

# 表格索引

# 符号对照表

| 符号 | 符号名称 |
| --- | --- |
| bit | 比特 |
| Hz | 赫兹 |
| KB | 千字节 |
| μm | 微米 |
| μs | 微秒 |
| byte | 字节 |

# 缩略语对照表

| 缩略语 | 英文全称 | 中文对照 |
| --- | --- | --- |
| FPGA | Field Programmable Gate Array | 现场可编程门阵列 |
| TID | Total Ionizing Dose | 总剂量效应 |
| SEE | Single Event Effect | 单粒子效应 |
| RTL | Register Transfer Level | 寄存器转换级 |
| IOB | Input/Output Buffer | 输入/输出缓冲器 |
| CLB | Configurable Logic Block | 可配置逻辑块 |
| LUT | Look Up Table | 查找表 |
| BRAM | Block Random Access Memory | 块随机存取存储器 |
| CAM | Content Addressable Memory | 内容可寻址存储器 |
| FIFO | First In First Out | 先入先出队列 |
| DLL | Delay Locked Loop | 延迟锁相环 |
| PLL | Phase Locked Loop | 锁相环 |
| ASIC | Application Specific Integrated Circuit | 专用集成电路 |
| SEU | Single Event Upset | 单粒子翻转 |
| SEL | Single Event Latchup | 单粒子锁定 |
| SEB | Single Event Burnout | 单粒子烧毁 |
| SEFI | Single Event Functional Interrupt | 单粒子功能中断 |
| SET | Single Event Transient | 单粒子瞬态脉冲 |
| TMR | Triple Modular Redundant | 三模冗余 |
| XTMR | Xilinx Triple Modular Redundant | Xilinx 三模冗余 |
| DCT | Discrete Cosine Transform | 离散余弦变换 |
| MED | Median Edge Detector | 中值检测器 |
| MSE | Mean Squared Error | 均方误差 |
| PSNR | Peak Signal to Noise Rate | 峰值信噪比 |

# 目录

# 第一章　绪论

古时候有嫦娥奔月的传说故事，当代有阿波罗（Apollo）号飞船的探月，对月球的向往一直以来都是人类的梦想。从 21 世纪以来，世界各国都相继开始了地月球的探索活动，开启了探月的热潮。新世纪的中国也提出了嫦娥系列月球科学考察任务，到目前为止，我国已经成果发射了“嫦娥 1 号”，“嫦娥 2 号”，“嫦娥 3 号”等月球科学考察探测器。

在所有的探测活动中，图像是获取信息的最直观的最直接的方式。同时，更高分辨率相机的应用导致图像数据量的指数增长，这样对于电子图像处理系统来说也是巨大的考验，有限的深空图像处理系统的通信传输带宽与大数据量的传输需求的矛盾也更加迫在眉睫。因此，高效率低复杂度的图像压缩系统称为解决这一矛盾的主要途径。

同时，深空探测活动所处的特殊环境，使得系统暴露在各种太空高能粒子（质子、电子、α 粒子以及各种射线）的影响，使得电子系统发生故障，其中属单粒子效应最为常见。因此，对系统进行可靠性设计，对于保障系统的正常功能的实现是很有必要的。

## 1.1 课题背景

### 1.1.1　星载系统特点

深空探测系统因其独特的工作环境，对基于电子系统的可靠性和性能的要求是非常严格的。深空环境中的辐照效应主要是来自宇宙中辐射带的带电粒子，其中总剂量效应（Total Ionizing Dose，TID）与单粒子效应（Single Event Effect，SEE）对电子器件的影响较大，FPGA 器件发生了上述效应后对整个图像压缩系统的功能影响是致命的[1]。所以，对于深空探测实践来说，应用具有高可靠性的星载图像压缩系统是很有必要的。

随着人类深空探测活动的实践积累和快速发展，人们对图像的视觉的感受要求越来越高，高质量的图像保证了实践中来自图像信息的有效性和充分性，这也使得工程采集的图像数据量呈现几何倍数增长，然而，有限的数据传输带宽和通信带宽等硬件资源限制了整个系统的性能，两者之间的矛盾日益显著，高效率的图像压缩技术是解决这些问题的重要手段，同时由于硬件资源的有限，对低复杂度易于硬件实现的图像压缩算法也是迫切需要的。因此，高性能低复杂度的图像压缩算法的研究和应用，为深空探测的进一步发展提供了保障。

### 1.1.2 图像压缩技术

现阶段常用的静态图像压缩算法有：JPEG、JPEG-2000 和 JPEG-LS 等。具体介绍如下。

（1）JPEG 标准[2]

对连续色调的静态图像，JPEG 压缩算法可以实现无损压缩，也可以实现有损压缩，其压缩流程为：

a）将目标图像以一定规格的块为单位进行分割，并从左到右从上到下依次对块进行编码；

b）对小块图像进行离散余弦变换，得到变换系数；

c）最后对变换系数转化为标量，然后进行量化；

c）扫描量化值，生成变长序列；

d）哈夫曼熵编码，生成码流。

低比特率时，从 JPEG 压缩后的码流中恢复的图像存在马赛克现象，效果比较不理想。

（2）JPEG-2000 标准[3]

JPEG-2000 标准采用了离散小波变换加算数编码机制可以达到非常好的图像压缩效果，利用同一个算法实现无损与有损压缩，有以下特征[4]：

a）同时支持无损和有损压缩；

b）压缩效率高；

c）码流随机访问和处理；

d）具有较好的抗误码等。

JPEG-2000 压缩标准虽然有良好的性能表现，却伴随着算法复杂度的提升，不利于硬件实现。

（3）JPEG-LS 标准[5]

JPEG-LS 图像压缩标准是一种全新的无损/近无损压缩技术，其主要思想是：当前像素左上方的几个相邻像素作为当前像素的上下文，利用上下文来预测误差，利用一定的误差分布，对预测误差进行 Golomb-Rice 编码。有一下特点：

a）算法复杂度低；

b）算法性能较好；

c）易于硬件实现

JPEG-LS 的上述特点正好有效解决了星载图像压缩系统在图像处理方面的所面临的问题，也是本设计的设计依据。

## 1.2 可靠性研究现状

深空中高能离子辐射和所选用的 FPGA 器件的结构，使得基于 FPGA 的深空图像处理系统会更加容易遭受单粒子的影响。过去的几年时间里，对于国内的近 40 颗地球同步卫星的异常情况统计显示[6]：卫星出现错误情况中，高达 71%是由于空间辐射效应导致的，其中单粒子效应引发的错误占 55%。因此针对单粒子效应，基于 FPGA 的深空图像编码器必须采取有效的容错技术和可靠性措施，以保证星载图像处理系统的正常工作。故障统计具体如下表所示。

表1.1 星载系统故障统计

| 类型 | 故障次数 | 所占百分比 |
|---|---|---|
| 单粒子翻转 | 621 次 | 39.08% |
| 电子诱发电磁脉冲 | 293 次 | 18.44% |
| 静电放电 | 215 次 | 13.53% |
| 其他 | 460 次 | 28.95% |
| 总计 | 1589 次 | 100% |

可以看出，单粒子效应影响了严重影响星载电子系统的正常使用，由于器件工艺尺寸不断缩小和密度的不断增加、频率的不断提高和内核电压的不断降低，星载图像压缩编码器受单粒子辐照的影响就越来越显著。纵观星载图像压缩编码器容错和可靠性技术的研究历史，研究主要集中在两个方面，第一，通过工艺的改进增强 FPGA 器件本身的可靠性，第二，通过增加冗余增加星载图像压缩编码器逻辑的可靠性。

本文从 RTL 代码设计和系统功能模块层次研究针对单粒子效应的容错和可靠性技术。

## 1.3 本文创新

对本文主要创新点归纳如下：

第一，基于 JPEG-LS 图像压缩算法，本设计通过对行像素缓存逻辑的优化，实现系统对图像的实时压缩处理并输出码流数据，不再需要额外的大容量片外存储来对整幅图像进行缓存，这很大程度上提升了系统处理图像的效率，同时节省了硬件资源。

第二，本文对系统部分逻辑模块进行了三模冗余处理，保证系统工作可靠性的同时，减少硬件资源开销。

第三，根据对输入信号的要求，设计了几种针对本系统的容错措施，并且通过仿真验证了设计的有效性。

## 1.4 本文内容安排

本文根据 XX-XX 图像与数据处理 FPGA 项目需求，以 JPEG-LS 图像压缩算法为基础，利用 Xilinx Virtex2 系列 FPGA 芯片（XQR2V3000）完成了星载图像压缩系统的实现。算法与系统开发平台为 Xilinx ISE 9.2i，结合 Modelsim se6.5e 和 Chipscope 通过了仿真验证和板级确认测试。

本文详细介绍了 JPEG-LS 图像压缩算法的原理以及基于硬件平台的实现方法，并且对算法中的关键技术点进行了分析；另外从 SRAM 型 FPGA 的内部结构以及工作原理层面介绍了空间辐照效应的产生机制和对星载 FPGA 系统的影响，并从多角度分析了避免单粒子影响的措施。

本文主要分为 6 个章节，主要内容安排如下：

第一章 绪论

简单介绍了图像压缩技术和容错及可靠性技术的相关背景和发展，着重分析了星载图像压缩编码器容错及可靠性设计的必要性和重要意义，列举出了关于星载图像压缩编码器容错及可靠性的研究现状。并且，给出了本文的几个创新点。

第二章 基于 SRAM 型 FPGA 的抗辐照分析

首先对 SRAM 型 FPGA 的结构和工作原理进行研究分析；其次研究空间辐照效应以及对 FPGA 器件的影响；在此基础上，提出针对不同类型的影响导致的器件故障的应对措施。

第三章 JPEG-LS 图像压缩算法的研究和实现

介绍 JPEG-LS 图像压缩算法的原理，实现过程中的重要技术点，针对 XX-XX 图像与数据处理 FPGA 项目技术指标进行优化与实现，压缩算法的可靠性研究和实现。

第四章 图像压缩系统容错技术与可靠性设计

包括深空抗辐照设计，以及图像压缩系统的图像源可靠性设计。具体为，1）针对单粒子翻转提出三模冗余设计，包括原理，实现平台，以及实现步骤；2）图像源异常容错设计；3）单帧独立设计；4）滤波措施；5）强直通模式。

第五章 系统测试与验证

本部分主要是对 JPEG-LS 图像压缩算法的验证，根据 XX-XX 图像与数据处理 FPGA 项目需求，详细验证了本设计是否满足设计的各项要求，除此之外，验证避免单粒子效应措施的有效性，图像源异常的容错处理的有效性，以及其他验证。

第六章 结论和展望

对本设计整体进行总结。

# 第二章　基于 SRAM 型 FPGA 的抗辐照分析

现场可编程门阵列(Field Programmable Gate Array，FPGA)具有以下鲜明的特点：集成密度高、功耗较低、开发灵活性强、适用性广等，因此，在深空探测领域中，FPGA 是相当合适的选择，并且随着科技发展，FPGA 发挥出来的作用也越来越无可取代。

因此，深空探测 FPGA 系统的可靠性问题也越来越重要。常见的 FPGA 器件可以分为：静态随机存取存储器（SRAM）型 FPGA、Flash 型 FPGA 以及反熔丝型 FPGA。其中，SRAM 型是通过查找表结构配置比特文件实现对 FPGA 的配置；Flash 型通过电荷控制门晶体管开关实现；反熔丝型则是通过金属层之间的绝缘层的通断实现对 FPGA 的配置[7]，正常情况下是断开的，熔断后就会导通。通常反熔丝型不易遭受深空辐射的影响，但容量较低，一般用于系统的控制模块。

星载系统通常选用 SRAM 型 FPGA，不可避免深空辐照的影响，本章从 SRAM 型 FPGA 的结构和工作原理出发，对其进行抗辐照分析。

## 2.1 FPGA 内部结构

目前，FPGA 主要还是基于查找表技术实现的，并附加了额外的常用功能模块。典型的 Xilinx FPGA 芯片结构如图 2. 1 所示。

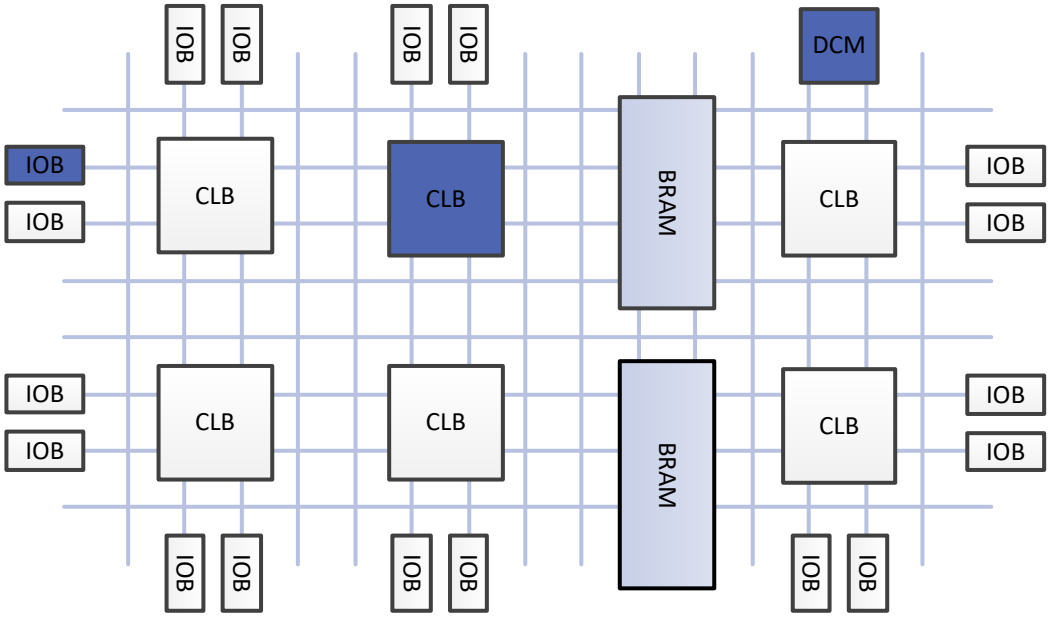



图2. 1　　FPGA 内部结构

FPGA 器件主要包括以下几个功能模块[8]：

（1）可编程输入输出单元（IOB）：FPGA 芯片与外部电路的接口；

（2）可配置逻辑块（CLB）：基本逻辑单元，主要由 Slice 组成，可实现逻辑，可配置为存储器。

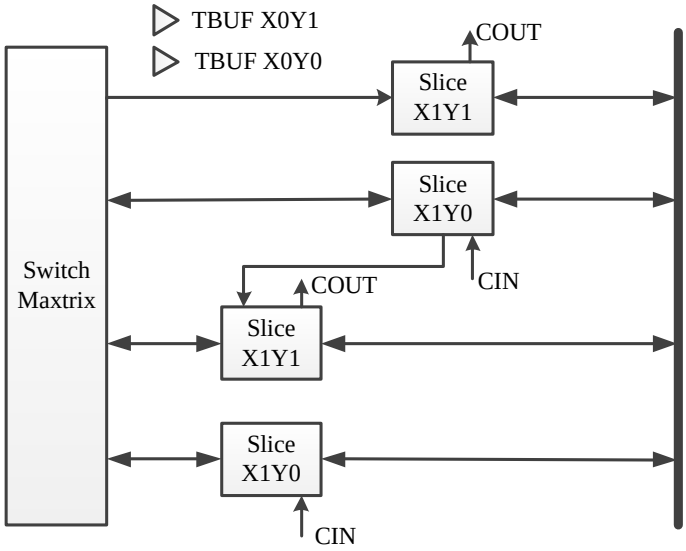



图2. 2　　CLB 结构示意图

（3）嵌入式块 RAM（BRAM）：配置为不同类型的存储器；

（4）布线资源：连接 FPGA 器件内部的所有单元，连接方式决定信号的性能；

（5）底层内嵌功能单元：FPGA 内部的一些软核，主要包括 DLL、PLL、DSP 和 CPU 等。

（6）内嵌专用硬核：FPGA 处理能力较强的硬核，比如专用乘法器。

## 2.2 FPGA 的工作原理

### 2.2.1　FPGA 工作原理

对于一个 n 输入的数字逻辑电路，输出最多有 $2^n$ 种不同的结果。查找表（Look Up Table，LUT）类似于一个存储器，用于存储逻辑电路分所有可能输出结果，输入变量相当于存储器的地址，根据地址索引存储器中相应地址内容，作为逻辑运算结果输出[9]。如图 2.3 所示。

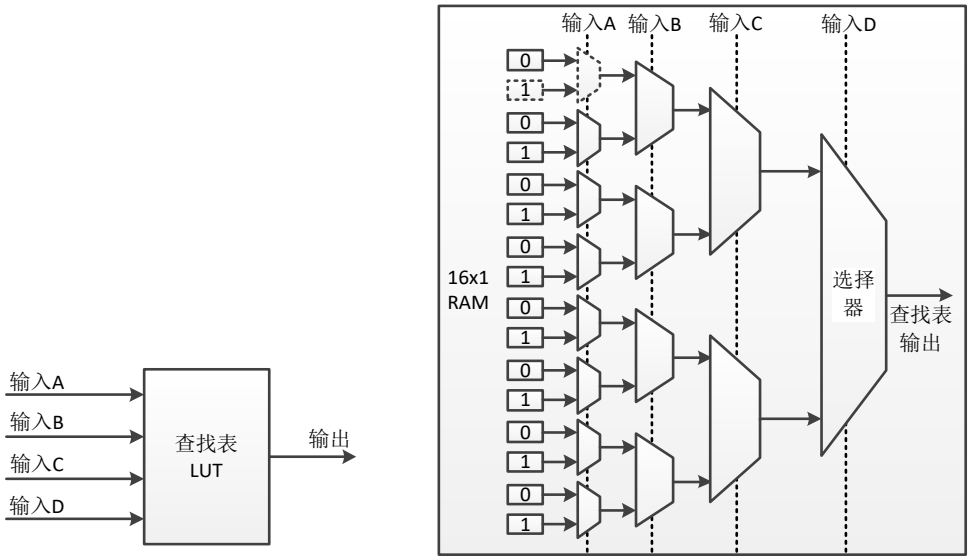



图2.3 LUT 内部结构

### 2.2.2 编程方式

首先，用户将与 FPGA 的工作状态有关的配置信息烧写入 RAM，然后 FPGA 通过读取到的信息来决定其工作状态。烧写配置信息的过程称为编程方式有以下几种：

（1）并行模式：一片 PROM 或者 Flash 存储器配置一块 FPGA；

（2）主从模式：一片 PROM 控制多块 FPGA，对其进行配置编程；

（3）串行模式：串行 PROM 配置 FPGA；

（4）外设模式：FPGA 作为外设与微处理器连接，由微处理器对 FPGA 进行配置。

生产 FPGA 器件的两大厂商 Xilinx 和 Altera 采用的工艺都是基于静态随机读取存储器的，在使用时，首先上电后，FPGA 读取并加载预先存储在外部存储器中的配置信息，然后处于正常工作状态，对系统断电后所有的配置信息被擦除，因此可以重复配置，灵活性较强[10]。

## 2.3 太空辐照效应

深空探测应用中，单粒子效应（Single Event Effect，SEE）对 SRAM 型 FPGA 影响较大[11]。

### 2.3.1 单粒子效应

单粒子效应是指单个高能粒子，在电子元件灵敏区电离，使带电粒子数量发生暂时改变，进而影响电子元件正常工作状态的一种辐射现象。常见的单粒子效应有以下几种[12]：单粒子翻转、单粒子锁定、单粒子烧毁、单粒子功能中断和单粒子瞬态脉冲

等[15]-[19]。

相对来说，单粒子翻转是空间辐照现象中最为常见的一种。指的是高能粒子在电子元器件发生碰撞，运动轨迹附近的电荷被电极捕获形成瞬态电流，当该电流达到一定的门限值就会触发逻辑电路状态翻转[13]。以 CMOS 反相器为例进行如下说明[14]。

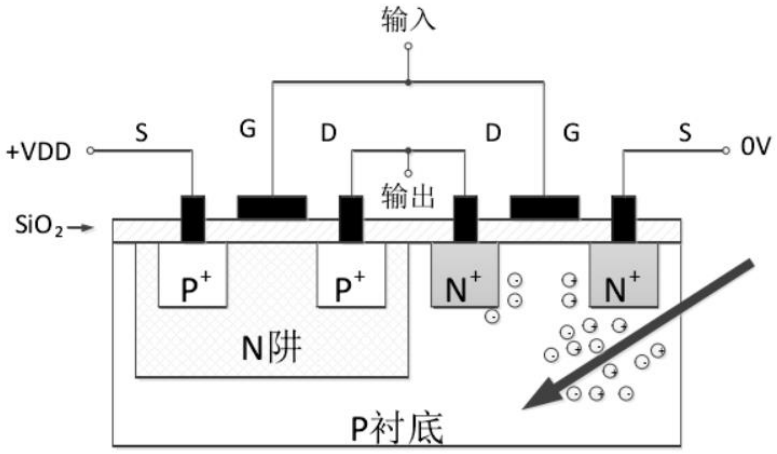



图2. 4　　CMOS 反相器的单粒子翻转图示

COMS 反相器结构如图所示，当输入为低电平时，参考反相器工作原理，输出端为高电平。单粒子照射后，高能粒子入射到反相器的 P 衬底，并在运动轨迹周围产生电子空穴对。在内部电场的作用下，带正电的空穴移动到 NMOS 晶体管的源极，而带负电的电子则会移动到晶体管漏极。电荷数量累计到一定数量时，CMOS 漏极电压变低，使得 CMOS 反相器的输出端由高电压变为低电压，造成输出状态错误。这样最终导致后续电路逻辑状态异常。

