中国科学院大学

University of Chinese Academy of Sciences

硕士学位论文

JPEG-LS 近无损图像压缩码率控制算法及其硬件实现

作者姓名：陈聪

指导教师：周盛雨 副研究员

中国科学院国家空间科学中心

学位类别：工学硕士

学科专业：计算机应用技术

培养单位：中国科学院国家空间科学中心

2018 年 6 月

硕士学位论文

JPEG-LS 近无损图像压缩码率控制算法及其硬件实现

作者姓名：陈聪

指导教师：周盛雨 副研究员

中国科学院国家空间科学中心

学位类别：工学硕士

学科专业：计算机应用技术

培养单位：中国科学院国家空间科学中心

2018 年 6 月

A Rate Control Algorithm of JPEG-LS Near-lossless Image Compression and Its Hardware Implemention

A thesis submitted to

University of Chinese Academy of Sciences

in partial fulfillment of the requirement

for the degree of

Master of Science

in Computer Application Technology

By

Chen Cong

Supervisor : Professor Zhou Shengyu

National Space Science Center, CAS

June 2018

摘 要

随着航天技术的飞速发展，利用卫星所获得遥感图像的数据量呈几何级增长。而星载硬件系统资源及传输带宽均十分有限，为了缓解数据存储及传输的压力，开发一套高效的星载图像压缩系统十分重要。

JPEG-LS 是 1998 年 ITU 基于 HP 实验室的 LOCO-I 算法所提出的一种无损/近无损图像压缩算法。该算法基于预测的编码方式，复杂度低，性能良好，硬件实现简单，适合星载硬件系统这一资源有限的场景。但它也存在码率不可控制的缺点，这也限制了 JPEG-LS 的进一步应用。为了实现基于 JPEG-LS 的码率可控的星上图像压缩模块，需要设计出针对 JPEG-LS 的码率控制算法。

本文在分析了 JPEG-LS 算法标准的基础上，结合现有的码率控制方法，设计出一种基于动态码率表的码率控制算法。测试表明，与现有方法相比，该算法收敛速度快，且压缩后重建图像质量更高。在该算法基础上，本文使用 Verilog HDL 硬件描述语言在 FPGA 平台上设计并实现了 JPEG-LS 近无损图像压缩模块。较之现有的设计，该模块编码效率更高，对图像的处理速度可达 60M Pixel/s。

关键词：JPEG-LS，星载图像压缩，码率控制，FPGA

Abstract

With rapid development of space technology, the quantity of images acquiring from satellite grows faster than in the past. However, the hardware resource and transmission bandwidth of satellite is limited. It is important to design a high performance satellite image compression to make it easy for storing and transmitting satellite images.

JPEG-LS is a lossless/near-lossless image compression standard based on LOCO-I algorithm proposed by HP laboratory. This algorithm accomplishes image compression based on prediction, and performs well in satellite image compression. It is easy to implement in low complexity on FPGA. Unfortunately, its bit rate is hard to control, which limits its use. So it is necessary to propose a bit rate control algorithm for the compression method.

A bit rate control algorithm based on a dynamic bit rate table is proposed after a detailed analysis of JPEG-LS image compression standard. Test shows that this algorithm can restrict the bit rate to the desired value rapidly in the progress of image compression and get a high-performance in the reconstruction image compared to the methods in the state of art. Moreover, a JPEG-LS near-lossless image compression system described in Verilog HDL is proposed on FPGA based on the algorithm. It promotes the compression efficiency. The system can achieve a speed of 60M Pixel/s.

Key Words: JPEG-LS, Satellite image compression, Bit rate control, FPGA

目 录

图目录

表目录

第1章 引言

1.1 研究背景

视觉是人类感知世界的重要手段，统计资料表明，人类获取信息的 70%来自视觉系统[1]。视觉能够让人更加直接地了解世界，认识世界，而图像作为视觉认识世界的结果，对人类来说便也更加重要。而随着计算机技术和网络技术突飞猛进地发展，人类获取和处理信息的方式也发生了革命性的改变，图像能够以数字化的形式存储在计算机系统中，并在网络空间中自由传输，这就使得图像的使用范围更加广泛，人们对图像处理的需求也随之不断增加。由于数字化图像的清晰度越来越高，数量越来越大，图像压缩算法的研究变成了当务之急。