照射结束后，输入端仍是低电平，此时 NMOS 晶体管断开，PMOS 晶体管导通，反相器输出恢复为高电平状态。

### 2.3.2　SRAM 型 FPGA 的单粒子效应

本系统是基于 Xlinx 公司的 Virtex2 系列 FPGA 芯片，属于 SRAM 类型。对本设计中用到的器件影响较大的是单粒子翻转和单粒子锁定。

根据电子元器件以及单粒子锁定发生的机理分析可知：单粒子锁定属于硬错误，单纯的通过采取一些逻辑电路设计方法也是无法避免的。然而单粒子翻转却不同，具体分析如下：

通过前文可知，SRAM 型 FPGA 的逻辑电路是通过查找表来实现的，而查找表是容易遭受到单粒子翻转的影响的，根据查找表的结构可知，查找表主要是对逻辑电路的状态进行存储与读取，通常来说，查找表结构实现的逻辑运算往往与控制信号有

关，在电路正常工作过程中，查找表的部分单元一旦发生翻转，使得原始的逻辑电路输出结果发生异常，这将导致整个后续电路的状态故障。

## 2.4 单粒子效应的应对措施

（1）重新上电配置

FPGA 器件在加电时会对内部模块进行内部初始化，并且根据配置信息重新进行配置。重新上电后的系统处于初始状态，是最简单的一种消除异常错误的方法[15]。

（2）系统监控与配置刷新

由高可靠性的反熔丝 FPGA 负责从配置存储器中读取 XlinxFPGA 的配置数据信息，与标准的配置数据进行对比校验，对出现错误的数据按照位置信息重新配置。可分为两种模式：回读检测刷新和定时刷新[20]。

a）回读检测刷新：通过读取配置存储器中的配置信息，并检测其是否发生故障。如果发现回读数据中存在 bit 错误时，对对应的 bit 位进行刷新重配置操作，将正确的数据 bit 写入配置存储器中。

b）定时刷新：以一定的时间周期向配置存储器载入整个 CLB 配置段，实现定时刷新。时间周期的选择是根据每个器件发生单粒子翻转的频率设定的。

（3）减少 Half-Latch

在 Virtex 系列 FPGA 中存在 Half-Latch 结构。Half-Latch 的作用是生成用户设计中未显式指定的逻辑常量，对太空辐射敏感。在对 FPGA 上电配置时，对 Half-Latch 进行统一初始化,不受配置信息控制，这意味着当其发生翻转后,除非终止系统工作并进行重新配置,否则无法修复 Half-Latch 翻转导致的故障[21]。根据 Half-Latch 结构原理和作用可知，是对电路设计中未指定的常数进行赋值，所以，只要在电路设计环节，不遗漏对常数的指定和赋值，就能避免 Half-Latch 结构的使用，从而避免该部分的翻转。

（4）三模冗余设计

三模冗余 (Triple Modular Redundancy，TMR)技术，是最常用的一种抗单粒子效应的容错措施。其设计方法是：三个完全相同的电路模块互为备份，同一时刻对同一个任务或者功能执行同一操作，之后将三个模块的输出结果进行三选二的少数服从多数选择，选择的结果即为电路最终的输出。如果三个电路模块中有任意一个发生单粒子故障，电路状态翻转，判决后的最终输出为正常的状态[22]。由于三个模块是互相独立的，两个模块同时出现错误是极小概率事件，故可以大大提高系统的可靠性。同时，三模冗余设计也造成了硬件资源使用的 3 倍增长，增加了系统的功耗，因此可以根据功能性能的重要程度对系统进行部分三模冗余处理，这样保障了系统可靠性的同时，

节省了资源和降低了功耗。

（5）设备器件冗余

与三模冗余技术相同，从宏观上对设备器件实现三模指的是在多片 FPGA 上实现配置管理和设计，并通过对输出结果的比较判断来决定最终的正确输出，比如两块或者三块 FPGA 的实现冗余设计。对于 XX-XX 图像与数据处理 FPGA 项目中的两块 FPGA 实现方案，每一块 FPGA 上包含同一个设计的两个备份（主份和备份）在输出端链接抗辐射设备用于判决输出，能够有效的保证系统正常可靠地工作，这样的器件冗余设计方案会导致开发成本数倍增长。

## 2.5 本章小结

整体上来说，本章从 FPGA 的结构和工作原理出发，分析太空辐照现象以及对 SRAM 型 FPGA 的影响，并且根据这些影响的性质和类别介绍了星载 FPGA 系统应对太空辐照的措施。介绍了几种常用的处理方法，根据项目背景详细介绍了三模冗余机制和器件冗余的方法。

# 第三章　JPEG-LS 图像压缩算法的研究和实现

## 3.1 JPEG-LS 图像压缩算法

JPEG-LS 图像压缩算法能实现的是对静态图像进行无损/近无损压缩。如果从压缩码流中恢复的重建图像与原图像完全相同，则是无损压缩，否则叫做有损压缩，当两者误差限定在一定的范围时叫做近无损压缩[22]。

JPEG-LS 图像压缩算法的结构框图如图 3. 1 所示。

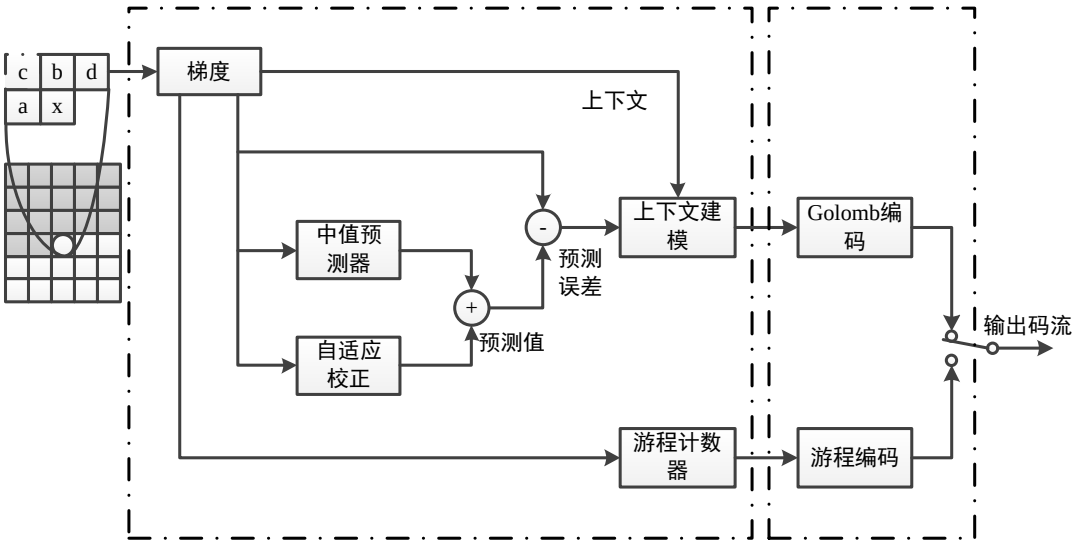



图3. 1　　JPEG-LS 图像压缩算法结构框图

上图中，通过预测，上下文建模和 Golomb 编码过程实现的是普通压缩，是对输入图像进行近无损压缩；如果图像中有大块区域像素灰度比较平滑，则可以通过游程模式进行压缩。具体选择哪种编码模式，是根据上下文信息来确定。该算法对图像压缩过程中，不需要对图像进行离散余弦变换（DCT）和算数编码等，只进行预测和 Golomb-Rice 编码，算法复杂度较低。普通模式下编码步骤为[23]：

- 求当前像素的上下文参数；
- 根据上下文模板中的相邻像素值进行预测，得到当前像素的预测值，以上述上下文参数修正预测值；
- 利用预测值与原像素得到预测误差，对预测误差进行修正编码；
- 更新上下文的相关参数；
- 对预测残差进行 Golomb-Rice 编码。

### 3.1.2 上下文建模

上下文建模是指通过像素间的梯度，得到一个索引参数。包括梯度计算、梯度量化和求索引值三部分。

（1）计算梯度值

利用当前像素相邻进的几个像素来计算三个梯度值。

$$D_1 = x_d - x_b, D_2 = x_b - x_c, D_3 = x_c - x_a \tag{3-1}$$

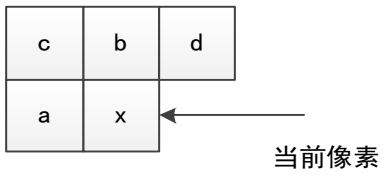



图3.2 梯度计算像素关系

（2）量化梯度值

根据步骤（1）计算得到的三个梯度值与门限参数的关系，得到 5 个索引 $Q_i$，计算方法如式（3-2）。$T_1$，$T_2$，$T_3$ 为门限参数值。

$$Q_i = sign(D_i)\begin{cases} 0, & D_i = 0 \\ 1, & 0 < |D_i| < T_1 \\ 2, & T_1 \le |D_i| < T_2 \\ 3, & T_2 \le |D_i| < T_3 \\ 4, & T_3 \le |D_i| \end{cases} \tag{3-2}$$

（3）计算上下文索引值

对上一步骤计算出的量化值 $Q_i$ 的绝对值按照下式计算，得出 Q 值，Q 就是上下文索引。首先是对量化后的梯度 $Q_i$ 进行合并处理。步骤（2）得到的三个索引组成一个矢量$\{Q_1,Q_2,Q_3\}$，其第一个非零元素如果是小于零，则对该矢量取反，变换为$\{-Q_1,-Q_2,-Q_3\}$，同时符号参数 sign 也取-1，否则，sign 取+1。然后通过式（3-3）计算得到当前像素的最终上下文索引参数 Q。

$$Q = \begin{cases} 360 + Q_3, & Q_1 = 0, Q_2 = 0 \\ 324 + (Q_2 - 1) \times 9 + (Q_3 + 4), & Q_1 = 0, Q_2 \ne 0 \\ (Q_1 - 1) \times 81 + (Q_2 + 4) \times 9 + (Q_3 + 4), & Q_1 \ne 0 \end{cases} \tag{3-3}$$

### 3.1.3 预测

JPEG-LS 图像压缩算法的预测部分是进行二维预测，即用当前像素同一扫描行的前几个像素和上一行的邻近像素来预测。对像素的预测包括边缘检测、预测值修正。

（1）边缘检测

JEPG-LS 采用中值预测器(Median Edge Detector, MED)计算当前样本像素 $I_x$ 的预测值 $P_x$。利用邻近像素的重建值来预测当前像素的预测值。形式如下：

$$P_x=\begin{cases}\min(R_a,R_b), & R_c\geq\max(R_a,R_b)\\ \max(R_a,R_b), & R_c\leq\min(R_a,R_b)\\ R_a+R_b-R_c, & else\end{cases} \tag{3-4}$$

其中 $R_a$，$R_b$，$R_c$ 和 $R_d$ 是上下文模板中的像素值的重建值。当重建值取 $R_b$ 时，表示待编码像素左侧存在垂直边缘，当预测值取 $R_a$ 时，表示待编码像素上方存在水平边缘，当预测值取 $R_a+R_b-R_c$ 时，表示没有检测到边缘。中值预测器操作简单，只涉及加减比较运算，但是在图像高频细节部分会产生较大的预测误差。

（2）预测修正

对预测值修正的目的是为了减少处理过程中的误差。对预测值进行修正用到下述方法。

```
if (Sign==1)
    Px+=C[Q];
else
    Px-=C[Q];
if(Px>MaxVal)
    Px=MaxVal;
else if(Px<0)
    Px=0;
```

利用梯度符号、修正计数器和 Maxval 三个参数对像素预测值进行修正。其中 Sign 是梯度符号；C[Q]预测矫正值；Maxval 为输入图像像素的最大值，对于 8bit 图像，其值为 $2^8$-1。

### 3.1.4 预测残差计算

预测残差是样本像素值与其预测值的差值如式（3-5）。

$$Errval=Ix-Px \tag{3-5}$$

$I_x$是当前像素的样本值，如果符号参数 sign 的值为负，则对预测残差取反，如式（3-6）；否则，无需对预测残差做任何处理。

$$Errval = -Errval \tag{3-6}$$

如果采用的是无损模式，则可以直接对预测残差进行编码；如果采用的是近无损模式，则还需要对预测残差进行量化，以 2Near+1 大小的量化间隔对预测残差进行均匀量化如式（3-7）。其中参数 Near 叫做量化失真参数，是图像像素样本值与恢复后的图像对应像素之间的可接受的最大绝对差值。

$$Q(Errval) = sign(Errval)\left\lfloor \frac{|Errval| + Near}{2Near + 1} \right\rfloor \tag{3-7}$$

此时，量化后的预测残差的符号可正可负，近似服从拉普拉斯分布，而几何分布的 Golomb-Rice 编码效果比较理想，故需要对预测残差分布变换为几何分布，方法如下：

$$MErrval = \begin{cases} 2Errval, & Errval \geq 0 \\ -2Errval - 1, & Errval < 0 \end{cases} \tag{3-8}$$

### 3.1.5 上下文参数更新

在 JEPG-LS 算法中对每一个上下文用到了 4 个计数器，用于保存当前上下文相关的信息。分别是：

A[Q]——每个上下文预测误差的绝对值和，用来估计编码参数 k。不同比特深度的图像对应的初始值不同。

B[Q]——每个上下文预测误差和（有符号）；初始值为 0；

C[Q]——预测修正值，用于矫正像素的预测值。初始值是 0；

N[Q]——每个上下文发生的次数，初始值是 1。

对每一个像素编码完成后都需要对上下文参数进行更新，保证在下一个像素编码时，所有的寄存器状态都与当前像素匹配。更新按照以下步骤顺序进行。

首先，

```
A[Q] = A[Q] + abs(Errval);
B[Q] = B[Q] + Errval*(2*Near + 1);
```

```
if(N[Q] == RESET){
        A[Q]>>= 1 ; B[Q] >>= 1 ; C[Q] >>= 1 ;
}
N[Q]++ ;
if(B[Q]<= -N[Q]){
        B[Q] += N[Q];
        if(C[Q] > MIN_C)
                C[Q]--  ;
        if(B[Q] <= -N[Q])
                B[Q] = 1-N[Q];
}
else if( B[Q] >0) {
        B[Q] -= N[Q];
        if(C[Q] < MAX_C) C[Q]++ ;
        if(B[Q] > 0) B[Q] = 0;}
```

### 3.1.6 Golomb-Rice 编码

Golomb 编码只能对非负整数进行编码，当编码字符服从几何分布时，使用可以取得最优效果。较大数字出现的概率远小于较小数字，用较短的码长编码较小的数字，较长的码长编码较大的数字[24]。

以正整数 m 为参数，对 n 进行 Golomb 编码，分为两部分：n%m 的二进制码和 n/m 的一元码。对几何分布的被编码变量，总存在一个 m 使得编码后的平均码长最短。

当 m 的取值为 $2^k$ 时的 Golomb 编码叫做 Golomb−Rice 编码。此时，对 n 的编码就可以表示为：（n/2 的一元码表示）加上 n 的低 k 位

参数 k 的确定：

假设被编码变量服从几何分布，那么通过近似估计可以得到参数 k 为：

$$k = \min\{k' \mid 2^{k'} \times N \geq A\} \tag{3-9}$$

在实现时，可以按照以下方式求出：

$$for(k = 0\ ;\ (N \ll k) < A\ ;\ k++) \tag{3-10}$$

其中，A 和 N 是上下文参数寄存器。

### 3.1.7 游程编码

在编码过程中当前像素 $I_x$ 与几个邻近像素点的偏差小于失真量化参数 Near 时，

就进行游程编码。基本原理是，对字符串用重复的字符和重复的次数表示，这将会使信源符号的长度小于原始数据的长度，从而实现数据的压缩[25]。比如信源符号串为：

11111222233333455555

对应的游程编码为：（1, 5）（2, 4）（3, 5）（4, 1）（5, 6）。

在无损压缩模式时，编码器必须读取后续的样本，只有这些样本值 $I_x$ 等于进入游程模式时的第一个重建值 $R_a$，游程长度会持续累加，知道发现样本值与重建值不同或差值过大，或者到达行尾。在近无损压缩时，如果读入的样值 $I_x$ 与 $R_a$ 不超过 Near，游程长度也会继续累加，当游程遇到中断样本（当前样本值与第一个重建值不同）而终止时，还必须对中断样本进行编码。

## 3.2 JPEG-LS 图像压缩算法的硬件实现

### 3.2.1 算法实现难点问题

JPEG−LS 静态图像压缩编码算法在无损压缩模式下硬件实现较为简单，编码器直接接收相机传来的图像数据，按照流水线的方式依次对每个像素进行预测编码，不需要对图像数据进行缓存，处理速度高。而在普通压缩模式下，在硬件实现上存在以下困难：

（1）流水线处理的实现

每个像素点在进入编码器的第一个时钟周期就要用到上一个像素点的重建值，而像素重建值的计算十分复杂，要经过多个时钟周期才能计算得出，所以导致每个像素点要等待上一个像素点重建值计算完成后才能开始进入编码器进行编码。这样就使得模块之间无法同时工作，即同一时刻大部分模块都处于空闲状态，硬件资源没有充分利用，编码器处理速度大大降低。

（2）像素缓存

由于编码器处理速度大大降低，处理速度跟不上图像输入速度，所以需要增加片外存储将存储图像缓存下来，这样的方式增加了硬件实现的复杂度以及系统功耗，同时失去了 JPEG−LS 能够实时编码处理的优点。

（3）压缩过程中的乘除运算

计算像素重建值的过程过于复杂，连续用到了大位宽的除法器和乘法器，相比原来无损算法中只有简单的加减运算，运算量成倍增加，逻辑时延加长，编码器的工作时钟频率大大降低，这会进一步降低编码器的处理效率，硬件实现上难度加大。

### 3.2.2 算法实现结构模块化设计

算法硬件实现过程中，需要对当前像素的邻近像素进行读取，包括上一行相邻的

像素和当前行相邻的像素，因此，需要对图像进行缓存。但是，缓存整副图像所需要的存储器资源过大，因此，只对图像上一行像素进行缓存，在整个预测过程中用到的像素值是像素的重建值，故编码结束后，对当前像素重建值进行存储。对于图像上特殊位置的像素（部分边缘像素）进行特殊处理。编码过程中，有多个参数是根据图像确定的，本设计的输入图像源大小为 1072*1028*16bit，这样一些参数可以确定，不再进行计算，减少了系统的资源开销，这些参数包括：量化梯度的门限（$T_1, T_2, T_3$）、量化失真参数（Near）、MAX_C、MIN_C、limit、Range 和 qbpp 等。

JPEG-LS 图像压缩算法的硬件实现分为 3 个模块，第一个模块主要进行像素缓存与读取；第二个模块进行像素预测，并得出待编码的预测残差；最后一个模块进行编码输出。如下图所示。

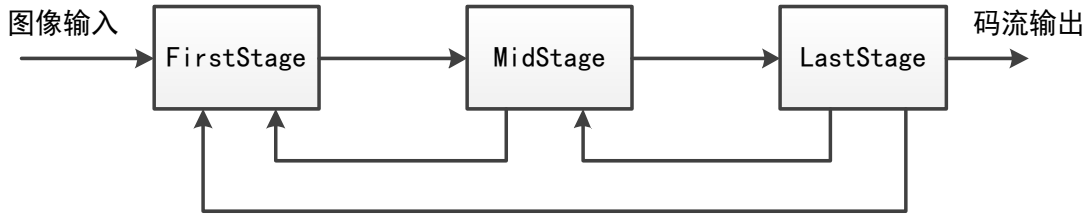



图3.3　　JPEG-LS 模块化实现结构

下面将对每一个模块进行详细介绍。

（1）FirstStage 模块设计

在本模块主要功能是获取当前像素邻近的 4 个像素，同时对输入的当前像素进行处理，为下一步计算做准备，模块化实现框图如下：

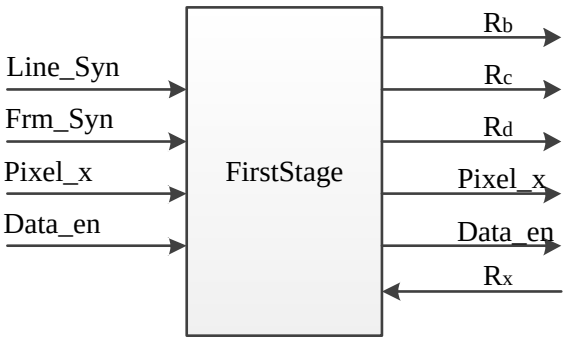



图3.4　　FirstStage 模块

在进行无损压缩时，在本模块直接将输入像素值打拍后输入双端口 RAM，作为行缓存数据，行缓存模块将产生 $R_a, R_b, R_c, R_d$ 四个值作为预测模块的输入，其中 $R_a$ 是上一个输入的像素点 $I_x$；进行有损压缩时，RAM 存储器接收来自中间级的像素重

建值。

（2）MidStage 模块设计

中间级模块的功能是计算像素的预测误差和重建值，并计算 Golomb-Rice 编码参数 k 和 Merrval。其中预测误差是作为编码模块的输入；重建值存储在行缓存模块中。模块结构如下图。

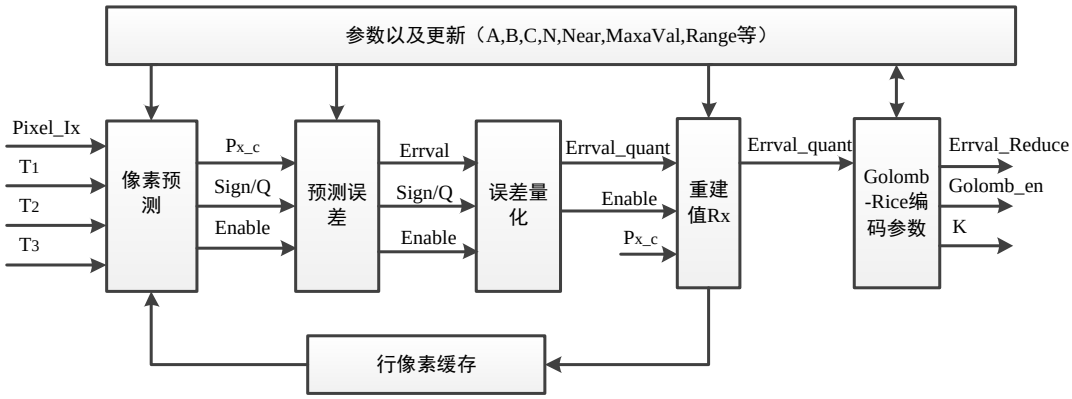



图3.5　　MidStage 模块

编码参数根据算法实现流程对本模块进行进一步划分。

a）求预测值和上下文索引模块

计算预测值：以边缘检测的方法，利用行缓存存储器中的像素重建值就能够计算出当前像素的预测值；

同样地，按照算法原理，首先计算像素间的梯度，接着对梯度值进行量化，最后求出上下文索引值 Q。

计算梯度值是两个相邻像素进行相减运算，但实际上相邻像素的大小关系并不确定，相减的结果符号未知，因此利用硬件语言实现时，增加 1bit 的高位，作为像素的符号位，相减结果的最高位同样是符号位，这样就有利于后续运算的判断和实现。

对梯度值的量化，是根据量化的几个门限参数（$T_1, T_2, T_3$）确定的，门限参数是由 Near 确定，Near 值与压缩比直接相关。在实际的 XX-XX 图像与数据处理 FPGA 项目应用中，压缩模式下涉及到的压缩比取值为 2:1, 4:1 和 8:1，对应的 Near 值分 0, 3 和 11，其中 2:1 为无损压缩，4:1 和 8:1 是近无损压缩。

计算上下文索引：首先求出符号参数 Sign，在梯度融合后的基础上，利用矢量{$Q_1$，$Q_2$，$Q_3$}求出索引值。实现时，将乘法运算拆分为移位并位操作，很大程度上降低了运算的复杂度。比如（假设 $Q_i$ 为 4 位）：

$$9\times Q_i = \{Q_i, 3'b000\} + \{4'b0000, Q_i\} \tag{3-11}$$

b）预测误差

本模块包括预测误差的计算和对其进行量化两个过程。

- 预测值修正

在求预测误差前，首先对预测值进行修正，根据符号 Sign 和修正参数 C[Q]对预测值 $P_x$ 得到 $P_{x\_C}$。

- 预测误差

预测误差是像素预测值与原像素值之间的差值，符号与 Sign 有关，可正可负，用 Errval 表示。

- 量化预测误差

对预测误差量化的计算公式如下，用 Errval_quant 表示：

$$\frac{Errval}{2Near+1}=Errval\times\frac{2^{20}}{2Near+1}\times\frac{1}{2^{20}} \tag{3-12}$$

在设计量化模块时，根据以上变换，将除法操作转换为乘法来实现。对于数字电路，通过左右移位来实现对操作数的乘除运算，为了方便操作取，移位量取 20，用 Shift 表示，其中的中间值 $\frac{2^{20}}{2Near+1}$ 用 multi 表示。

- 重建值

计算像素的重建值 $R_x$，并对结果进行换缓存。

c）Golomb-Rice 编码参数

本模块的功能包括以下三点：

- 计算 k 值；
- 对 ErrVal_Q 进行限幅：判断 ErrVal_Q 是否在[-range/2, range/2]之间，如果不在，则相应的加上一个 range 或者减去一个 range。限幅后得到 ErrVal_reduce；
- 上下文参数更新：根据 ErrVal_quant 来更新参数 B[Q]，根据 ErrVal_reduce 来更新参数 A[Q]。然后再根据 B[Q]和 N[Q]来更新参数 C[Q]。

前端行缓存模块完成的功能是按行依次接收每个像素点 $I_x$，然后将 $I_x$ 以及其相邻像素点 a，b，c，d 一起送给中间预测模块处理，设置双端口 RAM 用以缓存上一行的像素值。这样每次接收到 $I_x$ 后，只需从 RAM 中读取 d，再将上一个像素值的 b，c，x，分别赋予本像素值的 c，b，a，即可形成本次输出 a，b，c，d，x。但要注意的是在行开头和行结尾以及第一行处要特殊处理：行开头时，从 RAM 中读取 b，d，将 b 赋给 a，令 c 为 0。行结尾时，将上次的 d 赋给本次的 b 和 d。第一行时，将 b，c，d 全赋值为 0。每次进入游程模式时，都需将进入游程时的 d 保存下来，当退出游程时将这个 d 作为当前像素点的 b 值打出。

（3）LastStage 模块设计

本模块结构化为如下图所示。

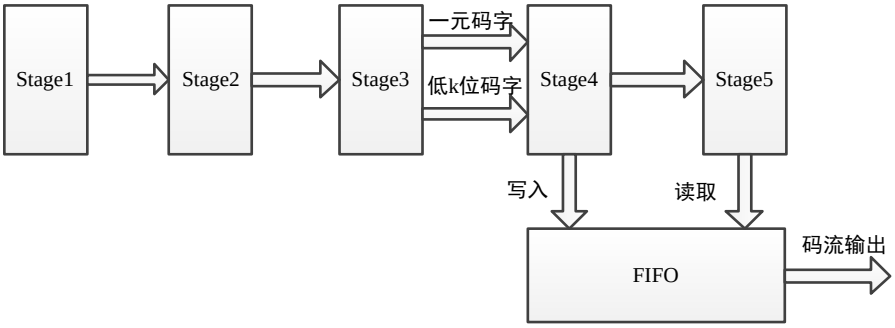



图3.6　　LastStage 模块结构

本模块主要负责对量化误差进行 Golomb-Rice 编码，前端模块的输出参数作为本模块的输入信号，进行编码，同时对输出码流进行组织输出。根据算法编码原则，编码过程分为两部分，一元码部分和低 k 位。最终将 64bit 编码数据写入 FIFO，再每 16 位输出。对两部分编码进行组织时，对于一元码，是非常整齐有规律的，故将低 k 位的编码变化为与一元码相同形式的码字，这样很大程度上简化了编码组织的复杂度。最终只需要明确每部分码字长度，通过桶型结构即可确定合并后的码字。以 128 位的移位寄存器从高位到低位以此存储接受的码字，每到达 64bit，将高 64 位作为写入 FIFO 的码字。

### 3.2.3　关键性问题

在本设计中涉及到的几个关键技术点包括：行像素缓存，行起止时序关系以及算法模式的选择问题。

（1）像素缓存

前端行缓存模块完成的功能顺序存储图像每行的像素 $I_x$，然后将当前像素以及所需的几个像素 a，b，c 和 d 传递到下一级进行处理。设置双端口 RAM 用与缓存上一行的像素值。这样每次接收到 $I_x$ 后，只需从 RAM 中读取 d，再将上一个像素值的 b，c，x，分别赋予本像素值的 c，b，a，即可形成本次输出 a，b，c，d，x。但要注意的是在行开头和行结尾以及第一行处要特殊处理：行开头时，从 RAM 中读取 b，d，将 b 赋给 a，令 c 为 0。行结尾时，将上次的 d 赋给本次的 b 和 d。第一行时，将 b，c，d 全赋值为 0。

每次进入游程模式时，都需将进入游程时的 d 保存下来，当退出游程时将这个 d 作为当前像素点的 b 值打出。

（2）行起始时序设计

将行同步信号 syn 打一拍后得到 syn_1d，通过比较 syn 和 syn_1d 检测到行同步信号的上升沿，用 first 信号表示，在检测到 first 信号为 1 时，立即送给 RAM 一个读使能有效信号，从 RAM 中读取 b。在检测到 data_en 有效时，也送给 RAM 一个读使能信号，从 RAM 中读取 d

对于行开头像素点，需要从 RAM 中依次读取 b 和 d，然后和第一个像素点 Ix 一起输出，由于读取 b 的读使能信号是由 first 产生的，d 的读使能信号是由 data_en 信号产生的，为了确保先读出 b，后读出 d，必须将 data_en 信号打两拍，由 data_en_2d 触发产生 d 的读使能信号，这样，在 Ix 传入本级模块四个时钟周期后，模块会得到从 RAM 中传来的相应的 d，再将 d 打一拍存入 d_reg 中，故需要将 Ix 打五拍，将 Ix_5d 与 b，d 做组合逻辑运算后得到相应的 a，b，c，d，x 输出。

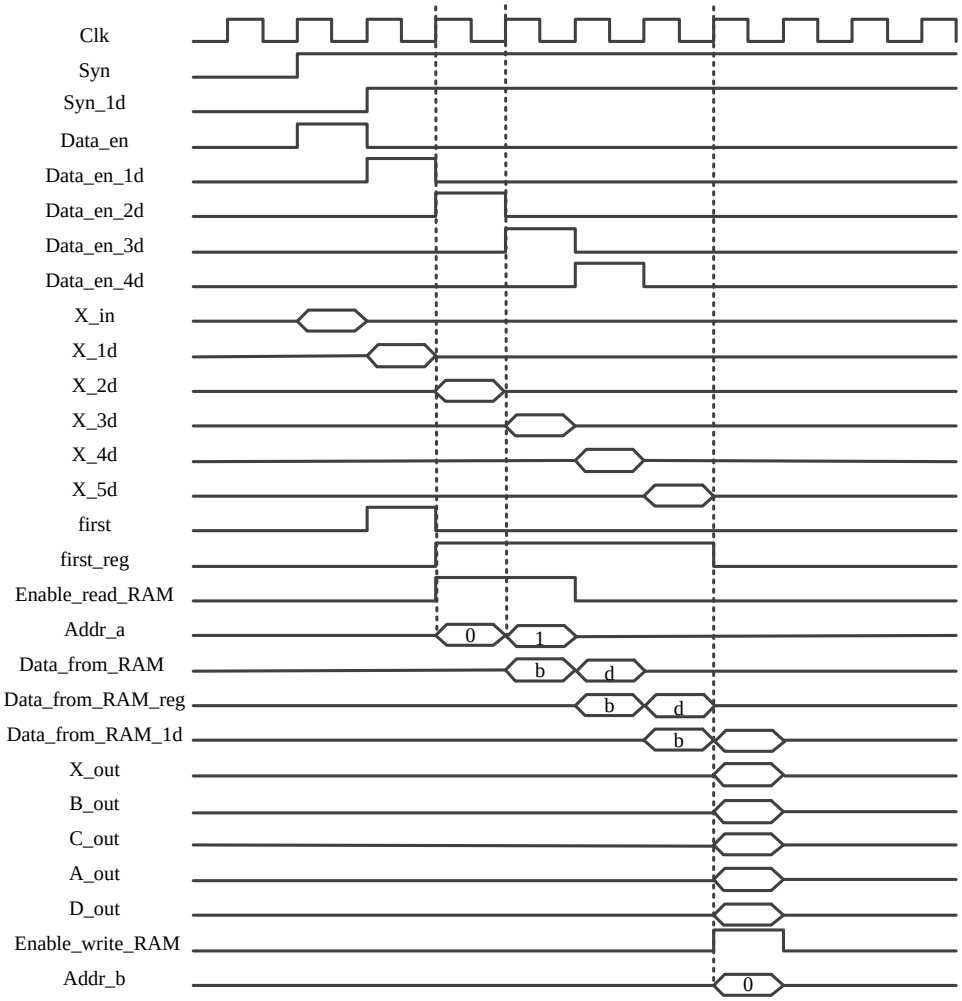



图3.7　　行起始时序关系

（3）行结尾时序设计

在行结尾处，为了判断最后一个像素点，我们规定每行最后一个像素点与行同步

信号结尾处对齐，我们设置 last 信号来判断行结尾，为了使最后一个像素点的输出时刻能判断出其为最后一个像素点，将 last 信号打三拍产生 last_3d，在最后一个像素点输出时刻，由 last_3d 判断其是否为本行最后一个像素点，若是则按最后一个像素点的处理方法处理后输出。

由于下一行开头时会对 RAM 的读写地址清零，这会造成本行正在往 RAM 中写入的像素点发生错误，为了避免这一点，必须保证行间距不得小于 5。

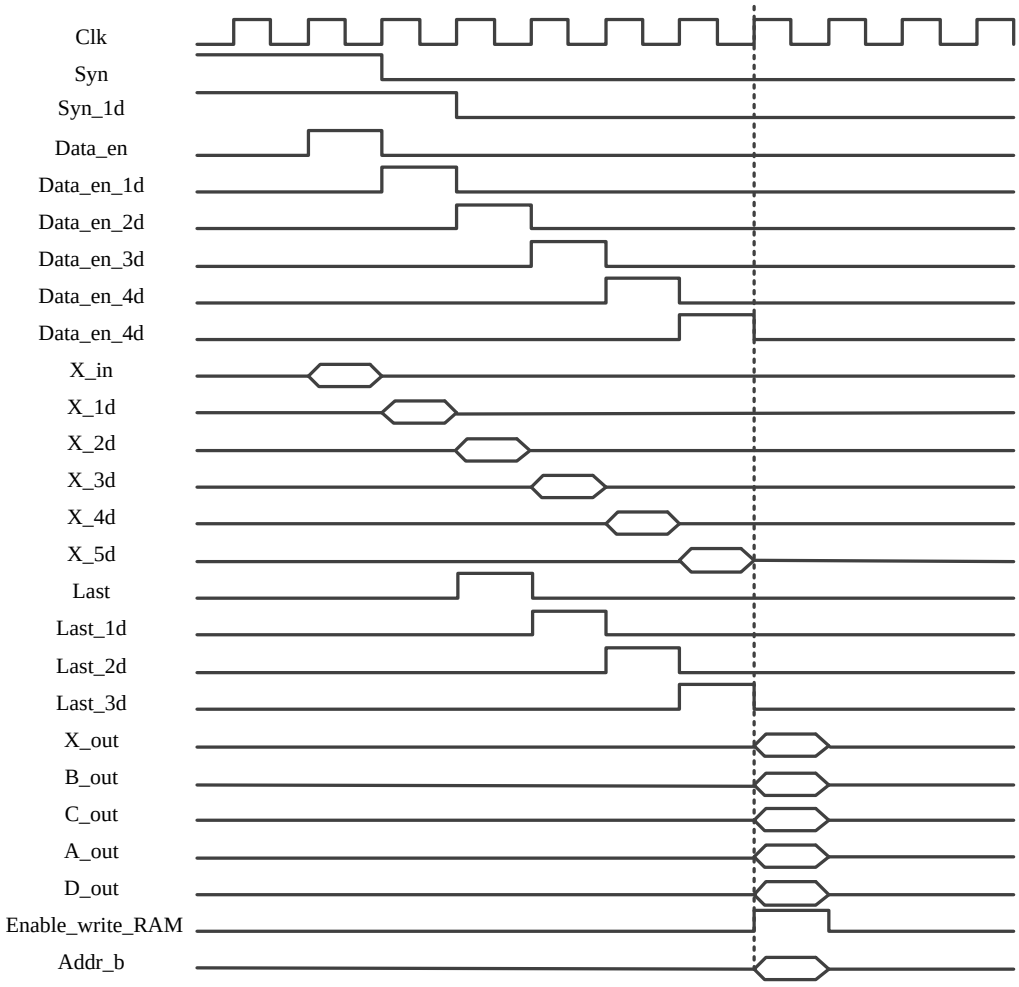



图3.8　　行结束时序关系

（4）游程模式控制

当判断级发现前端送来的 abcdx 相等时，便进入游程模式，同时送给前端一个 in_runmode 信号，标志处于游程模式处理状态，这个信号一直持续到游程结束，当前端发现 in_runmode 信号有上升沿时，便将进入游程时的 d 值存储在 d_reg 之中保存下来，当退出游程时将 d_reg 赋给进入常规模式的第一个点的 b 值。

在间断输入数据情况下 in_runmode 信号只能持续到游程结束，但是当下一个点

到来时 in_runmode 信号可能早已拉低，这样就无法判断之前处于游程之中，所以重新生成了 in_runmode_reg 信号，一直持续到下一个点到来，这样就能判断到之前处于游程模式之中，从而将 d_reg 赋给将要输出的 b 值。

在游程模式下，需要判断当上一个点的 a 不等于 x，原来的做法是判断上上个 x 和上个 x 是否相等，但是在间断输入数据下无法得知上上个 x 的值，于是将这种做法改为：每次都判断当前输出的 a 和 x 是否相等，产生一个 a_equal_x 信号，这个信号会一直延续到下一个点的输出时刻，以此判断。

## 3.3 本章小结

本章从 JPEG−LS 图像压缩算法的原理出发，详细介绍了算法的处理流程，最主要的就是预测，上下文建模和 Golomb−Rice 编码；在算法原理的基础上，进行了算法的 FPGA 实现，实现过程与算法的处理步骤基本一致。除此之外，对短发实现过程中的几个重要的难点问题进行了分析说明，包括图像像素数据的行缓存方法，流水线设计的实现和算法接口的行起止时序设计。

# 第四章　容错技术与可靠性设计

## 4.1 三模冗余

三模冗余（Triple Modular Redundancy，TMR）是对电路功能模块进行复制，三个完全相同的模块互为备份，同时执行同一操作，对执行完成的三个结果进行三选二的少数服从多数判决，得到最终的输出结果[21]。当三个模块中的任意一个发生故障状态翻转，判决输出为正确的状态。由于三个模块是互相独立的，两个模块同时出现错误是极小概率事件，故可以大大提高系统的可靠性。

输入为三变量的多数选择器的逻辑关系可以表示 TMR 的判决输出逻辑关系，其真值表如下所示。

表4.1 三选二电路真值表

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

根据真值表可以得出多数选择器的逻辑函数表达式为：

$$F = \overline{\overline{AB} \cdot \overline{BC} \cdot \overline{AC}} = AB + BC + AC \tag{4-1}$$

### 4.1.2　传统的三模冗余设计

传统的三模冗余设计方法是用对所有寄存器和功能模块增加两个额外的备份，一共三模，使用多数选择器对三模的输出进行判决，得到模块的最终输出[26]。理论上来说，当三模中的任何一个模块中发生错误翻转，其他两个模块正常工作，多数选择器将正确的结果选择输出，如下图所示。

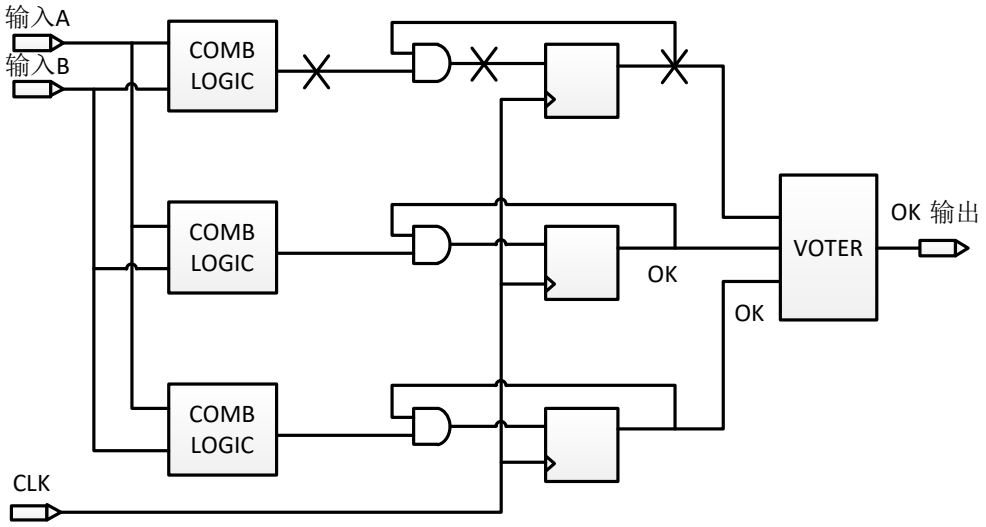



图4.1　　传统的 TMR 实现

传统的 TMR 实现方法应用于可重配置 FPGA 中时存在两个缺陷[26]：

（1）传统的 TMR 设计只是对功能模块进行了三模设计，判决电路的多数选择器仍然不能避免遭受单粒子影响，可能会发生翻转，这样即便是进行多数选择判决，其输出结果仍会发生错误。但是基于 0.25μm 以下工艺尺寸的 CMOS 器件是没办法避免单粒子瞬态效应(SET)的，如下图所示。