图像压缩是通过一些适当的信源编码算法，在图像质量在可接受范围内的前提下，用尽可能少的数据量来表示数字化图像的过程。图像数据之所以存在可以被压缩的空间，一方面是因为在一幅特定图像中存在着局部相关性，即在图像的某个区域内其像素值存在着某种相关关系，这使得用较少的数据量表示更多的图像信息成为可能，意味着在图像压缩的过程中图像中存在着一些冗余信息可以丢弃，而在恢复图像的时候，这些冗余信息可以近似地甚至无损地恢复；另一方面是人眼对于图像信息的感知精度存在着局限，在可忍受的范围内降低一些图像信号精度，既可以通过较少的数据量表示数字化图像，也不会降低人眼的视觉体验。

在星载图像压缩领域，随着遥感技术的不断发展，通过卫星平台获取图像变得更加便捷，卫星图像的时间以及空间分辨率也在不断提高，这就导致了遥感图像数据量呈几何级增长的态势，有些卫星的原始数据已经达到了 Gbits 的级别[2]。所获得星载图像数量及其数据量的日益庞大，使得有限的信道带宽和数据存储容量与传输大量遥感数据之间的矛盾日益突出，这使得数据压缩技术尤其是遥感图像数据压缩的研究变得非常重要[3]。一般而言，图像分辨率越高就意味着该图像中存在着更多的局部相关性，可以丢弃的冗余信息量也越大。

压缩卫星图像是节省通信信道和存储容量、提高信息的传输和存储速率的必然要求；同时数据压缩的过程也是数据加密的过程，有利于数据传输过程的保密性[4]。

1.2 星载图像压缩的发展

卫星图像的压缩比起一般图像更加困难，因为卫星图像的局部相关性较之一般图像更弱，且使用者感兴趣的信息较多，导致可以丢弃的冗余信息较少，为星载图像压缩系统选取合适的图像压缩算法就变得尤为重要。星载图像压缩系统所采用的主流压缩算法有：差分脉冲调制编码（DPCM）、矢量量化（VQ）以及变换编码（如 DCT、小波变换），在实际应用中，法国的遥感卫星 SPOT2~4 采用了 DPCM 方法，其中，SPOT4 的压缩比约为 1.5 倍，SPOT5 采用了 DCT 方法，压缩比约为 3 倍。但是实验表明，当压缩比超过 3 的时候，DPCM 与 DCT 方法所压缩的图像质量会明显下降 [5]。而矢量量化方法在压缩比较高的情景下表现更好，美国的“Eyeglass”卫星正是采用了这种方法达到了 12 倍的压缩比。但是，矢量量化方法硬件开销很大，星载压缩系统很难将这种算法高效实现。

进入新世纪后，基于小波变换的图像压缩算法逐渐兴盛起来。2004 年美国的勇气号和机遇号探测器是最早使用者之一，2009 年法国的 Pleiades 采用了 JPEG2000 作为图像压缩的核心算法，而 JPEG2000 正是基于离散小波变换(DWT)实现的。在国内小波变换也逐渐应用到星载图像压缩领域，2010 年发射的嫦娥二号卫星的星载图像压缩系统中采用了基于小波变换的 SPIHT 算法。

1.3 星载图像压缩算法简介

1.3.1 基于预测的编码方式

在图像中一般存在着局部相关性。在对单个像素值进行编码时，可以通过对该像素的相邻像素值采取某种预测算法得到预测值，再将预测值与实际像素值作差，对所得误差值进行编码。采取适当的预测方法和编码方法，便可以用更少的码字进行像素值编码。比起对单像素进行直接编码，对像素值进行预测后再与实际像素值作差，然后对该差值进行编码效果会更好[6]。在预测编码方

法中，最主要的方法是差分脉冲编码。

基于预测的编码方式最大的优点是算法简单，硬件实现的复杂度低，适用于星载图像压缩等硬件资源有限的场合。尤其是在无损压缩领域，基于预测的编码方式具有很大的实用价值；而其也可以用于有损压缩中，通过对预测值与实际值之间误差的量化，将误差的值集中在更方便编码的区间内，可以适应较低倍率的有损压缩。

这种方法也有一定的缺点。首先，该方法的压缩码率波动较大，不同类型的图像用同种预测方法进行压缩效果可能千差万别，一般情况下，局部相关性较弱的图像，当使用预测方法进行时压缩效果较差。同时，算法中的预测器对不同类型的图像适用性也不尽相同，若采用较为简单的预测器适用的图像类型就会变少，而采用复杂预测器的话就会导致算法复杂度的增加。而该方法的另一缺点就在于对误码较为敏感。该方法压缩后的图像在解码时依赖于上下文，若某一像素点解码错误将会导致解码误差的累加，造成严重的图像质量下降。由于基于预测的编码方式的上述特点，该方法更适用于无损压缩或者低倍率的有损压缩中。而其压缩码率不可控和抗误码性能较差，也需要设计算法进行改进。