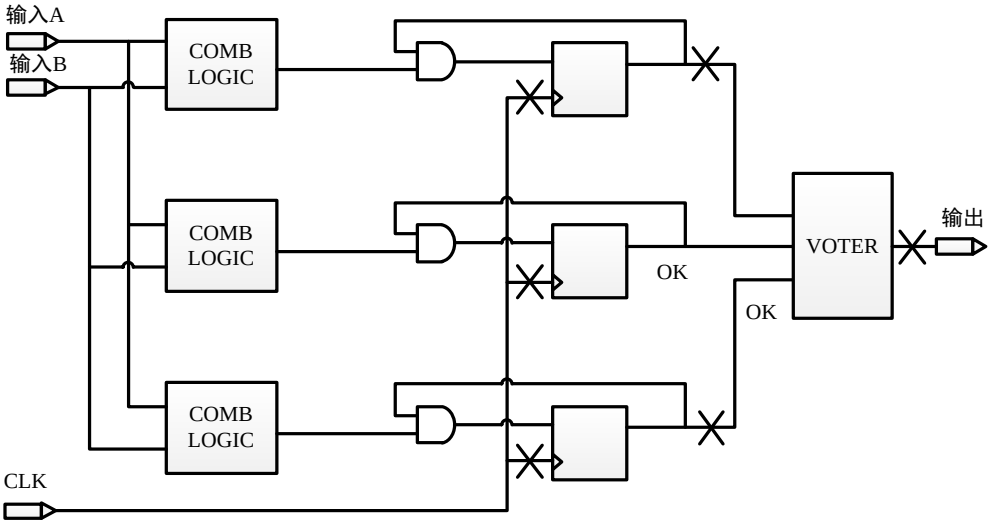



图4.2　　传统的三模 SET 示意图

（2）虽然传统的 TMR 实现能够有效地减少来自单粒子翻转造成的单一故障，但是缺少纠正这一错误的方法。在三模设计中，当同一部分的另外一个域同样发生翻转，那么多数判决器将输出错误的结果。不能避免单粒子效应引起的错误，但是可以

极大的减少错误输出的可能性。

三模设计中的每一部分有一下三种工作状态：

（1）完整的：所有的配置存储器和设计存储器的比特位都是正确的；

（2）单一域错误：一个域的配置或者状态发生翻转产生错误输出，但多数选择器上的最终输出是正确的。

（3）多个域错误：多个域的配置或者状态同时发生翻转产生错误输出，多数选择器输出错误结果。

避免多数选择器错误输出的关键在于尽可能的减少单一域发生翻转故障的恢复时间，这样就可以减少产生多域错误可能性。因此 Xilinx 提出了一种改进的 TMR 实现方法，其主要思想是利用多数选择器的正确输出来几乎实时纠正发生翻转域的状态。

### 4.1.3 改进的 TMR 方案

根据上文描述的传统 TMR 存在的缺陷，Xlinx 提出改进的实现方案 XTMR, 如下：将整个三模设计划分为三个功能模块，每一个功能模块都进行相关的可靠性设计。事实上，冗余域设计的起始和结束都是在 PCB 板上的物理实现，不会受到太空辐射的影响。所有的输入、输出和多数选择器电路均进行的三模冗余设计，从而消除了这些资源的单点错误故障。具体如下[26]。

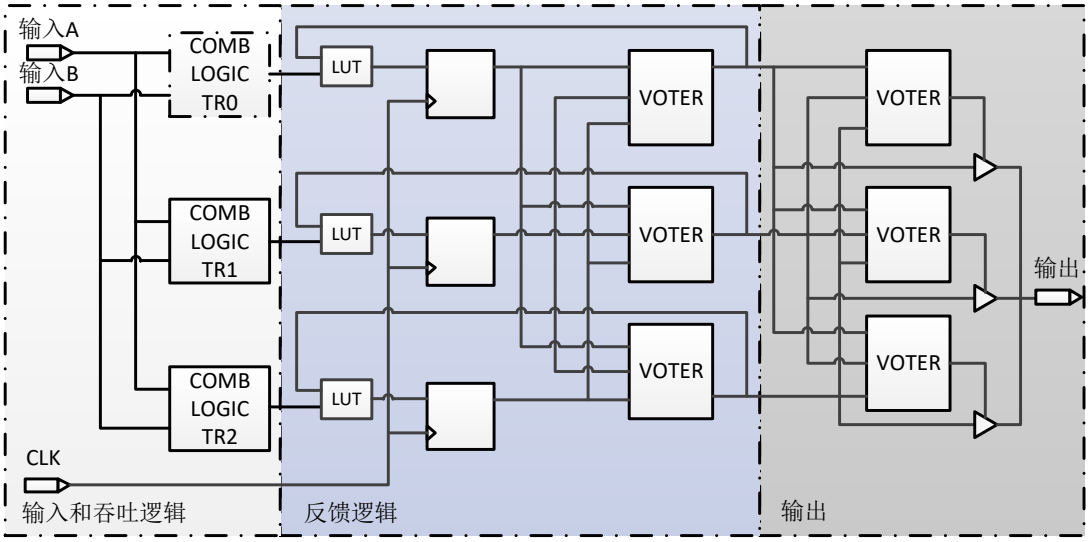



图4. 3 改进的 XTMR 实现

（1）输入和吞吐逻辑

首先将输入和吞吐逻辑进行三模处理，每一个域相互独立工作。

（2）XTMR 反馈逻辑

XTMR 在所有的反馈路径中插入多数选择器，保证了冗余状态机之间的同步。

但是综合选项设置会影响设计中的反馈电路。为了尽可能降低多域翻转导致的不正确输出的风险，必须保证状态机的综合方式，即就是尽管一些状态不频繁使用，反馈逻辑需一直频繁的执行。

（3）XTMR 输出

Xilinx 公司的 FPGA 架构不具有辐射硬化的多数选择器电路，故对选择电路进行冗余设计，最终的输出是由器件封装的引脚引出。

如果设计中的吞吐逻辑或者状态机出现单点故障，一个冗余域的行为状态与其他两个不同，该域的输出多数选择器检测到当前域的异常并且禁用三态缓冲器，使其引脚处于高阻态其他两个域保持正常工作，保证了最终的正确输出。

### 4.1.4 三模冗余设计步骤

对数字系统进行 TMR 设计可以使用 Xilinx 公司的专用工具 TMRTool。需要注意的是，所使用的 TMRTool 工具版本必须与数字系统开发所使用的 Xilinx ISE 版本对应一致。本次设计所采用的两个工具版本为 ISE 9.2i 和 TMRTool 9.2i。进行三模冗余设计的流程如下：

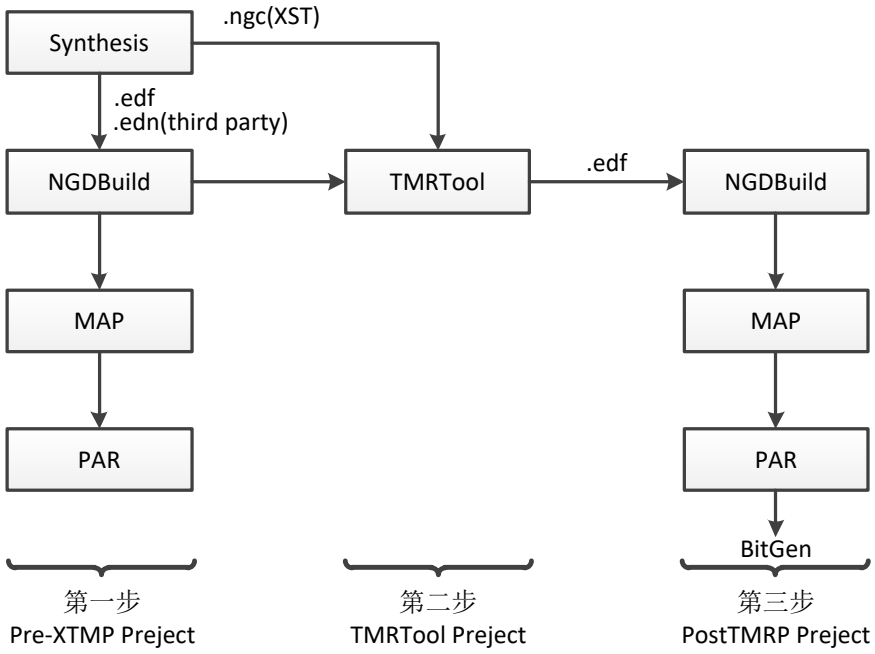



图4. 4　　XTMR 设计步骤

其中，XST 是 Xilinx 公司开发工具 ISE 附带综合工具，生成的网标文件格式为.ngc，可直接作为 TMRTool 工具的输入文件。NGDBuild 的作用是将其他厂商综合工具综合生成的网表文件转换为 TMRTool 支持的文件格式，主要包括.ngd 和.ngo 格式。

（1）创建三模前工程

使用 ISE 实现对目标工程的创建[27]，该工程是三模设计前的，用于生成三模工具 TMRTool 所需的输入网表文件（格式为.ngc）。在生成网表文件之前，必须保证原工程已经经过严格的功能、性能和时序仿真验证，并且无遗留问题。综合后实现时设置实现属性，将 Map Properties 项目下的 Pack I/O Registers/Latches into IOBs 设置为 For input only。完成 Implement Design 后生成工程顶层网表文件。

（2）创建 TMRTool 工程[26]

创建一个空白工程，添加输入文件具体步骤为：

a）在 TMRTool 工具的 source 窗口空白处右击选择“set top-level source”；

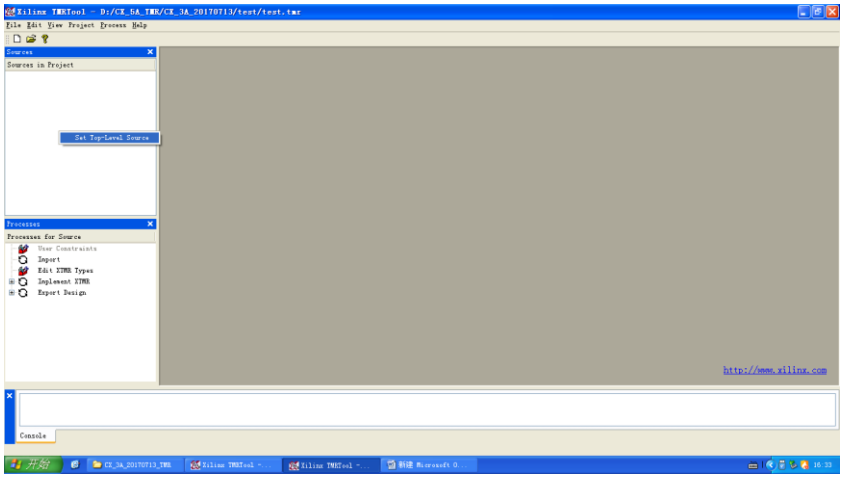


图4. 5　　TMRTool 添加顶层文件

b）在 TMRTool 工具的 source 窗口空白处右击选择“add source”，添加其他 IP 核 ngc 文件；

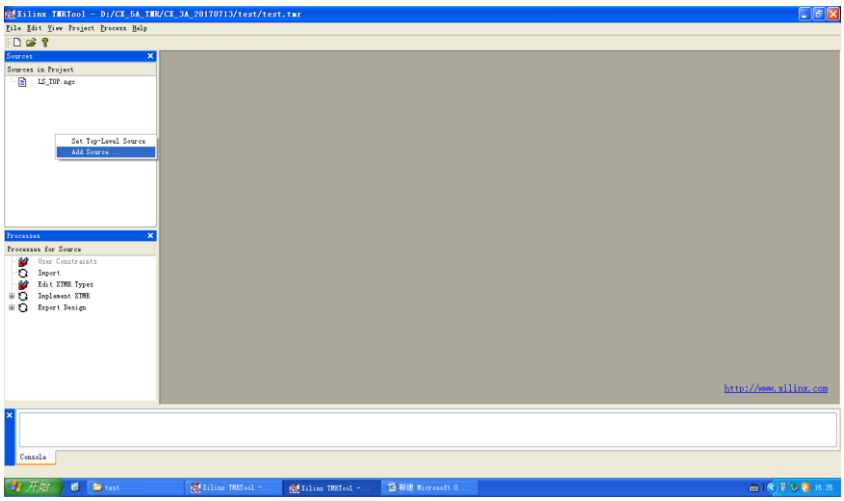


图4. 6　　TMRTool 添加 网标约束文件

c）在右侧 processes for source 窗口中双击“import”，等待完成；

d）双击“Edit XTMR Types”，右侧出现类型设置窗口，包括两种模式，一种是 Component 模式，一种是 Hierarchy 模式；

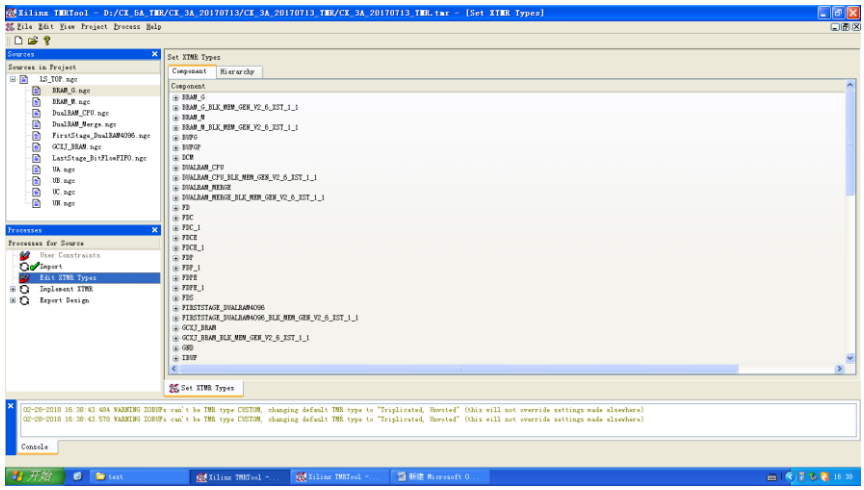


图4. 7　　TMRTool 信号选择

在后者选项卡中可以按照系统模块分级对信号进行单独的三模设置。

e）对所有的组件和信号完成设置后，右击 Implement XTMR 进行属性设置，设置完成后双击 Implement XTMR，等待结束；

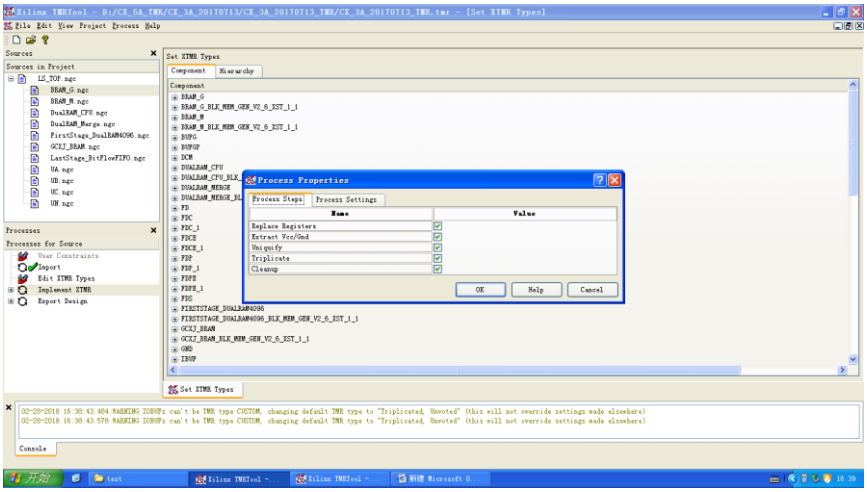


图4. 8　　TMRTool 设置实现参数

f）双击 Export Design，生成 EDF 文件。

（3）创建三模后工程

a）ISE New Project

创建工程名，选择工程路径。【Top-Level Source Type】:HDL 为创建普通工程；

EDIF 是创建三模工程。故选择 EDIF，Next。

b）选择工程输入文件

Input Design：选择三模生成后缀为.edf 的文件

Constraint File：选择工程用户约束文件 UCF。

c）设置芯片与器件类型；

d）进行 Implement Design 属性设置，最终生成 bit 文件。

## 4.2 图像源容错技术

本系统输入的图像分为两种类型：大图图像和小图图像。其中大图图像规格为 1028*1072*16bit；小图图像规格为 256byte。每种图像包括不同的帧头巴克码，标识每一帧图像的开始。系统接收图像时，只有检测到正确的帧头信息才判定为一帧图像开始输入，否则一直处于等待和检测状态。图像在传输过程中还可能发生数据丢失的情况，当数据量丢失过多，当前图像就没有处理的意义了，故对图像长度也进行检测。

### 4.2.1 大图图像容错技术

（1）大图巴克码错误检测

正常的大图图像帧头巴克码为 0x4954CE1F066B0000，可以容忍任意 4bit 的错误。错误比特数小于等于 4，认为是正确的巴克码标志，大于 4 个，则被认为非巴克码标志，不启动数据处理部分。

（2）大图帧长度错误检测

一帧图像正常大小是 1072*1028*16bit，以逆程分隔，每次发一行数据。帧长度计数器在，帧头检测有效后启动，在图像输入结束后停止计数。帧长度错误可分为以下两种情况：

a）长帧：如果一帧图像大于正常图像长度，则将多余的部分丢弃，在此基础上判定当前图像为正常帧。

b）短帧：若一帧图像大小不足 1072*1028*16bit，则在选通信号无效的情况下，进行计时，若选通信号无效的时间大于 1072μs + 178μs*2 = 1428μs，则产生结束信号送到压缩以及打包等相关模块，帧长度错误计数加 1，结束当前帧的处理，等待下一帧到来。

### 4.2.2 小图图像容错技术

（1）小图巴克码错误检测

正常的图像帧头巴克码为 0xEB90，可以容忍任意 1bit 的错误；错误小于等于 1，认为是正确的巴克码标志，大于 1 个，则被认为非巴克码标志，不启动数据处理部分。

（2）小图帧长度错误检测

一帧图像正常大小是 256*8bit，帧长度错误可分为两种情况：

a）长帧：在选通信号有效地情况下，可能会出现一帧的长度大于 256*8bit，这种情况下，设计中会将大于 256*8bit 的部分舍弃。

b）短帧：若一帧图像大小不足 256*8bit，小图的模式是以三帧为单位打一包，一包总数据字节数为 1000，正常情况下三帧的有效数据为 768byte，若为短帧，则将不足的地方用 0XFF 填充。

在对帧长度计数的过程中，可能存在超长帧，导致计数器溢出并重新累加到有效计数点，误判为帧长度正常的情况，在设计计数电路时，当计数器第一次达到帧长度值时，触发使能信号，结合使能信号和计数状态就可以有效的避免计数溢出异常状态。

## 4.3 单帧独立技术

对于图像压缩系统而言，系统中存在各种状态寄存器和累加器，用于表示压缩过程所处的状态以及压缩过程中对像素值进行修正的参数，这些状态与压缩过程是相匹配的，编码器对前后两帧图像的处理是完全独立的，除了帧计数寄存器的累加，没有任何相关性，它们图像的压缩起着至关重要的作用。因此，当状态不对应，压缩的有效性就无法保证。单帧独立处理方法就是在每帧图像压缩处理前，对压缩系统的各种参数和寄存器进行复位操作，保证每一帧开始的状态是正确的，同时也避免了错误的蔓延。

如果编码过程中由于特殊原因造成编码状态异常，编码器无法进行自修复，将状态恢复到正常，因此在设计单帧独立处理机制时，必须利用外部输入信号来控制编码状态恢复，使得编码器恢复到初始状态。

设计单帧独立时，状态的恢复是利用外部复位信号对编码器进行初始化操作，这就设计到对外部复位信号的同步处理问题。

复位信号是对编码器寄存器进行复位操作，分为异步复位和同步复位。

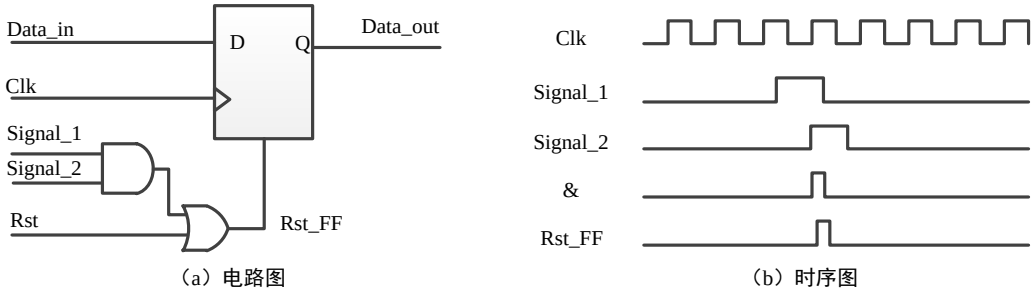
异步复位工作方式是独立于驱动时钟的，可以使用 FPGA 触发器的异步复位端口，可以节省系统的资源开销。但是异步复位存在一些严重的问题。第一，如果用来复位的信号是由前端组合逻辑产生，那么对于电路而言，生成复位信号过程中可能存在竞争冒险现象，使得复位信号中存在毛刺，控制信号中的毛刺可能导致电路瘫痪；第二，复位信号与时钟信号相位关系的不确定性，导致触发器的建立保持时间要求得不到满足，最终可能导致亚稳态的产生，下面进行具体分析。

（1）复位信号中的毛刺

数字电路中信号的跳变需要一定的时间。同时，在 FPGA 内部信号通过连线与逻辑门都会有延迟，延迟时间大小受多种因素影响。通常多个信号在跳变时，达到稳定状态所需要的时间也各不相同，这样的事实在逻辑电路中会造成逻辑毛刺，叫做电路冒险[29]。

任何一个门电路只要有两个输入信号同时向则相反的方向变化，其输出都有可能产生毛刺。冒险现象会对电路的稳定性造成巨大影响，尤其是时钟信号和一些控制信号，一点毛刺就可能是的电路系统的功能瘫痪。下面进行说明。

信号 Signal_1 和 Signal_2 是产生复位信号的两个相关信号(编码结束信号与帧尾信号)，寄存器 FF 是编码器的某个关键信号产生寄存器，如图 4.9（a）所示。在一帧图像结束后并且编码完成时，信号 Signal_1 和 Signal_2 同时产生高电平脉冲，进而生成复位信号，对 FF 进初始化操作。但是在实际电路中，信号 Signal_1 和 Signal_2 来自不同的逻辑，不会同时置高。




（a）电路图　　（b）时序图

图4. 9　　单帧独立异步复位

根据上图中（b）可以看出，信号 Signal_1 和 Signal_2 延迟的不确定性，可能会导致出现图（b）中所示的时序关系，使得复位信号产生一个瞬时高电平脉冲，这是由于电路是异步复位的缘故，一旦复位信号高电平有效，电路立刻复位，造成编码功能紊乱，输入信号被复位，输出数据发生异常。这不是期望的的结果。

对于上述问题，要消除电路中组合逻辑产生的毛刺对电路的影响，就必须对输入的信号进行处理，增加额外的逻辑来抑制不正常的数据输出。

为了消除上述电路中产生的不正常复位信号，提出以下同步复位设计方案，如图 4.10 所示。

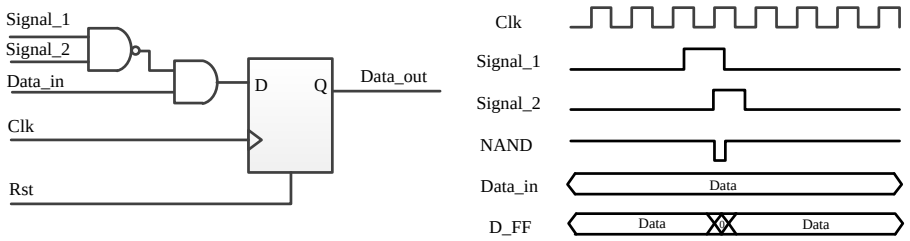



图4. 10　单帧独立同步复位

对于上述用同步处理的单帧独立设计，与非逻辑产生的复位信号控制触发器的 D 端数据输入，编码器还未进入复位状态时，如果组合逻辑产生了复位毛刺，不会对电路输出产生影响如上图（b）时序关系所示。

（2）异步信号的亚稳态问题

亚稳态（Meta-stability）是在特定的时间内一种介于 0 和 1 电平之间的一个不确定状态，是跳变沿的一个固有特性。在两个时钟域之间信号的传输时，如果原时钟域下输出信号的变化正好在新时钟域时钟的建立保持时间窗口中，则会出现亚稳态[30]。其一般出现在 D 触发器的 Q 输出端，正常的采样也会有亚稳态的出现。只有在满足建立保持时间的时候，跳变沿经历采样、亚稳态之后进入一个正确稳定的状态；否则，在跳变沿处出现相当长时间的亚稳态（这将导致下一级进入亚稳态），最终进入一个错误的稳定状态。如图 4.11 所示。

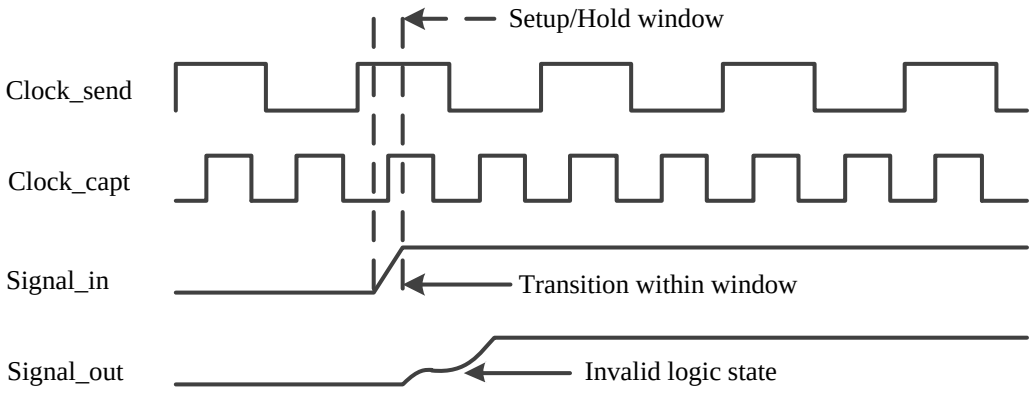



图4. 11　亚稳态的产生

建立时间（Setup Time）：指在触发器的时钟上升沿到来以前，数据稳定不变的时间。如果建立时间不够将会把错误的数据在上升沿打入触发器。建立时间决定了该触发器之间组合逻辑的最大延迟[31]。

保持时间（Hold Time）：指在触发器的时钟上升沿到来以后，数据稳定不变的时间。如果建立时间不够，正确的数据同样不能打入触发器。保持时间决定了该触发器

之间组合逻辑的最小延迟[31]。

对于异步复位的单帧独立设计，只要组合罗即产生有效的复位信号，无需等到时钟有效沿的到来，就对触发器进行复位操作，当复位操作发生在触发器的有效沿附近，并且位于数据跳变的建立保持时间窗口内，将发生亚稳态，造成数据输出异常。如图 4. 12 所示。

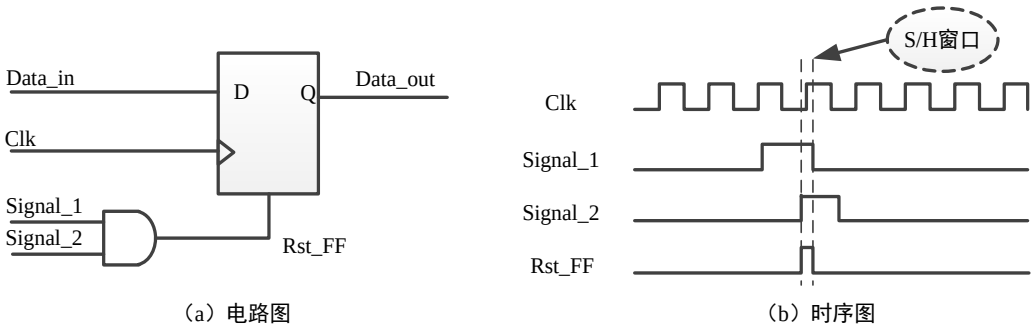



图4. 12　异步复位信号

基于以上分析，在本系统中单帧独立复位信号避免直接使用异步复位方式，使用 D 触发器本身的异步复位端口，必须对输入的异步信号进行同步处理，避免毛刺以及亚稳态的产生。

## 4.4 逻辑毛刺滤波

图像压缩系统 LVDS 接口与外部图像源相连，外部输入的信号包括串行图像数据（data_in），输入随路时钟(clk)，以及行同步信号(line_syn)，如图 4.13 所示。其中行同步信号是低电平有效，且在行同步信号有效器件，输入有效的图像数据。在行与行之间，存在一个逆程，该逆程长度的最小值为 16 个随路时钟周期，当长度超过 65530 个时钟周期时，则判定为一帧图像输入结束。

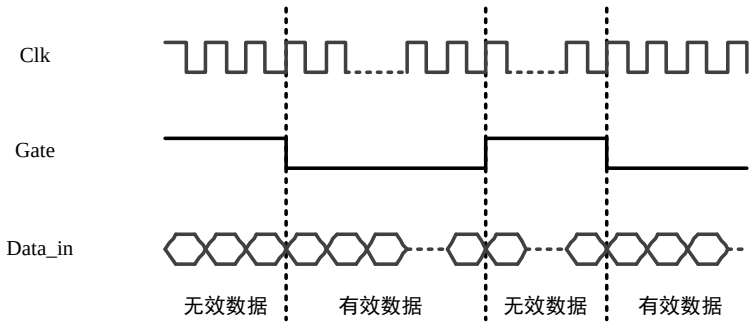



图4. 13　图像输入接口信号时序

当输入信号中，Gate 信号的低电平有效期间出现持续时间小于 16 个时钟周期的高电平（信号毛刺），图像接收端就有可能将一行数据判断为两行或者多行，造成图像数据损坏，为了避免上述问题，对输入的信号进行滤毛刺处理，其处理方法为：利用输入随路时钟对输入 Gate 信号进行 16 级寄存器打拍处理，然后产生一个新的信号 Syn，该信号就不存在上述毛刺，不会造成图像行误判。

## 4.5 强制直通模式

在本系统中，强制直通模式是区别与其他正常压缩模式的，是一种测试模式 ，可以用来验证系统数据链路链接的正确性，配置指令为 0xF1。在正常模式下，输入图像数据首先是进行一系列检测，只有检测到有效的图形数据时（有效的图像帧头和长度），产生一个图像有效使能信号，压缩模块输入端接收到该使能信号，才会进行压缩处理，最后打包输出；而处于强制直通模式，对输入的图像直接进行处理，处理过程不经过压缩模块，直接打包输出。当系统出现异常情况时，配置系统模式为强制直通模式，可方便有效的进行故障树分析。

## 4.6 流水线设计

在高速数字电路设计中，通常会用到流水线处理方法。流水线处理思想是：对于一个复杂的处理过程，通过合理划分，将其分为多个顺序步骤依次进行处理。这样在每一个步骤中输入输出都经过寄存器打拍，使得整个处理过程关键路径延时大大提高，因此可以大幅提高系统速率[28]。

图像压缩算法对图像的处理过程是按照步骤一步一步进行的，这正好与流水线思想相吻合，只有部分阶段会与较前面的步骤有参数反馈，比如 JPEG-LS 算法中参数的更新，不包括模块之间的控制信号。所以对于算法的整体是不能全部进行流水线设计的，但是通过有效的二级模块设计和划分，可以有效的避免上述弊端，正如本设计中的 FirstStage 模块、MidStage 模块和 LastStage 模块，在每个模块内中的子模块都是按照流水线。为了保证算法模块的关键路径延时尽可能的小，对每一个模块的输出进行寄存器打拍处理，实现内部流水结构。

## 4.7 跨时钟域处理

在一个 FPGA 系统设计中，经常需要处理多个时钟来源，不同的时钟域有不同的时钟频率和时钟相位。如何处理好多个时钟信号在 FPGA 内部的关系，让数据以及相关的控制信号在不同的时钟域之间准确的传递就是一个比较困难的问题。在这种情况

下，建立与保持时间就显得尤为重要。

在前文已经详细叙述了亚稳态现象以及产生亚稳态的原因。亚稳态的产生是信号在经过触发器时，不满足触发器的建立保持时间，导致对信号的采样发生错误。对信号的跨时钟域处理的目的就是为了最大限度的降低亚稳态发生的几率，保证信号在不同时钟域之间传输的正确性，进而保证了系统的可靠性。信号的跨时钟域处理最主要的就是同步器的设计。

一般情况下，信号跨时钟域处理根据前后时钟域时钟的速率和信号类型的不同可以分为以下几种情况：控制信号从慢时钟域到快时钟域，控制信号从快时钟域到慢时钟域；总线信号从慢时钟域到快时钟域，总线信号从快时钟域到慢时钟域。具体进行以下分析。

### 4.7.1 控制信号跨时钟域处理

通常，控制信号都是单比特信号，对单比特信号的跨时钟域同步处理一般采用的方法是多级寄存器处理。但是，对于前后时钟域时钟速率的不同，还需要进行额外的处理。

（1）慢时钟到快时钟

单比特信号从慢时钟到快时钟传输时，通常首先在快时钟域下对其进行同步打拍（一般是两级触发器）处理。针对不同的信号需要增加额外的处理逻辑，保证信号的正确传输。比如：电平信号的同步、信号跳变沿同步和脉冲同步。

a）电平信号的同步处理

对电平信号同步处理要求输入的高电平或者低电平信号的电平保持时间必须多于快时的两个周期；输入的电平信号再次变为有效之前先进入无效状态，即就是每次状态有效只被判定为一次有效事件。用两级触发器打拍处理即可以保证信号跨时钟域时的正确传输，当然打拍级数越多则发生亚稳态的可能就越低，如图 4.14 所示。

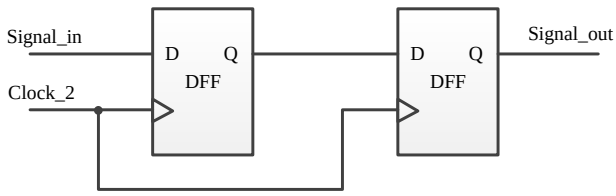



图4. 14　电平信号两级寄存同步处理

b）信号跳变沿同步处理

跳变沿同步处理是将一个电平的上升沿或者下降沿传递到下一个时钟域。同步的方法是在电平同步器的基础上，在输出端增加一个触发器，将此触发器的反向输出与

电平同步的结果相与，就在新的时钟域产生一个脉冲信号，脉冲宽度为一个时钟周期，如图 4.15 所示。同时如果将反向器加在电平同步的输出端，将得到电平信号的另一个跳变沿的同步输出。

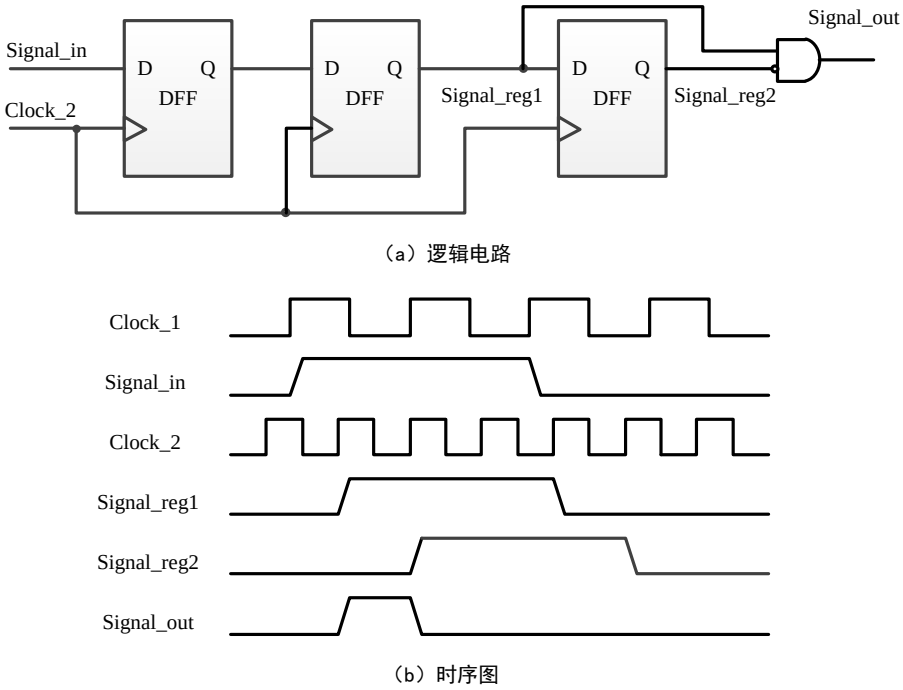



图4. 15　信号跳变沿同步处理

c）脉冲同步处理

将一个时钟域的脉冲信号（持续一个时钟周期）同步到另一个时钟域，两个脉冲之间的间隔不小于两个时钟周期，否则在新时钟域下可能出现脉冲合并现象。同步方法如下：在原时钟域利用输入信号来驱动 D 触发器，在将触发器的两个输出（同相和反相）通过与运算，结果作为该触发器的输入，对该输出进行跳变沿检测，如图 4.16 所示。

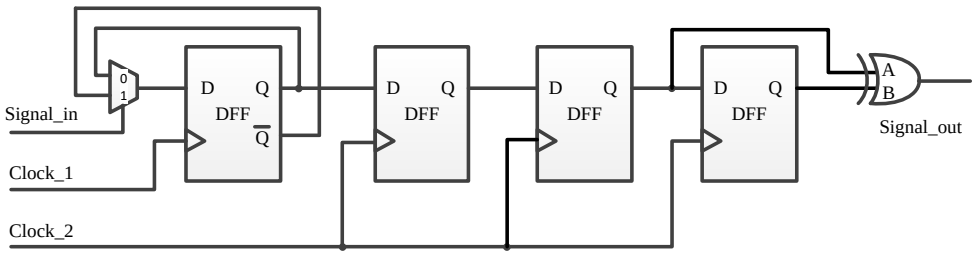



图4. 16　脉冲信号同步

如上图，在脉冲同步器中选择器与第一个触发器组成的逻辑的作用是，在第一个时钟域内，将脉冲信号转变为电平信号，在此基础上在对同相输出进行跳变沿同步处理。

（2）快时钟到慢时钟

两级寄存器打拍同步处理方法仅适用于从慢时钟到快时钟的信号同步处理，否则前级时钟频率快于后级时钟时，后级对数据的采样就有可能失效。此时应采用结绳法来处理信号的跨时钟域同步问题。

结绳法的思想是：双锁存法无法采样到变化较快的信号，那么可以让变化快的信号保持足够长的时间，之后再进行双锁存法对其进行同步处理。双锁存法是利用输入信号的上升沿（或者下降沿）将脉冲无限延长，直到可以采样到正确数据，然后复位，如图 4.17 所示。

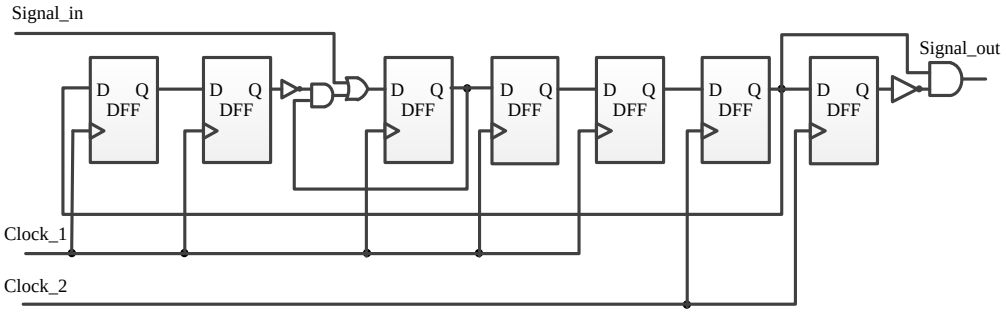



图4. 17　结绳法逻辑电路

### 4.7.2　总线信号跨时钟域处理

不同于单比特信号，总线信号是并行的多位单比特信号，这样在跨时钟域同步处理时，每一位数据都有可能发生跳变，很难保证每一位信号都满足触发器的建立保持时间要求，因此对总线信号的跨时钟域同步方法通常采用以下两种方式：握手协议控制数据传输和 RAM 隔离。

（1）握手通信同步方法

握手原则是指在数据传输的前后级时钟域分别产生用于表示数据传输状态的信号来保证数据的有效性和正确性，这些控制信号包括：发送请求信号和接收确认信号。具体的实现过程如下。

时钟域 A 和时钟域 B 之间进行数据传输，首先由 A 域产生发送请求信号 req（request），经过同步后到达 B 域，当在时钟域 B 中检测到有效（高电平）的 req 信号，从数据总线接收数据，同时产生并发送确认信号 ack（acknowledge），ack 经过同步后到达 A 域，电路检测到有效的 ack 信号后，将 req 信号置为低电平，在 B 域检测到拉低的 req 信号后，也将 ack 信号置为低电平，当 A 域检测到 ack 为低电平后，表

示一次数据发送成功，完成后才可以进行下一次数据的传输，传输过程的时序关系如图 4. 18 所示。

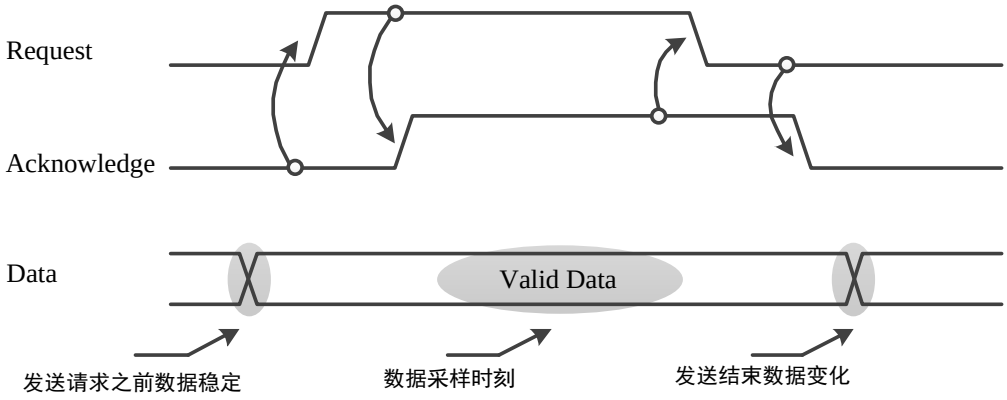



图4. 18　握手原则

在传输过程中 req、ack 信号的同步处理采用前文中的方法。但是整个通信过程需要多次进行状态的反馈，在为检测到有效的状态标志信号，一直处于等待状态，一次有效的数据传输过程需要较多的时间，传输效率较低，不太适合较大数据量的同步传输。同时，如果前级时钟速率较后级大很多，会造成数据拥堵。

（2）RAM 隔离同步方法

最常见的隔离方法是采用异步 FIFO。异步 FIFO 的指读写时钟不同，读写控制信号分别来自不同的时钟域。一般用于时钟域的隔离。首先将需要同步的数据进行缓存，然后进行读取输出。即保证了数据传输的可靠性，又保证了传输的速率。

FIFO 的满状态是根据读端的地址判断，空状态是根据写端的状态判断，因此存在读写地址分别在读写端的同步问题。在对地址指针进行编码时，采用格雷码，能有效的避免亚稳态的产生。

因为存在地址同步的问题，对数据的同步也可以采用双端口 RAM 处理，读写地址均来自各自的时钟域，不存在额外的同步问题，但是会有较大的资源开销。

### 4.7.3　设计中跨时钟域处理说明

（1）相机时钟域到本地时钟域

本次对信号的跨时钟域处理由两部分组成，一部分是大图图像数据链路；一部分是小图图像数据链路。对于大图图像数据而言，是从相机的 32MHz 随链路时钟域到本地 32MHz 时钟域，对小图图像数据而言，是从 10MHz 随路时钟域到本地 32MHz 时钟域。现对相关信号作如下说明。