JPEG 的无损/近无损压缩标准简称为 JPEG-LS，于 1998 年正式公布，用于取代原 JPEG 的连续色调静止图像无损压缩模式，也是一种基于预测编码的图像压缩标准[7-8]。JPEG-LS 存在两种不同的压缩模式：普通编码模式和游长编码模式。其中普通模式编码的基本思想是将待编码像素点已出现过的相邻的像素点，建立一个可以用来对待编码像素点进行预测的上下文模型，通过简单的预测和修正之后，得到预测值与实际值之差即为误差值，在通过对误差值进行映射使其全部为正后，通过 Golomb-Rice 来编码映射后的误差。而游长模式编码则是通过对具有图像中大块平滑区域进行压缩，从而达到大幅度降低压缩码率的一种算法，所谓平滑区域即图像中出现的大片像素值相同或相近的区域。JPEG-LS 图像压缩算法的最大优点就在于不需要进行复杂的变换，通过预测像素值与原像素值作差，再将差值进行编码即可完成压缩，易于通过硬件实现。

1.3.2 矢量量化编码（VQ）

矢量量化起源于 70 年代后期，它的方法是将若干个待压缩的标量数据拼接成矢量，然后在矢量空间将矢量进行整体量化，可以达到在不损失太多数据的前提下对矢量中的标量数据进行压缩的目的。在传统的基于预测方式和基于变换方式的编码算法中，都是对单纯的标量数据进行操作，使待编码的数据变得更加容易编码。而在矢量量化编码中，输入数据几个一组地分成许多组，每个组构成一个 k 维矢量，然后以矢量为单位逐个对每个矢量进行量化并进行编码。通过率失真理论对于矢量量化编码进行分析可以得出，矢量量化编码比起标量量化压缩效果要更好。

在矢量量化编码中，如何产生和搜索码本是算法的关键。码本是由特定算法生成的具有代表性的矢量集合，集合中的每一个矢量都有唯一对应的序号。在矢量量化图像系统中，编码和解码过程都需要对码本进行搜索，所以编码器中和解码器中的码本相同，保证了能够进行正确的解码过程。而码本是一组矢量的集合，若图像压缩需要高保真度，码本中所包含的矢量会更丰富；而当需要更简单的编码器结构时，则需要码本更精简且更具有代表性。矢量量化的码本是根据训练矢量集合来设计的，编码器根据特定的算法，在将输入图像的像素值分组为若干个矢量后，在码本中进行搜索，对这些矢量匹配，然后对匹配码的码本序号进行编码，从而通过一个码字序号就可以表示单个矢量中所包含的所有图像像素值的信息。

矢量量化可以进行较高压缩倍率的有损压缩，最大的优势在于解码过程简单，不需要复现编码过程，只要在码本中搜索接收到的码字序号，从而可以找到该序号所代表的矢量信息，便可在有限的误差范围内重构与原始图像相似的图像数据。而与解码器的简便相对的是编码器的复杂，编码过程中为了匹配待编码矢量而进行的码本搜索，其复杂度随着矢量维数 K 进行指数增长。如果要追求重建图像误差更小，势必增大码本的容量，对码本的搜索时间也会随着码本容量的增长而增长。另外，矢量量化的编码过程对原图像数据进行了分组操作，所以使用矢量量化编码压缩过并重建后的图像会出现方块效应，造成图像质量的下降。

对于矢量量化编码来说，关键的技术是权衡好码本容量与重建图像误差之

间的关系和对码本进行快速搜索的算法，都会带来较高的编码成本，这使其不太适用于星载数据压缩这一硬件资源有限的情境下。同时矢量量化编码的质量也取决于码本对所编码图像的适应性，适应性高会使编码误差更小。而对于类型繁多的遥感图像来说，很难找到合适的码本可以适应所有类型的遥感图像，若有也会导致码本搜索复杂度的硬件开销到无法接受的程度，所以这方面的研究也在逐渐减少。