（2）本地时钟域到打包时钟域

大图图像链路数据经过的压缩处理是在本地 32MHz 时钟域下进行的，在进行打包处理是在 48MHz 时钟域下进行的，所以对大图图像数据链路信号需要作跨时钟域处理。小图图像数据了链路数据不需要进行压缩，但也需要对串并转换后的数据和标志信号进行打包处理，对链路信号进行本地 32MHz 时钟域到 48MHz 时钟域的跨时钟与处理。相关信号具体如下表所示。同样地，CPU 链路信号也需要进行本地到打包时钟域的跨时钟域处理。

## 4.8 本章小结

本章主要针对不同因素对系统的可靠性和容错机制进行分析和实现。对于应对太空单粒子效应，对系统电路进行三模冗余处理，并且利用 Xlinx 改进的 TMR 技术实现了整个三模电路的可靠性；另外，对于图像源异常，提出并设计了相关的容错机制，保证了从源头到整个系统的可靠性。再有，为了保证整个系统电路状态的稳定性，进行了其他的设计。

# 第五章 系统测试与验证

对整个系统的测试包括 JPEG-LS 图像压缩算法的验证和对系统可靠性和的验证。

首先是测试平台的搭建。对系统可靠性的仿真验证平台是 Xilinx ISE 12.4 和 ModelsimSE 6.5e，根据实际待测的功能设计 TestBench 产生系统的输入信号，验证系统输出结果。对算法的验证利用上位机和地检链接，输入测试图像，系统输出的是压缩后的打包码流，如图 5.1 所示。其中上位机进行控制图像输入以及相关参数配置；地检系统负责对输入图像的压缩处理，最终输出打包后的码流；数据采集系统对码流数据进行接收存储。所以要对输出的数据进行解包解压缩处理，最终得到恢复后的图像帧。

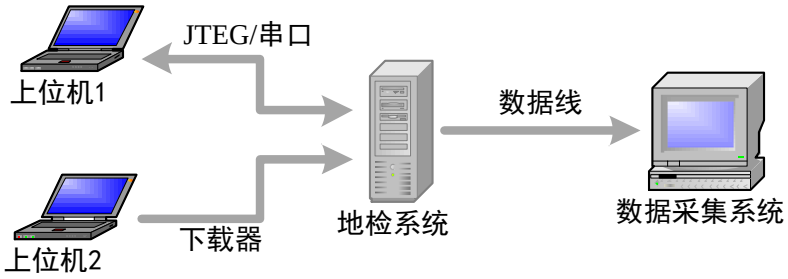



图5.1 图像压缩处理平台

在仿真验证过程中，根据不同的需求设计不同的 TestBench，在 Modelsim 的波形框中分析相关信号的时序关系和状态，达到对仿真验证的目的。

对算法的验证，首先生成每个压缩模式下的 bit 流文件，在 ISE 的 impact 工具进行文件件烧写，最终得到不同压缩模式下的恢复图像，利用 photoshop 软件，打开图像设置正确的选项，只直观的了解系统对图像的压缩功能是否正常；利用批处理软件对压缩前后图像进行分析处理，得到压缩的指标信息，得到算法的压缩性能情况。

## 5.1 系统资源占用分析

（1）TMR 前资源时钟分析

对系统各个时钟的时钟分析结果如下：

Timing summary:

Timing errors: 0 Score: 0

Constraints cover 4048088 paths, 0 nets, and 33502 connections

Design statistics：

Minimum period: 18.716ns{1} (Maximum frequency: 53.430MHz)

Minimum input required time before clock: 1.262ns

Maximum output delay after clock: 12.651ns

三模前系统资源占用分析如下：

表5.2 TMR 前系统资源消耗情况

| Device Utilization Summary | | | | |
|---|---|---|---|---|
| Logic Utilization | Used | Available | Utilization | Note(s) |
| Number of Slice Flip Flops | 6164 | 28672 | 21% | |
| Number of 4 input LUTs | 6205 | 28672 | 21% | |
| Logic Distribution | | | | |
| Number of occupied Slices | 5913 | 14336 | 41% | |
| Number of Slices containing only related logic | 5913 | 5913 | 100% | |
| Number of Slices containing unrelated logic | 0 | 5913 | 0% | |
| Total Number of 4 input LUTs | 6764 | 28672 | 23% | |

（2）TMR 后资源与时钟分析

对系统时钟分析结果如下：

Timing summary:

Timing errors: 0 Score: 0

Constraints cover 13991129 paths, 0 nets, and 78099 connections

Design statistics:

Minimum period: 23.738ns (Maximum frequency: 42.127MHz)

Maximum output delay after clock: 9.698ns

三模后资源占用分析如下：

表5.3 TMR 后系统资源消耗情况

| Device Utilization Summary | | | | |
|---|---|---|---|---|
| Logic Utilization | Used | Available | Utilization | Note(s) |
| Number of Slice Flip Flops | 12123 | 28672 | 42% | |
| Number of 4 input LUTs | 15561 | 28672 | 54% | |
| Logic Distribution | | | | |
| Number of occupied Slices | 13537 | 14336 | 94% | |
| Number of Slices containing only related logic | 13537 | 13537 | 100% | |
| Number of Slices containing unrelated logic | 0 | 13537 | 0% | |
| Total Number of 4 input LUTs | 16792 | 28672 | 58% | |

可以看到虽然性能有部分下降，但仍然满足设计要求中时钟 80%的降额。实际测试、联试验证了功能正常。

## 5.2 JPEG-LS 算法性能测试

对算法功能的验证主要是测试不同压缩模式下，对图像的压缩是否正常。几种压缩模式包括：2:1 压缩、4:1 压缩和 8:1 压缩。大图图像压缩性能检测是根据在不同压缩比模式下进行数据压缩打包，不同压缩比下，有不同的压缩性能要求。将输出的打包数据解包解压缩，并且与输入的原图数据进行对比，计算出客观性能指标，包括均方误差（Mean Squared Error，MSE）以及峰值信噪比（Peak Signal to Noise Rate，PSNR）等；也可以根据主观查看，本测试选择客观指标进行性能说明。

图像压缩 PSNR 性能测试是对压缩后的图像做恢复处理，在将原图像与恢复出来的图像进行信噪比对比，结果能够有效反应数据压缩的处理性能。测试结果见表 5. 4——表 5. 6 所示。

为了更直观的对图像压缩效果进行判断，以所有测试图像中的一幅图像为例，用 PS 图像工具打开原图像和压缩后的回复图像，测试结果如下：

表5.4 大图图像 8 倍压缩结果

| 图像名称（8 倍图像压缩） | 包计数 | 包大小 | MSE | MSD | PSNR |
|---|---|---|---|---|---|
| Test_image_09.raw | 272 | 0X0110 | 43.79 | 11 | 79.92dB |
| Test_image_12.raw | 272 | 0X0110 | 43.88 | 11 | 79.91dB |
| Test_image_15.raw | 273 | 0X0111 | 43.78 | 11 | 79.92dB |
| Test_image_18.raw | 273 | 0X0111 | 43.95 | 11 | 79.90dB |
| Test_image_21.raw | 273 | 0X0111 | 43.85 | 11 | 79.91dB |
| Test_image_0352.raw | 161 | 0X00A1 | 35.10 | 11 | 80.88dB |
| Test_image_0355.raw | 161 | 0X00A1 | 35.34 | 11 | 80.85dB |
| Test_image_0358.raw | 161 | 0X00A1 | 35.17 | 11 | 80.87dB |
| Test_image_0401.raw | 161 | 0X00A1 | 35.34 | 11 | 80.85dB |
| Test_image_0404.raw | 161 | 0X00A1 | 35.31 | 11 | 80.85dB |
| Test_image_35.raw | 164 | 0X00A4 | 36.67 | 11 | 80.69dB |
| Test_image_38.raw | 164 | 0X00A4 | 36.40 | 11 | 80.71dB |
| Test_image_41.raw | 164 | 0X00A4 | 36.51 | 11 | 80.71dB |
| Test_image_53.raw | 164 | 0X00A4 | 36.43 | 11 | 80.71dB |
| Test_image_56.raw | 164 | 0X00A4 | 36.64 | 11 | 80.69dB |

其中的一幅图像的压缩测试结果直观显示如下，其中左边为原图，右边为恢复图。PSNR 为 80.69dB。

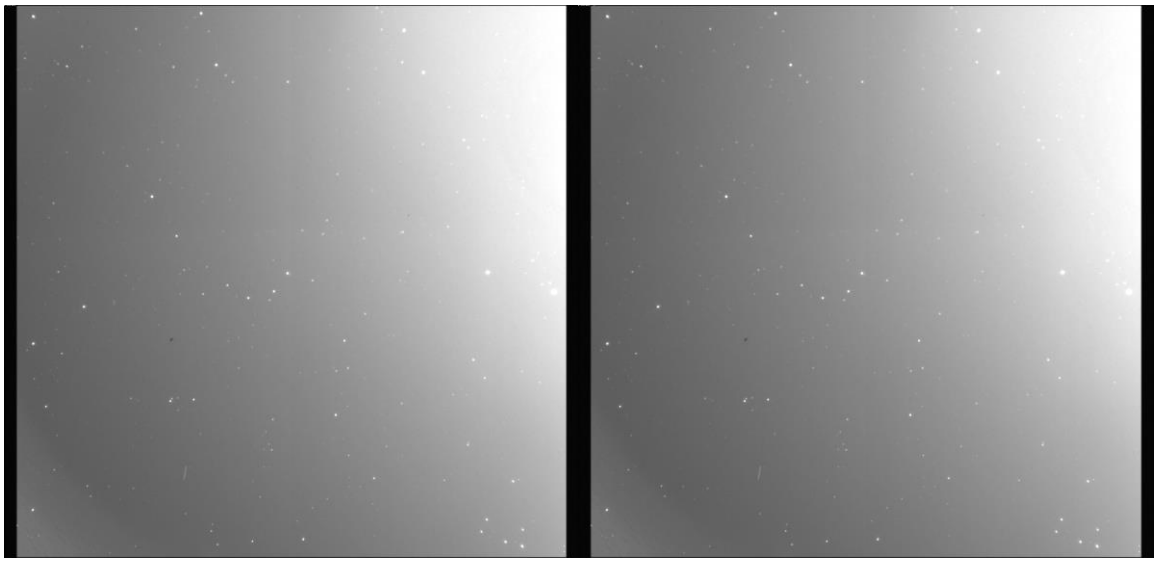


图5.2 8 倍压缩效果图

表5.5 大图图像 4 倍压缩结果

| 图像名称（4 倍图像压缩） | 包计数 | 包大小 | MSE | MSD | PSNR |
|---|---|---|---|---|---|
| Test_image_09.raw | 496 | 0X01F0 | 4.00 | 3 | 90.30dB |
| Test_image_12.raw | 496 | 0X01F0 | 4.00 | 3 | 90.31dB |
| Test_image_15.raw | 496 | 0X01F0 | 4.00 | 3 | 90.31dB |
| Test_image_18.raw | 497 | 0X01F1 | 4.00 | 3 | 90.31dB |
| Test_image_21.raw | 496 | 0X01F0 | 4.00 | 3 | 90.31dB |
| Test_image_0352.raw | 304 | 0X0130 | 4.00 | 3 | 90.31dB |
| Test_image_0355.raw | 306 | 0X0132 | 3.99 | 3 | 90.31dB |
| Test_image_0358.raw | 304 | 0X0130 | 4.00 | 3 | 90.31dB |
| Test_image_0401.raw | 305 | 0X0131 | 4.00 | 3 | 90.31dB |
| Test_image_0404.raw | 305 | 0X0131 | 4.00 | 3 | 90.31dB |
| Test_image_35.raw | 311 | 0X0137 | 4.00 | 3 | 90.31dB |
| Test_image_38.raw | 310 | 0X0136 | 3.99 | 3 | 90.32dB |
| Test_image_41.raw | 310 | 0X0136 | 4.00 | 3 | 90.31dB |
| Test_image_53.raw | 310 | 0X0136 | 4.00 | 3 | 90.30dB |
| Test_image_56.raw | 311 | 0X0137 | 4.00 | 3 | 90.30dB |

其中的一幅图像的压缩测试结果直观显示如下，其中左边为原图，右边为恢复图。PSNR 为 90.30dB。

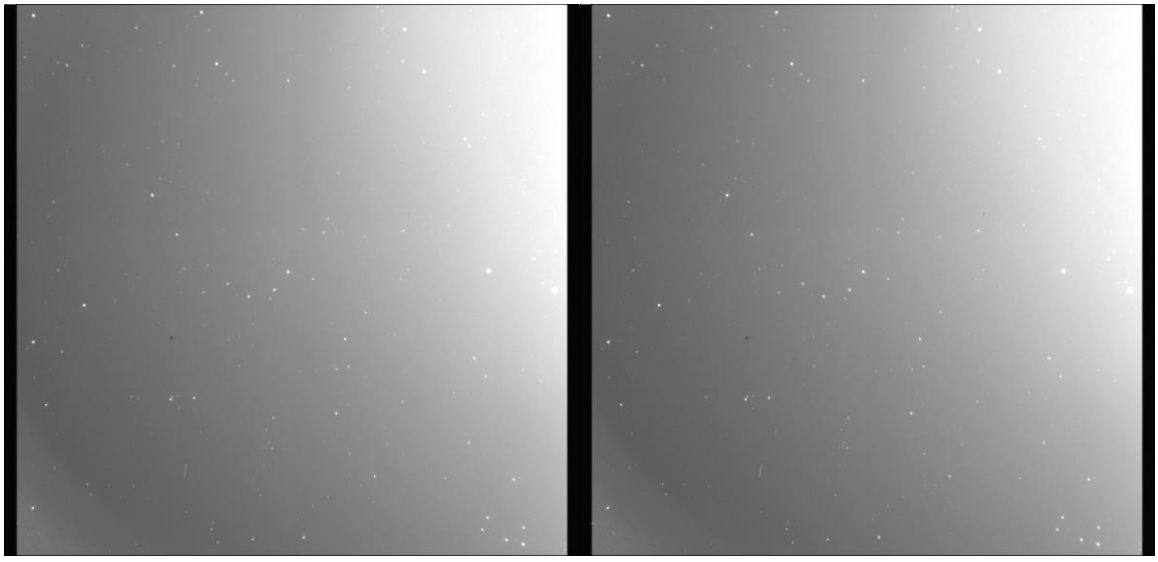


图5. 3　　4 倍压缩效果图

表5.6 大图图像 2 倍压缩结果

| 图像名称（2 倍图像压缩） | 包计数 | 包大小 | MSE |
|---|---|---|---|
| Test_image_09.raw | 874 | 0X036A | 0 |
| Test_image_12.raw | 874 | 0X036A | 0 |
| Test_image_15.raw | 875 | 0X036B | 0 |
| Test_image_18.raw | 875 | 0X036B | 0 |
| Test_image_21.raw | 875 | 0X036B | 0 |
| Test_image_0352.raw | 680 | 0X02A8 | 0 |
| Test_image_0355.raw | 681 | 0X02A9 | 0 |
| Test_image_0358.raw | 680 | 0X02A8 | 0 |
| Test_image_0401.raw | 681 | 0X02A9 | 0 |
| Test_image_0404.raw | 681 | 0X02A9 | 0 |
| Test_image_35.raw | 686 | 0X02AE | 0 |
| Test_image_38.raw | 686 | 0X02AE | 0 |
| Test_image_41.raw | 686 | 0X02AE | 0 |
| Test_image_53.raw | 686 | 0X02AE | 0 |
| Test_image_56.raw | 686 | 0X02AE | 0 |

其中的一幅图像的压缩测试结果直观显示如下，其中左边为原图，右边为恢复图。

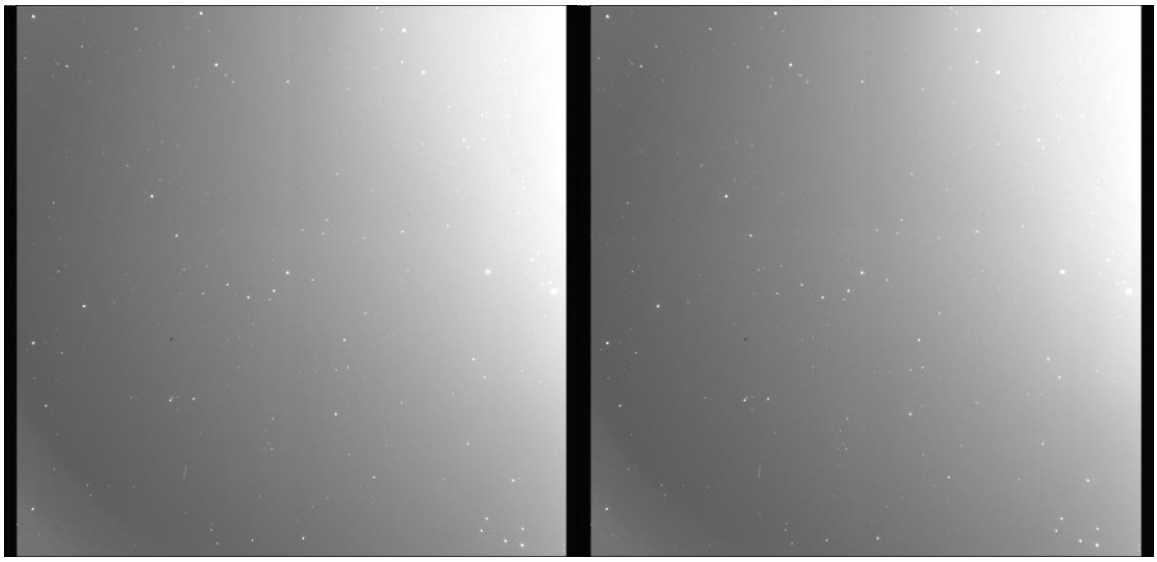


图5. 4　　2 倍无损压缩效果图

连续进行图像压缩处理时，当一幅图像在传输或者压缩过程中出现编码错误时，

根据本设计中的单帧独立原理可以预测到，最终回复出的图像应该只是在一帧中存在误码，不会影响到下一帧图像的处理，具体测试结果如下。图中右上图像出现错误，后续的图像处理结果正常。

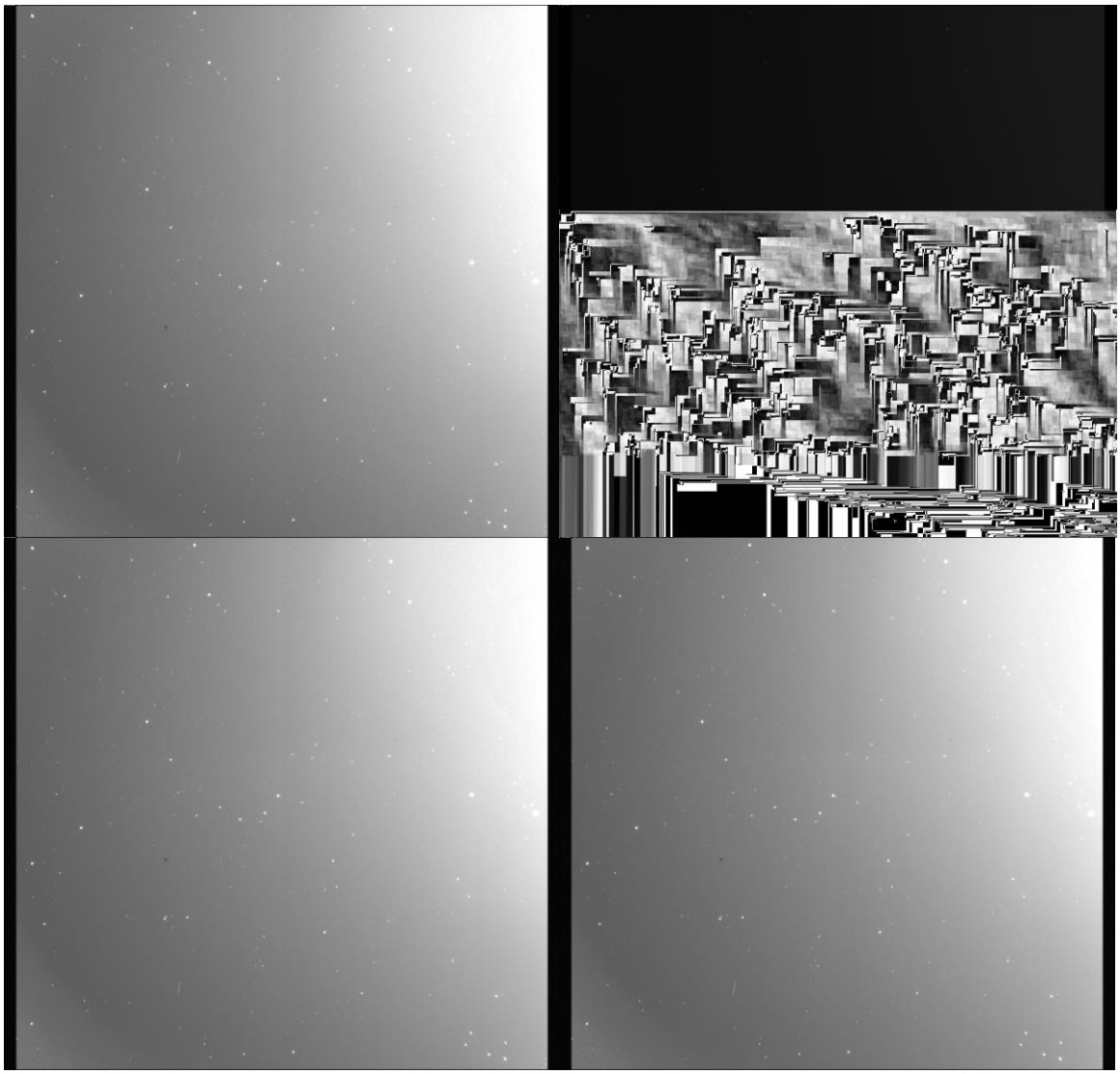


图5.5　　单帧独立测试结果

XX-XX 项目需求中对大图图像压缩的指标要求为：

（1）压缩比 2:1，无损压缩；

（2）压缩比 4:1，PSNR 不小于 80dB;

（3）压缩比 8:1，PSNR 不小于 71dB。

根据以上对压缩结果的统计表格数据以及对恢复的图像直观分析可以看出，各个压缩比下均满足要求指标。并且系统的单帧独立设计是有效的。

## 5.3 三模冗余有效性验证及结果

三模冗余设计主要是针对太空单粒子效应所采取的可靠性措施，器件电路发生单粒子效应是随机发生的，在实验测试中，几乎不可能捕捉单粒子故障的发生。因此在实际测试时，通过 Modelsim 进行仿真分析。通过在波形框中添加目标信号的每一模，然后根据其状态判断三模处理的有效性。

具体的方法是，利用 Xilinx ISE 工具创建三模后的工程，然后利用 ISE+Modelsim 联合仿真的方式进行。选取 TMR 电路中对应信号的 TR0，TR1，TR2 以及输出（o）信号，利用 Modelsim 中的 Force 功能模拟单粒子故障，验证该信号的输出是否正常。具体的测试结果如下：

（理论上来说，对进行了三模冗余处理的所有信号都需要进行验证，这里不一一例举，以 GCXJ_rec_en_control 信号为例，进行说明。）

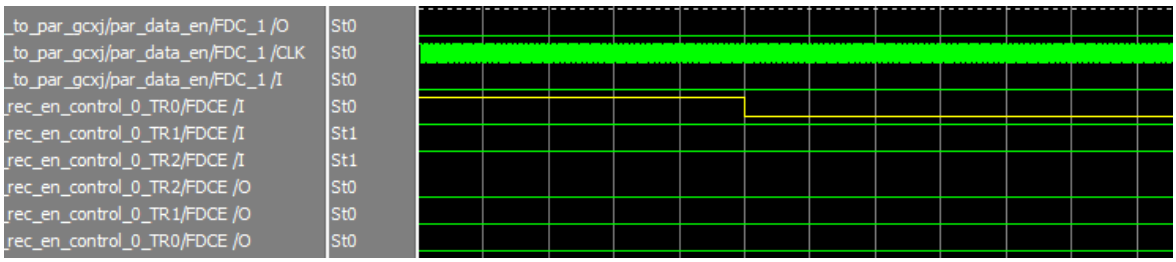



图5.6　　三模冗余验证结果

根据以上时序仿真波形可以看出，在仿真验证一段时间后，利用 Modelsim 工具的 force 命令将 GCXJ_rec_en_control 信号的一模强制置为低电平，如上图所示，而其余两模信号保持原值，验证的结果输出信号均为高电平，最终输出是正确的，说明本设计中的三模冗余处理是有效的。同时，可以看到：发生翻转的一模信号的输出同样是正确的，正式因为本次采用的是改进后的三模冗余处理方法。

## 5.4 图像源异常容错测试

图像源异常容错设计主要是为了保障输入压缩系统的图像的有效性以及图像规格的正确性，即对图像帧头巴克码的有效性检测和对图像帧长度的检测。对于输入的两种图像大图图像数据和小图图像数据，分别有不同的规格和巴克码。具体内容以及测试结果如下。

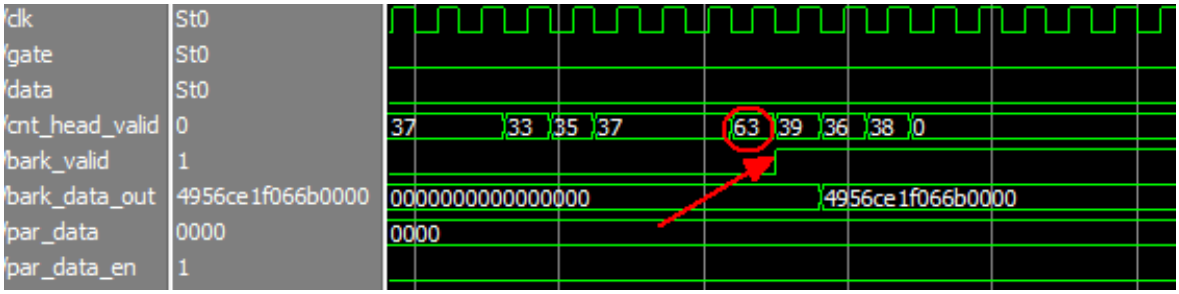
### 5.4.1 大图图像源异常测试

（1）帧头容错验证及结果

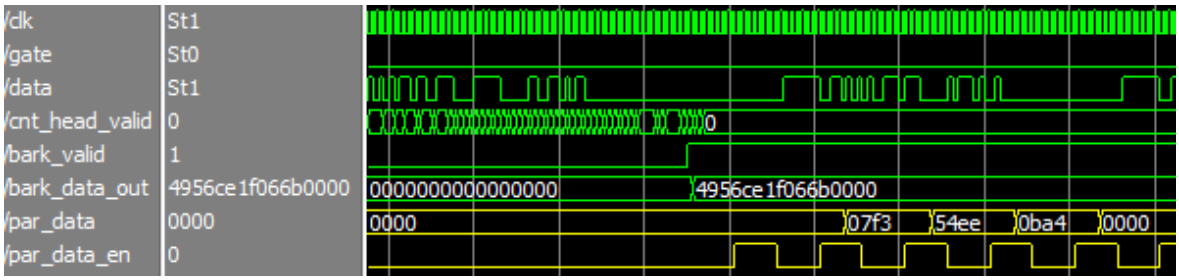
大图图像帧头巴克码数据位于图像的第一行，为 0x4954CE1F066B0000，共 64bit。输入图像后，首先对图像进行巴克码检测，当检测到有效的帧头信息，则电路判定是一帧有效图像输入的开始，否则一直处于检测状态，对当前的输入不做任何处理，视为无效数据。

帧头巴克码共 64bit 数据，如果能够检测到不少域 60bit 数据有效，则判定有效。测试的输入图像帧头分别有 1，2，3，4 和 5bit 错误，进行测试。结果如下：

a）巴克码 1bit 错误，测试结果：




图（a）

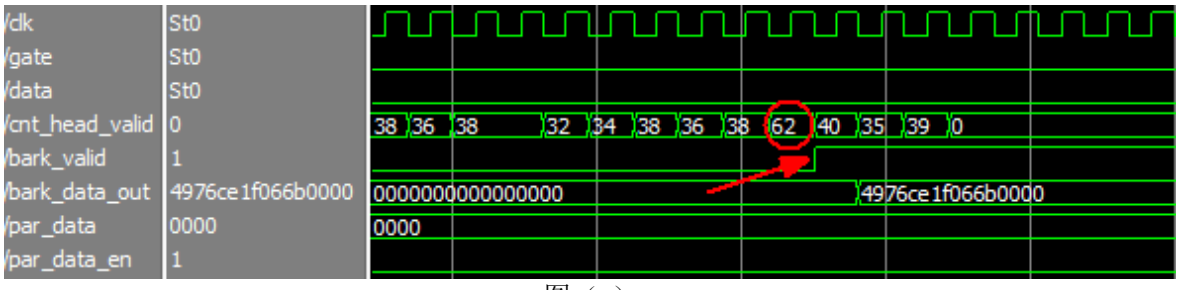



图（b）

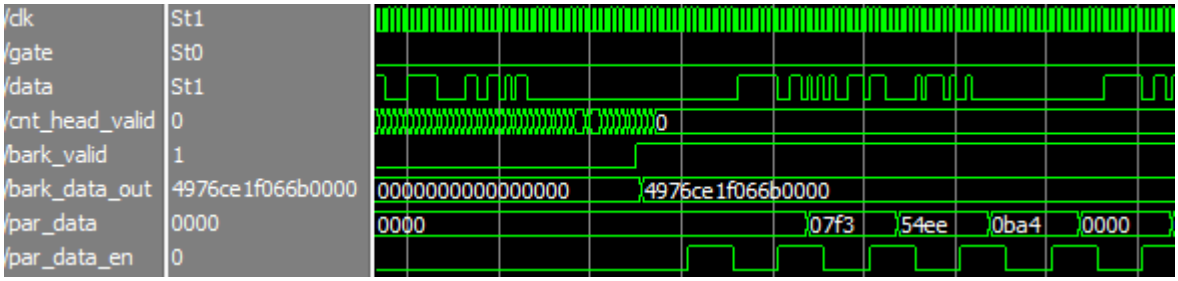
图5. 7　　大图帧头 1bit 错误测试结果

根据上图测试结果可知，当输入图像帧头巴克码数据有 1bit 错误时，检测到的有效比特数为 63，在可接受的范围内，故判定帧数据有效，帧头有效信号置为高电平，进而读取到图像像素值，如上图中所示。

b）巴克码 2bit 错误，测试结果：




图（a）

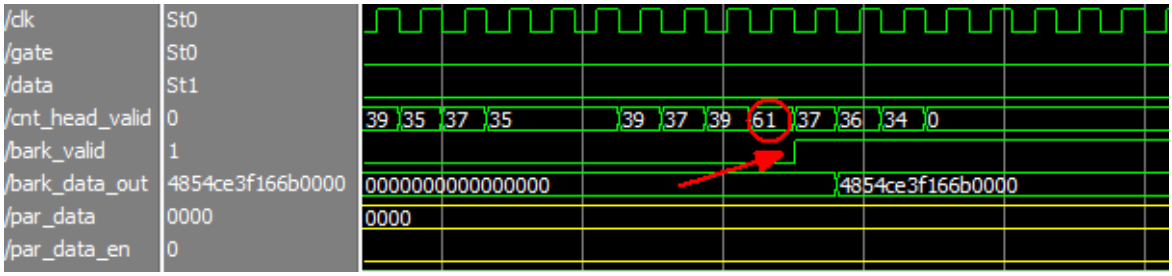



图（b）

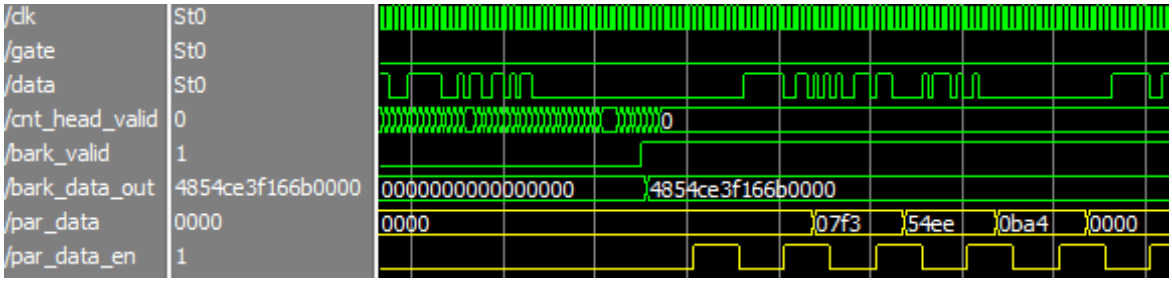
图5.8　　大图帧头 2bit 错误测试结果

根据上图测试结果可知，当输入图像帧头巴克码数据有 2bit 错误时，检测到的有效比特数为 62，在可接受的范围内，故判定帧数据有效，帧头有效信号置为高电平，进而读取到图像像素值，如上图中所示。

c）巴克码 3bit 错误，测试结果：




图（a）

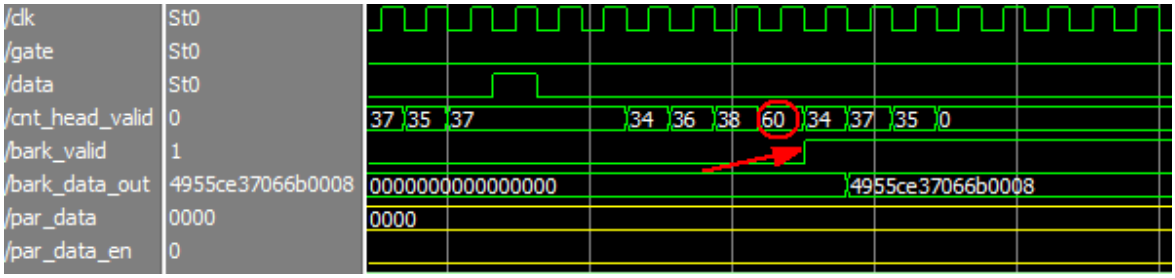



图（b）

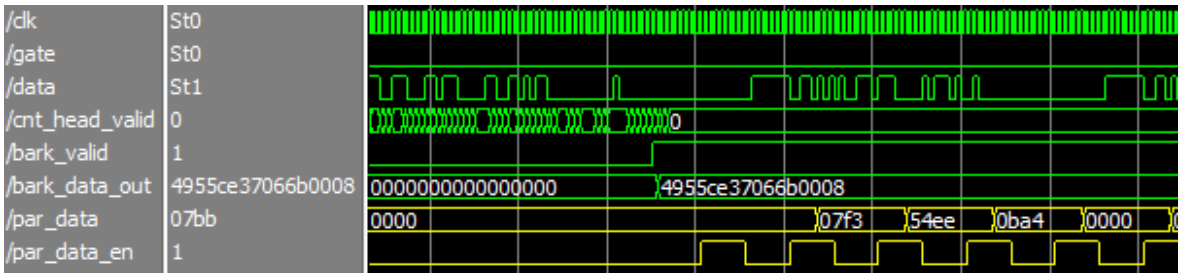
图5.9　　大图帧头 3bit 错误测试结果

根据上图测试结果可知，当输入图像帧头巴克码数据有 3bit 错误时，检测到的有效比特数为 61，在可接受的范围内，故判定帧数据有效，帧头有效信号置为高电平，进而读取到图像像素值，如上图中所示。

d）巴克码 4bit 错误，测试结果：




图（a）




图（b）

图5. 10　　大图帧头 4bit 错误测试结果

根据上图测试结果可知，当输入图像帧头巴克码数据有 4bit 错误时，检测到的有效比特数为 60，在可接受的范围内，故判定帧数据有效，帧头有效信号置为高电平，进而读取到图像像素值，如上图中所示。

e）巴克码 5bit 错误，测试结果：

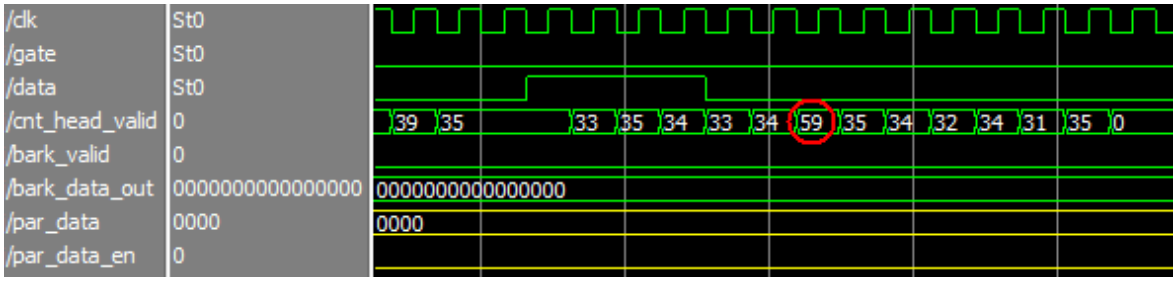



图5. 11　　大图帧头 5bit 错误测试结果

根据上图测试结果可知，当输入图像帧头巴克码数据有 5bit 错误时，检测到的有效比特数为 59，不满足对图像帧头的要求，故判定帧数据无效，帧头有效信号保持低电平，没有有效像素采集到。如上图中所示。

由以上测试的结果可以看出，当输入图像帧头巴克码中分别包括 1,2,3 和 4bit 的错误数据，系统能够正确的检测出，并且判定为有效的输入图像，将巴克码有效信号置为高电平有效，该信号是后续图像处理的使能信号，用于启动后续电路；当输入图像帧头有 5bit 错误数据时，系统判定的结果为有 59bit 有效，不满足设计要求，没有将帧头有效信号置高，视为无效数据。

（2）帧长容错验证及结果

输入的大图图像数据规格为 1072（列）*1028（行）*16bit，每个像素为 16bit。出去 4 个双字节的帧头，共 1102012 个像素点。当输入的图像数据量多于或者少于正常的一帧图像，则判定为图像帧长度异常。对大图图像帧长度的容错测试包括：

a）长帧：

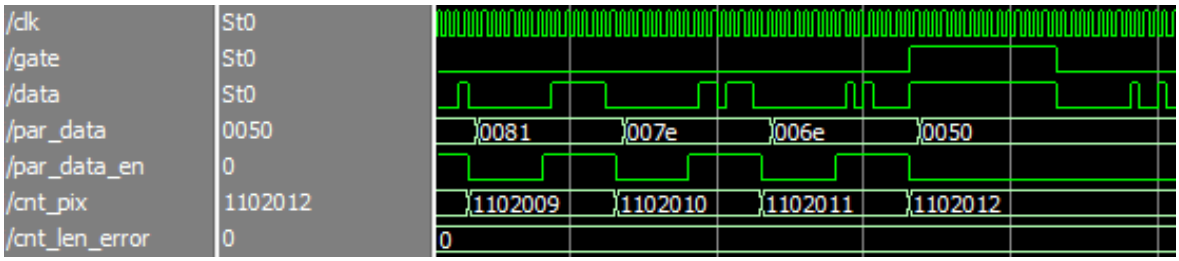
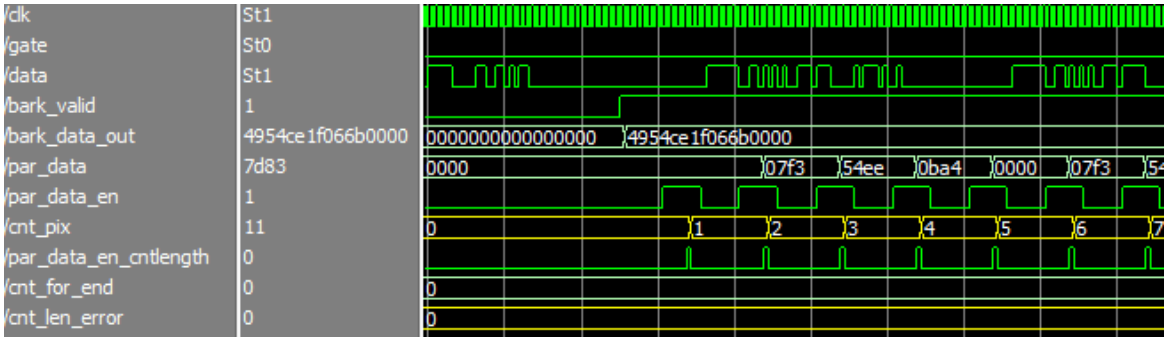


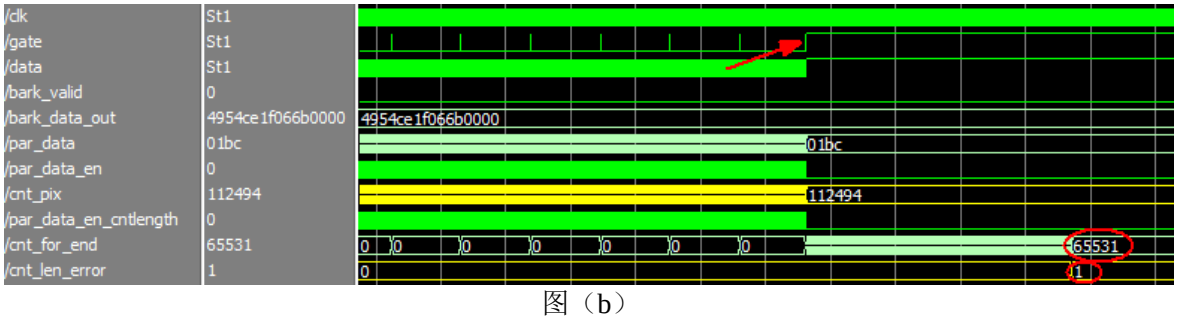

图5. 12　　大图长帧容错测试结果

输入的图像数据量多于正常的数据量，根据测试结果可以看出，输入的数据量（除去帧头 4 个像素的有效像素值）为 1102012，对图像进行截断处理，丢弃多于数据，并不更新帧长度错误计数。测试结果与设计一致。

b）短帧：




图（a）




图（b）

图5. 13　　大图短帧容错测试结果

根据上图测试结果可以看出，电路检测出的帧像素数量为 112494 个，不足一帧

图像的数据量，并且在图像输入结束后，等待 65531 个时钟周期，认为当前图像输入完成。此时根据像素值计数器中的值来判断当前图像是否长度异常，如图帧长度错误计数状态加 1。

由上述测试情况以及测试结果可以看出，当输入的图像数据量异常时，系统均能够检测出来异常。系统对图像源的异常处理比较可靠。

### 5.4.2 小图图像源异常测试

（1）帧头容错验证及结果

小图图像帧头巴克码数据位于图像开始的前几个字节，为 0xEB90，共 16bit。输入图像后，首先对图像进行巴克码检测，当检测到有效的帧头信息，则电路判定是一帧有效图像输入的开始，否则一直处于检测状态，对当前的输入不做任何处理，视为无效数据。