1.3.3 离散余弦变换编码（DCT）

与基于预测的编码方式不同的是，变换编码是从频域的角度，通过某种变换关系，减小图像信号的空间相关性，然后用某种编码方式对变换后产生的系数进行编码。

离散余弦变换编码就是这样一种编码方式。与矢量量化编码对图像的预处理过程类似，它的预处理方式是将图像视作一个由若干 n*n 小正方形区域组成的整体，对每一个块使用离散余弦变换，就会得到 n*n 的变换系数块，最终对系数进行编码，完成整个压缩过程。

法国空间研究中心的研究表明，由于压缩前对图像划分区域的操作，达到一定级别的压缩比时，重建图像的块边界会出现较为明显的边缘，失真较为严重，重构质量较低[9]。而卫星图像细节较多，分辨率往往很大，需要比较高的重构质量来保证图像细节的留存，因此，离散余弦变换不适合这种类型的图像压缩。

1.3.4 离散小波变换编码

小波是二十世纪 80 年代后期迅速发展起来的一门数学分支，逐渐发展成为分析非平稳信号的有力工具。小波图像压缩的特点是压缩比高，能量损失低，图像的基本特征不会被破坏，且信号传递过程抗干扰性强，可实现累进传输，而提升小波结构的提出，使得它的资源占用比传统结构更少，得到了更广泛的应用[10-13]。

ISO 于 2004 年提出的静止图像的编码标准 JPEG2000，核心算法是 EBCOT 算法，就是基于离散小波变换(DWT)实现的[14-16]。该标准抗误码特性和压缩性能较高、在较大压缩比下重构质量良好。但其在硬件实现中占用资源较大，计

算复杂度高，不利于在硬件资源受限的条件下实现，所以在星载图像压缩领域中使用较少。2005 年，CCSDS 提出了新的空间图像压缩标准 122.0-B-1，图像预处理采用了三级二维 5/3 小波或 9/7 小波变换，可以通过截断码流的方式来实现对压缩比的控制。与 JPEG2000 相比，该算法复杂度低，可以用较少的硬件资源取得较好的图像压缩质量和压缩比均衡效果，在星载图像压缩平台中取得了广泛的应用。

1.4 本文主要工作及内容安排

JPEG-LS 图像压缩标准是一种基于预测及 Golomb 编码的图像压缩算法，具有复杂度低、效率高以及容易进行硬件实现等特点。同时它也具有其他基于预测的编码方式的通病，即码率不容易控制。为了使其更好地应用于星载图像压缩，需要设计一种行之有效的码率控制算法以及硬件实现的方案。

目前存在一些可控制码率的 JPEG-LS 近无损图像压缩方案，但或多或少都存在一些不足。本文将针对星载图像压缩要求，设计一种针对 JPEG-LS 的码率控制算法，同时在此算法的基础上，设计一套码率可控的 JPEG-LS 近无损图像压缩硬件实现方案。这将提高星载图像的压缩灵活性，还可以降低其功耗，使图像压缩系统满足星载系统对硬件资源的要求。

本文分为五章，各章内容安排如下：

第一章：引言部分，对课题的选题背景、星载图像压缩的发展进行了概述，介绍了几种主要的图像压缩算法，同时描述了本文的主要工作。

第二章：JPEG-LS 图像压缩标准介绍，介绍 JPEG-LS 算法各个部分的主要内容。

第三章：JPEG-LS 码率控制算法的研究与改进，对影响 JPEG-LS 码率的因素进行研究，在现有码率控制算法的基础上，提出一种新的码率控制算法，并对算法进行评估。

第四章：JPEG-LS 近无损压缩编码器的硬件实现，对 JPEG-LS 近无损压缩编码器的 FPGA 实现框架结构和实现方法进行了详细描述，并评估其资源占用和性能。

第五章：总结全文，分析并指出目前还存在的问题，并讨论下一步的研究

方向。

第2章 JPEG-LS 图像压缩标准介绍

JPEG-LS 是 ITU 基于 HP 实验室提出的 LOCO-I 算法提出一种无损/近无损图像压缩算法。该算法可以通过较低的算法复杂度实现高性能的无损/近无损图像压缩，从而在航天、医学等诸多领域取得了广泛应用。该图像压缩标准基于差分预测编码技术，采用 Golomb 编码对预测值与实际值的误差进行编码，同时建立了上下文模型对编码参数进行修正，具体的算法实现过程如图 2.1 所示：