帧头巴克码共 16bit 数据，如果能够检测到不少域 15bit 数据有效，则判定有效。测试的输入图像帧头分别有 1bit 和 2bit 错误，进行测试。结果如下：

a）巴克码 1bit 错误，测试结果：

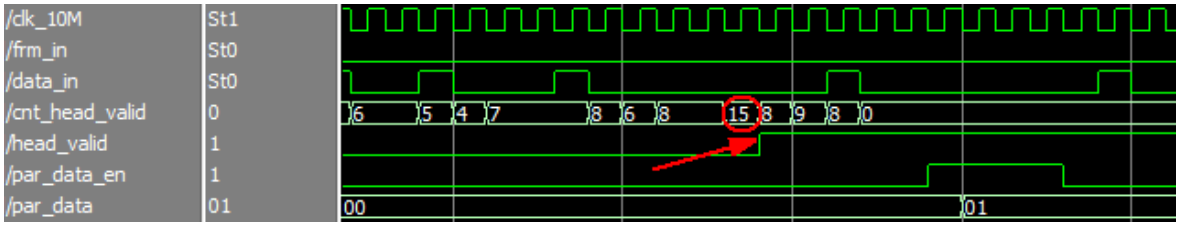



图5. 14　小图帧头 1bit 容错测试结果

根据上图测试结果可以看出，当小图帧头数据中包含有 1bit 数据错误，检测到的帧头有效数据为 15 位，此时电路判定输入图像数据有效，将帧头有效信号置为高电平，进而输出有效图像数据。

b）巴克码 2bit 错误，测试结果：

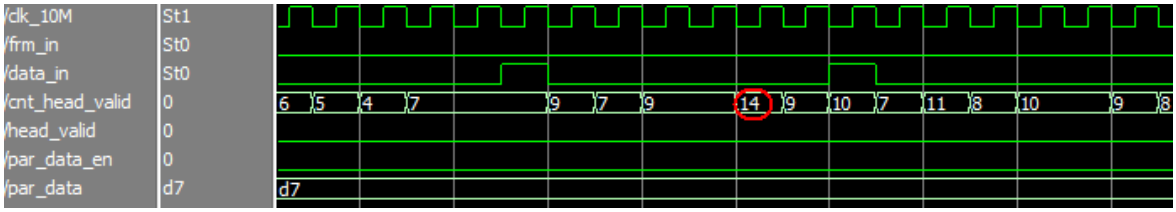



图5. 15　小图帧头 2bit 容错测试结果

根据上图测试结果可知，当输入图像帧头数据中包含 2bit 数据错误时，检测到的帧头有效数据为 14 位，与实际相符，但由于这种情况不满足设计要求，故判定为无

效输入数据。

由以上测试的结果可以看出，当输入图像帧头巴克码中分别包括 1bit 和 2bit 的错误数据，系统能够正确的检测出，并且判定为有效的输入图像，将巴克码有效信号置为高电平有效，该信号是后续图像处理的使能信号，用于启动后续电路；当输入图像帧头有超过 2bit 错误数据时，系统判定的结果为有 14bit 有效，不满足设计要求，没有将帧头有效信号置高，视为无效数据。

（2）帧长容错验证及结果

小图图像大小为 256 字节，出去 2 个字节的帧头，共有 254 字节的有效数据。当检测到有效的帧头数据时，启动字节计数器，对图像数据开始是计数，在输入的帧同步信号无效后结束计数，当此时字节计数值不是 254，判定为帧长度异常。

帧长度容错验证包括以下几种情况。

a）短帧：

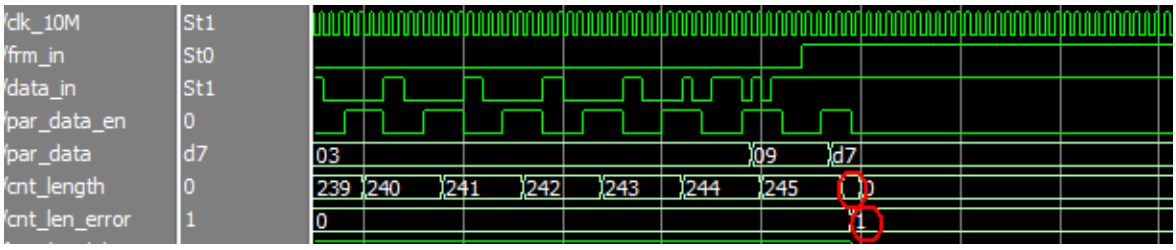
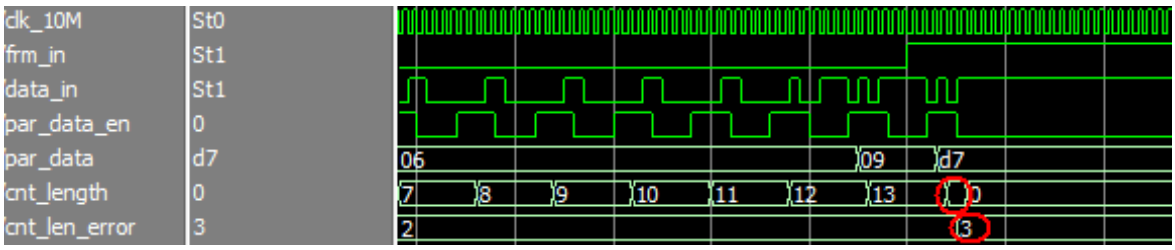


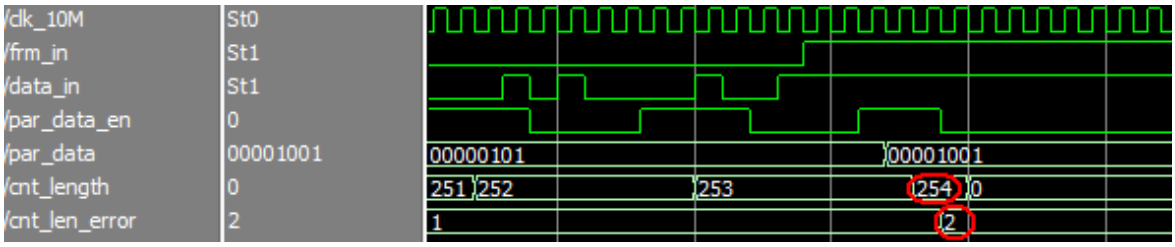

图5. 16　小图短帧测试结果

由上图测试结果可以看出，检测到的有效图像数据为 245 字节，图像数据量不足，判定当前输入图像为短帧，帧长度错误计数加 1。

b）长帧：




图（a）




图（b）

图5. 17　小图长帧测试结果

由上图中（a）图测试结果可以看出，帧长度计数器的值为 14，这是由于帧长度计数器为 8 位，长帧导致计数器溢出，实际检测到的有效图像数据为 270 字节，图像数据量超过正常一帧，判定当前输入图像为长帧，帧长度错误计数加 1。

由上图中（b）图测试的输入图像数据量为 512 字节，测试结果结果可以看出，帧长度计数器的值为 254，正常帧的有效长度为 254 字节，但是本次测试结果长度异常，帧长度错误计数加 1，有效的检测到这一异常图像源。未出现检测遗漏现象。检测电路工作可靠。

根据以上对图像长度异常情况的检测和测试结果可以看出，系统够有效的检测出小图图像长度的异常情况。

## 5.5 逻辑毛刺滤波验证

系统与相机的接口输入信号包括串行的图像数据信号、随路时钟信号以及门控信号。毛刺滤波主要是对门控信号的处理。当输入的门控信号中有异常的毛刺出现，通过滤波电路加以去除，保证门控信号的有效和稳定。在设计中，门控信信号中出现小于 16 个时钟周期的有效电平，均认为是毛刺。测试情况如下：

（1）1 个时钟的有效电平

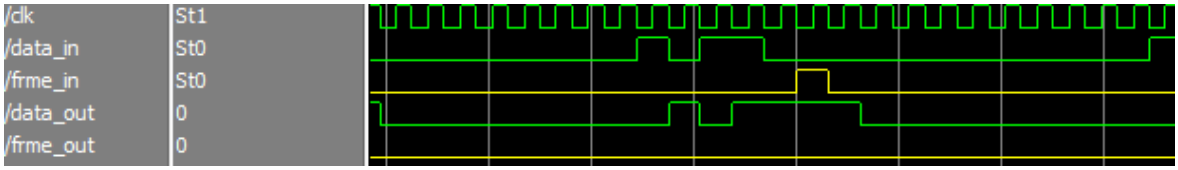



图5. 18　输入门控信号出现 1 个时钟毛刺

由上图测试结果可以看出，当输入的门控信号中含有 1 个时钟宽度的毛刺脉冲，通过滤波电路后，输出的门控信号中不再有毛刺现象，毛刺滤波电路有效。

（2）8 个时钟的有效电平

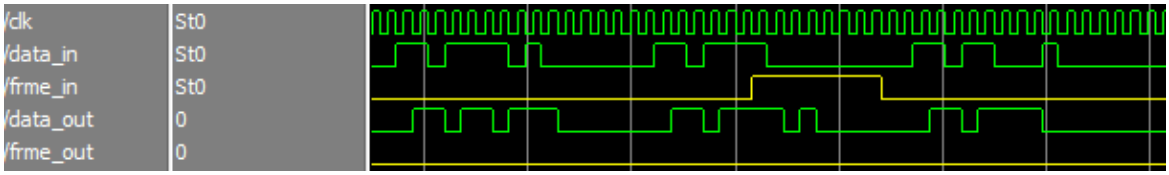



图5. 19　输入门控信号出现 8 个时钟毛刺

由上图测试结果可以看出，当输入的门控信号中含有 8 个时钟宽度的毛刺脉冲，

通过滤波电路后，输出的门控信号中不再有毛刺现象，毛刺滤波电路有效。

（3）15 个时钟的有效电平

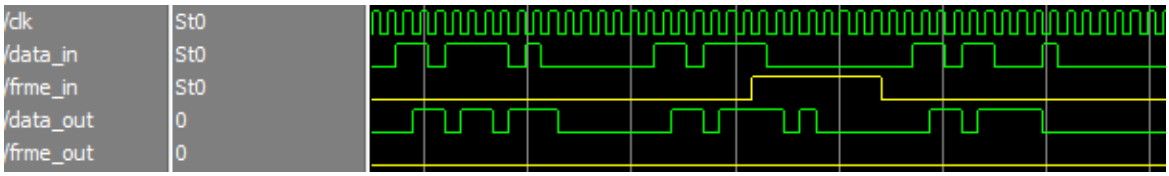



图5. 20　输入门控信号出现 15 个时钟毛刺

由上图测试结果可以看出，当输入的门控信号中含有 15 个时钟宽度的毛刺脉冲，通过滤波电路后，输出的门控信号中不再有毛刺现象，毛刺滤波电路有效。

（4）16 个时钟的有效电平

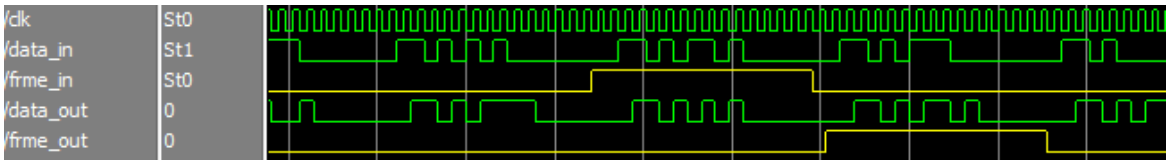



图5. 21　输入门控信号出现 16 个时钟毛刺

由上图测试结果可以看出，当输入的门控信号中含有 16 个时钟宽度的毛刺脉冲，通过滤波电路后，输出的门控信号将毛刺脉冲保留，这是由于 16 个时钟周期为行逆程的最小值，电路判定为逆程。与设计一致。

通过以上测试情况以及结果可以看出，当相机接口输入门控信号中含有不同长度的毛刺，当超出设计容限，也没有误过滤。系统对毛刺的处理有效可靠。

## 5.6 本章小结

本章是在前文对系统可靠性分析和设计的基础上，验证这些设计的有效性。首先对 JPEG-LS 图像压缩算法的性能进行了验证，验证结果满足设计要求；其次，通过三模冗余技术来避免星载系统在太空环境中的 FPGA 器件的单粒子效应，并验证了其有效性；再者，根据系统的实际情况进行了图像源容错设计验证、单帧独立验证、逻辑毛刺滤波验证。从验证和测试结果可以看出，以上所有可靠性设计均是有效的。

# 第六章 总结和展望

近几年，我国航天事业正在迅速发展，尤其以嫦娥系列受到了极大的关注。本设计的背景为某型号卫星上的图像压缩系统，用于对卫星采集的图像数据进行不同倍数的压缩与传输。

本文简单的介绍了几种常见的图像压缩算，包括 JPEG、JPEG2000 和 JPEG-LS，分析了以上几种算法的特点，进而指出了 JPEG-LS 算法对于深空探测系统的的独特优势。本文进一步研究了 JPEG-LS 图像压缩算法的原理，在此基础上行，利用 Xilinx 公司的 Virtex 系列芯片进行了算法的实现，并且在设计过程中进行了避免太空单粒子效应的三模冗余设计和其他可靠性设计。最终完成了基于 JPEG-LS 图像压缩算法的整个 FPGA 硬件系统设计。

本文所做的工作主要包括以下几个方面:

（1）通过对 SRAM 型 FPGA 进行深入的学习，详细介绍了其内部结构以及工作原理。从元器件结构角度出发，分析了 SRAM 型 FPGA 产生单粒子翻转的原因和和集中常见的单粒子效应，在此基础上介绍了避免单粒子故障的措施。

（2）详细分析了 XX-XX 项目的设计要求，对图像处理系统过程中可能出现的集中异常情况进行了容错设计，主要包括预防单粒子效应的三模冗余处理方法、对输入接口图像源异常的容错处理机制、输入接口门控信号的逻辑毛刺过滤电路设计和图像处理过程中避免错误积累的单帧独立设计。

（3）研究了 JPEG-LS 图像压缩算法原理，在熟悉原理的基础上，进行了算法的硬件设计，根据项目需求，最终实现了算法对图像的几种不同倍数的压缩处理功能。整个 FPGA 系统的三模前后资源占用均满足设计要求。

（4）针对项目中不同的功能性能等的要求，设计了完备的系统测试平台（TestBench），并对系统进行了前后仿真验证，验证结果满足设计要求。重点对本文中提到的集中可靠性设计进行了验证分析，结果满足要求。

本论文可以说是本作者研究生阶段科研过程中的总结，但是由于水平有限，再加上各种客观因素，所做的工作还可能存在缺陷，部分方面还需要进一步加深研究和学习探索：

（1）在系统仿真测试环节，由于测试内容的繁杂，导致仿真验证耗费的时间过大，应该学习其他测试方法，比如用脚本语言实现仿真平台的调用，这样很大程度上会提高仿真验证的效率。

（2）整体来说，本系统完全满足当前背景下的设计要求，部分逻辑电路时序余量较小。如果系统需求变化，可能会出现时序问题，需要进一步进行时序分析。

# 参考文献


[1] 王忠明. SRAM 型 FPGA 的单粒子效应评估技术研究[D]. 北京: 清华大学, 2001.

[2] Wallace G K. The JPEG still picture compression standard. Communications of the ACM, 1991, 34(4): 30-44.

[3] 魏江力, 柏正尧等译. JPEG-2000 图像压缩基础、标志和实践. 北京: 电子工业出版社, 2004(4), 37-40.

[4] 胡文江, 徐潇潇. JPEG-2000 标准中压缩算法原理分析. 电信快报. 2005, 12: 35-39.

[5] 曹青, 吴乐南. 静止图像无失真编码的新标准 JPEG-LS[J]. 信息化研究, 1999(2):12-14.

[6] 邢克飞. 星载信号处理平台单粒子效应检测与加固技术研究[D]. 国防科学技术大学, 2007.

[7] 李云松. Xilinx FPGA 设计基础[M]. 西安电子科技大学出版社, 2008.

[8] 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，2012.6.

[9] 汤琦, 蒋军敏. Xilinx FPGA 高级设计与应用[M], 北京: 清华大学出版社, 2012.

[10] 王波. 星载图像压缩编码器可靠性技术研究与实现[D]. 西安: 西安电子科技大学. 2016.

[11] 邢克飞, 杨俊, 王跃科,等. Xilinx SRAM 型 FPGA 抗辐射设计技术研究[J]. 宇航学报, 2007, 28(1): 123-129.

[12] 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(5).

[13] 贺朝会, 李国政, 罗金生,等. CMOS SRAM 单粒子翻转效应的解析分析[J]. 半导体学报, 2000, 21(2): 174-178.

[14] 董丽凤, 李艳丽, 王吉源. CMOS 集成电路闩锁效应抑制技术[J]. 电子与封装, 2010.10(009)L:28-30.

[15] 谭新宇. 航天图像压缩芯片的抗辐照设计[D]. 西安电子科技大学, 2014.单粒子锁定

[16] 王长河. 单粒子效应对卫星空间运行可靠性影响[J]. 微纳电子技术, 1998(1):1-8.

[17] 丁朋辉. SRAM 存储单元抗单粒子翻转研究[D]. 安徽大学, 2017.

[18] 郑晓云, 陶淑苹, 冯汝鹏,等. SRAM 型 FPGA 抗单粒子翻转技术研究[J]. 电子测量技术, 2015, 38(1):59-63.

[19] 胡洪凯, 施蕾, 董暘暘,等. SRAM 型 FPGA 空间应用的抗单粒子翻转设计[J]. 航天器环境工程, 2014, 31(5):510-515.

[20] 韩建伟, 张振龙, 封国强,等. 单粒子锁定极端敏感器件的试验及对我国航天安全的警示[J]. 航天器环境工程, 2008, 25(3): 263-268.

[21] 童天成. 全三模冗余星载计算机系统设计与实现[D]. 东华大学, 2015.

[22] 聂永康, 雷杰, 李云松,等. JPEG-LS 近无损图像编码器 VLSI 结构设计[J]. 西安电子科技大学学报(自然科学版), 2016, 43(4):75-80.

[23] 王海荣. 一种易于硬件实现的 JPEG-LS 无损图像压缩算法[J]. 科技视界, 2014(17):22-24.

[24] 孙健, 任国强, 吴钦章. 基于自适应指数哥伦布编码的图像压缩算法[J]. 光学精密工程, 2013, 21(11):2973-2979.

[25] 魏亚辉. 基于 FPGA 的遥感图像 JPEG-LS 压缩算法的研究与实现[J]. 信阳农林学院学报, 2016, 26(2):107-110.

[26] Xilinx. UG156: Xilinx TMRTool User Guide[EB/OL]. 2007, 12.

[27] Xilinx. ISE 9.1i Quick Start Tutorial。

[28] 李晓雯, 陈新凯, 李国林,等. 低功耗全流水线 JPEG-LS 无损图像编码器的 VLSI 设计[J]. 清华大学学报:自然科学版, 2007, 47(10):1654-1657.

[29] 章若冰. 电路竞争冒险及其改进策略研究[J]. 电脑知识与技术, 2017, 13(7):195-196.

[30] 杨岩岩, 司倩然, 马贤颖,等. FPGA 设计中的亚稳态问题及其预防方法研究[J]. 飞行器测控学报, 2014, 33(3):208-213.

[31] 陈宇泽, 宋绪勇. FPGA 设计中的相关问题及处理对策[J]. 中国高新区, 2017(13).

# 致谢

本论文是在导师王柯俨副教授的悉心指导下完成的。感谢李云松教授，以及实验室的吴宪云副教授、雷杰副教授在论文选题阶段给予我的指导。感谢王老师在我研究生三年期间对我学习科研给予的耐心指导和付出的巨大心血。王老师渊博的知识、敏锐的前沿学术洞察力、诲人不倦的师者风范和平易近人的态度，令我受益匪浅。同时，王老师严肃的科研态度、严谨的治学精神、精益求精的工作作风为我学生生活树立了良好的榜样，也为我今后的工作提供了积极向上的动力，一直以来都在持续地关注着我的科研动向和进度，每周一次的科研工作总结，让我养成了对自己的科研和学习定期总结的习惯，极大的提高了学习和科研的效率。值此论文完稿之际，谨对王老师的辛勤培育和谆谆教诲表示最衷心的感谢!

感谢导师吴宪云副教授，在科研期间对我的指导和关怀，在吴老师的项目组里学到了很多，吴老师科学的问题分析能力给力我很大的启发，是我在遇到问题时能够很快抓住问题的本质，从而提出合理的修改方法。

感谢雷杰副教授，在我做科研项目过程中对我的悉心教诲和指导，每次遇到棘手的问题都能在雷老师的帮助下迎刃而解。

衷心地感谢实验室的吴成柯老师、张静老师、刘凯老师、郭杰老师和何刚老师，为实验室创造了良好的学术氛围和科研环境，在各位老师的指导下，我不断的进步和成长。

感谢实验室的赵静、王波、丁胜朋、聂永康等师兄师姐，在进入图像所后，在他们的支持和友好的帮助下，我才能不断的学习进步，不仅在科研上给予我思想的启迪、方向方法的指导，而且在生活中给予我很大的帮助；感谢同在项目组的郝丰达同学，在项目上的相互的帮助；感谢实验室的江从辉、王康康、王茹、罗晓红、赵琛、李斐然、刘丹、卢运华、阮肇夏、赵熹、孔菲菲、李华清、孟昱、郭展扩同级研三同学和师弟师妹们一起营造的良好学习生活氛围，和大家在一起开心学习，开心生活。

特别感谢中科院西安光学精密仪器研究所的贺天兵，中科院空间中心的张学全、解彦以及三 0 四所的赵静工程师对我工作科研上的支持和帮助。

最后，感谢我的父母家人和女朋友，给予我的理解和支持，他们最无私的关怀和关心，是我不断前进的动力，使我对自己人生目标更加坚定和自信。


西安電子科技大學
XIDIAN UNIVERSITY

地址：西安市太白南路2号
邮编：710071
网址：www.xidian.edu.cn