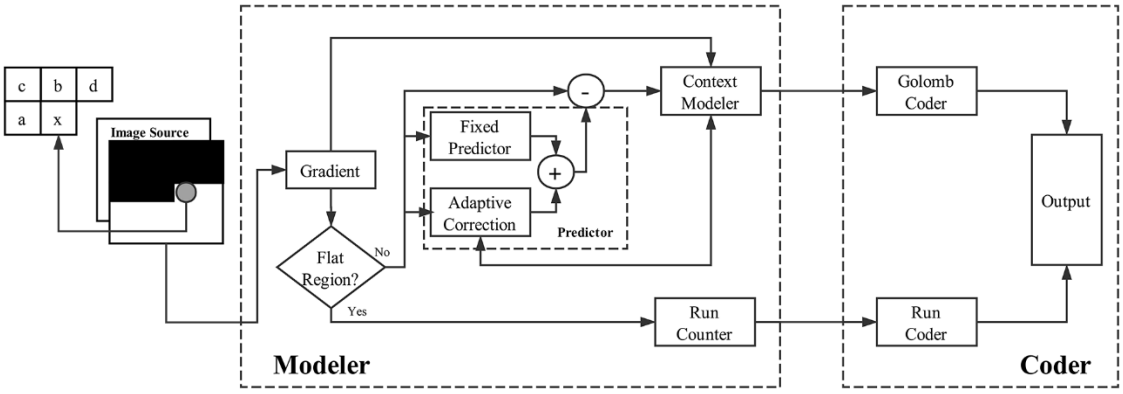

图 2.1 JPEG-LS 压缩算法框图

该编码器按行对像素值进行扫描并逐个进行编码，扫描至行尾时即转移到下行首个像素继续进行编码，直至遇到图像末尾。算法会在每次编码开始前，会对图像已出现的像素值进行梯度计算，得出一个上下文模式，而后根据图像处于何种上下文模式判断采取何种方式进行编码。在普通编码模式下，编码器首先采用简单的边缘检测器根据待编码像素的上下文模式进行预测，可以得到待编码像素的预测值，校正后与实际值作差即可得到误差值，该预测误差的分布满足双边几何分布[17]。最后对预测误差进行校正、映射等处理后，对其进行 Golomb 编码，即可得到最终的压缩码流[18]。而当梯度计算后，待编码像素点已出现的上下文像素值完全相等时，即可开始进行游长扫描。当遇到游长模式中断的条件导致扫描中断后，需要对游长计数进行编码，如果扫描中断是由于遇到中断点而导致的，则需要对中断点进行 Golomb 编码。

2.1 上下文建模

在 JPEG-LS 图像压缩标准中，上下文模型是算法的基础[19]。由于 JPEG-LS 图像压缩标准采用行扫描的方式进行编码，在上下文建模时，对每一个待编码像素点的处理都要参考编码过程中已出现过的上下文信息，即当前像素点的右上、上、左上以及左侧的像素点作为编码的前提条件。待编码像素点 x 及其编码所需的上下文信息的位置关系示意图如图 2.2 所示。

c	b	d
a	x	

图 2.2　JPEG-LS 压缩算法的上下文模型

编码器在图 2.2 所示上下文模型的基础上进行上下文梯度的计算，来判断对待编码像素点 x 进行编码时采用普通模式还是游长模式。

2.1.1 梯度计算

为了判断当前像素点处于何种上下文模式下，需要按照上述上下文模型和公式(2.1)所示的计算方式，计算上下文梯度值 D_1、D_2 和 D_3。其中I_a、I_b、I_c、I_d分别代表各个点的像素实际值。

$$\begin{cases} D_1 = I_d - I_b \\ D_2 = I_b - I_c \\ D_3 = I_c - I_a \end{cases} \tag{2.1}$$

2.1.2 编码模式选择

通过梯度计算，编码器将得到梯度值 D_1、D_2 和 D_3，此时需要编码器决定压缩模式。在无损压缩的情景下，如果所得三个梯度值均为 0，可认为编码器进入大片像素值相同的“平坦”区域，因此将会选择游长模式开始进行编码。后续将首先进入扫描模式，即以游长模式开始时上下文中 a 点的像素值 R_a 为参考量，若后续像素点的像素值与参考量相等则一直扫描下去，无需进行梯度计算。而当遇到不等的像素点或行末尾时中断扫描进入游长编码，此时对中断点将进行 Golomb 编码，中断点后的像素点则应重新进行梯度计算，之后判断应选择何种模式进行编码。

如果所得三个梯度值 D_1、D_2 和 D_3 均不为 0，则应选择普通模式进行编码。在进入普通编码模式时，接下来将进行梯度量化、Q 值计算、边缘检测以及后续

对误差的处理和编码等步骤，完成对一个像素点的编码。

2.2 普通编码模式

当编码器进入普通模式时，首先要进行上下文信息的处理，包括梯度的量化及 Q 值的计算，Q 值是对图像上下文模式索引，每个 Q 值对应一种上下文模式。之后便根据上下文信息对待编码像素值进行预测，再由实际像素值与预测像素值作差得到误差值，对误差值进行校正和映射后会得到一个值，最后对该值进行编码得到压缩码流。在 JPEG-LS 中，对待编码像素点的预测是基于一种简单的边缘检测器，而最后的编码则是限长 Golomb 编码。

2.2.1 梯度的量化

在普通模式下，需要对梯度计算所产生的三个梯度值进行一定程度的量化，来使图像的上下文模式更加集中。虽然更多的上下文模式将会使对待编码像素点的预测更加精确，但这样做也存在着一些缺点：一方面，大量的上下文信息下，平均每个上下文情况对应的已编码像素数量很少，在这种情况下收集过多的上下文信息便失去了意义；另一方面，如果上下文信息有很多，就需要更大的存储空间来存储上下文参数，这也提高了算法的资源占用。

为此，JPEG-LS 算法标准推荐使用三个非负的阈值 T_1、T_2 和 T_3 对上一步骤中产生的梯度值进行量化。无损压缩模式下，压缩 8bit 图像时，这三个阈值一般分别取值为 3、7 和 21，若压缩其他种类的图像时，阈值一般根据公式(2.2)所示的方式进行设置。

$$\begin{cases} T_1 = (bpp - 7) * 3 + near \\ T_2 = (bpp - 7) * 7 + 2 * near \\ T_3 = (bpp - 7) * 21 + 3 * near \end{cases} \tag{2.2}$$

其中，bpp 代表图像中存储每个像素值需要的位数，而 near 为失真量化参数。

公式(2.3)展示了一种对梯度值 D_1、D_2 和 D_3 的量化方式，其中 Q_i 为梯度值 D_i 所对应的的量化值。

$$Q_i = sign(D_i) * \begin{cases} 0, D_i = 0 \\ 1, 0 < |D_i| < T_1 \\ 2, T_1 \le |D_i| < T_2 \\ 3, T_2 \le |D_i| < T_3 \\ 4, |D_i| \ge T_3 \end{cases} \tag{2.3}$$

如果 D_i 不大于-T_3，则 Q_i 等于-4；若 D_i 大于-T_3 而小于-T_2，则 Q_i 等于-3；若 D_i 大于-T_2 小于-T_1，Q_i 等于-2；若 D_i 大于-T_1 小于 0，Q_i 等于-1；若 D_i 等于 0，Q_i 等于 0。之后的以此类推，直至 D_i 大于 T3，可得 Q_i 等于 4。这样就完成了梯度量化的工作。

2.2.2 量化梯度的合并及 Q 值计算

在产生上下文模式矢量(Q_1，Q_2，Q_3)后，上下文模式的数量大大减少了。然而，此时上下文模式还有进一步压缩的空间。统计规律发现，(Q_1，Q_2，Q_3)和(-Q_1，-Q_2，-Q_3)所对应的上下文模式具有相似的特性，所以 JPEG-LS 标准对这类上下文模式进行了合并。

当矢量(Q_1，Q_2，Q_3)的第一个非零元素为负时，则该矢量全部取反。同时定义一个变量 SIGN，若出现上述情况，令 SIGN 的值等于-1；若没有出现，则令 SIGN 的值为 1。

在完成上下文模式合并之后，便可以将矢量(Q_1，Q_2，Q_3)映射成唯一的整数 Q，表示待编码像素点 x 上下文模式的索引。一般情况下会将该矢量映射到一个连续的非负整数区间中。本文采用公式(2.4)所示的映射函数：

$$Q = (Q_1 * 9 + Q_2) * 9 + Q_3 \tag{2.4}$$

根据公式(2.3)中关于梯度值的量化情况，该映射函数的选取可以使矢量(Q_1，Q_2，Q_3)映射到[0,364]的区间内，且区间内的任何一个整数都有可能被映射到。该映射函数计算简单，只采用乘法和加法，有利于进行硬件实现。

2.2.3 边缘检测

JPEG-LS 为了方便实现，对待编码像素值的预测只是基于简单的边缘检测，边缘检测也是基于图 2-2 所示的上下文模型的。该边缘检测器假定在像素点 a、b、c 和 x 之间存在水平、垂直或者斜 45° 方向的边缘。若像素点 c 的值 R_c 大于像素点 a 的值 R_a 和像素点 b 的值 R_b，则认为边缘存在，预测 x 点的像素值为 R_a 和 R_b 中的最小值。反之若 R_c 小于 R_a 和 R_b 中的最小值，则预测 x 点的像素值为 R_a 和 R_b 中的最大值。而当 R_c 位于 R_a 和 R_b 之间时，则认为该上下文模型中不存在边缘，假设不成立。由于没有边缘时像素值不会有剧烈的变化，所以预测 x 点的像素值为 R_a+R_b-R_c，这样保证了 x 点的像素值也位于 R_a 和 R_b 之间，提高预测的准度。

2.2.4 预测值的校正

边缘检测完成之后，编码器会得到一个关于待编码像素点的预测值 Px，此时需要对 Px 进行校正。校正值 C[Q]是通过统计索引为 Q 的上下文模型下已被压缩过的像素点中得到的某些统计信息计算得出的。同时，校正过程需要调用在上下文建模中 Q 值计算检测出的变量 SIGN。预测值校正过程如公式(2.5)所示。当校正后的预测值超出图像的最大像素值范围或小于 0 时，则直接令预测值等于图像的最大像素值或 0。

$$Px' = Px + SIGN * C[Q] \tag{2.5}$$

之所以进行该步骤，是因为预测过程总会存在着误差。为了减轻系统偏差的影响，通过使用每个上下文模型独特的偏差值 C[Q]对预测值进行校正，使预测误差绝对值集中在 0 值附近，近似满足正态分布。

2.2.5 预测误差的计算

将待编码像素点的实际像素值 Ix 与公式(2.5)得到的预测值 Px 作差即可得到预测误差 Errval，同时注意若符号变量 SIGN 为负，预测误差 Errval 需要取相反值。在近无损压缩模式下，可以通过失真量化参数 near 对预测误差 Errval 进行量化，使预测误差 Errval 的绝对值范围进一步缩小，方便进行编码。近无损压缩模式下，量化误差公式如公式(2.6)所示。

$$Errval' = \frac{Errval + near}{2*near+1} \tag{2.6}$$

而同时通过公式(2.7)，对误差值进行重建，得到一个重建像素值 Rx，用于其右侧及下行像素点的压缩，这时重建像素值 Rx 与原像素值的误差在±near 范围之间。

$$Rx = Px + SIGN * [Errval * (2 * near + 1) - near] \tag{2.7}$$

2.2.6 预测误差的模减与映射

预测误差的范围为±(Range-1)，其中 Range 为图像中像素最大值加 1 得到，由于图像中像素的表示区间为[0,Range-1]，实际上长度为 Range 的区间便可以表示所有的预测误差。所以我们可以通过对预测误差进行模减的方式，将误差折叠到(-Integer（RANGE/2)，Integer（(RANGE+1) /2））范围内，这样可以使误差范围进一步收缩，方便编码。

当预测误差 Errval 小于-RANGE/2，便将预测误差 Errval 加上 RANGE 值。在完成这步处理后，若预测误差 Errval 大于等于(1+RANGE)/2，则需要将预测误差 Errval 减去 RANGE 值。经过处理后，所有预测误差 Errval 的值的绝对值都不会大于（RANGE+1) /2。

之后是对模减后误差的映射。一般情况下，会将负值和正值分别交叉映射到[0, RANGE)区间范围的偶数和奇数上。这样使待编码的值全部为正，方便进行编码。

2.2.7 编码参数的更新

JPEG-LS 算法标准中，每一种上下文模式对应一组上下文参数 A[Q]、B[Q]、C[Q]、N[Q]来保存相应的上下文信息，在编码过程中起到重要的作用。其中 A[Q]是对预测误差绝对值之和，B[Q]是对预测误差重建值之和，B[Q]的作用是更新偏差值 C[Q]。C[Q]在 2.2.4 节中已经介绍过，该值作用在边缘检测后得出预测像素值 Ix 后，对其进行修正。N[Q]则是上下文模式计数器，当索引 Q 对应的上下文模式出现时，N[Q]即+1，该参数和 A[Q]一同负责估计 Golomb 编码参数 k。上述参数中 A[Q]在 8bit 图像压缩时初始值一般为 4，16bit 图像压缩时初始值一般为 1024，N[Q]的初始值为 1，其他参数初始值均为 0。完成一次编码后，就需要在编码完成后对上下文参数进行更新。在更新后，若下轮压缩 Q 值与本轮相同，更新后的上下文参数将立即作用于压缩过程。

根据各个上下文参数所代表的含义，它们的更新方式如下所述。A[Q]的更新方式是直接与量化后预测误差 Errval 的绝对值进行相加操作，得到新的 A[Q]值。B[Q]则是与重建后的误差值，即量化后预测误差 Errval 与(2*near+1)相乘后与 B[Q]相加，得到新的 B[Q]值。此时若 N[Q]的值达到 RESET 值，所有上下文参数均右移一位，对之前的上下文信息进行压缩，一般情况下 RESET 值为 64。最后，将 N[Q]值加 1。

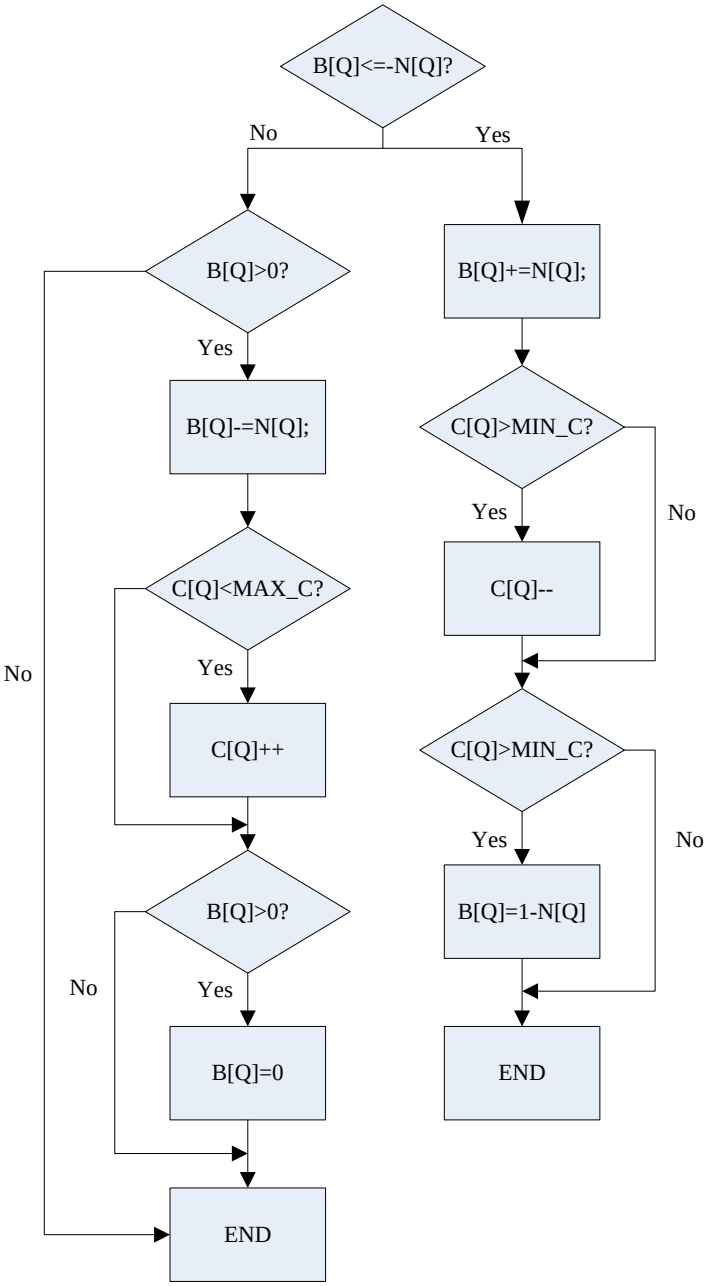

图 2.3　C[Q]更新流程图

更新完上述上下文参数后，再更新 C[Q]，同时伴随着对 B[Q]的一些修正，更新流程如图 2.3 所示。C[Q]的更新方式分为两种情况，当 B[Q]小于等于 N[Q]的相反数时，需要将 B[Q]的值与 N[Q]的值相加，得到新的 B[Q]值，新的 B[Q]值若依然小于等于 N[Q]的相反数，则直接令 B[Q]等于 1-N[Q]。此时若 C[Q]的值大于 MIN_C，则将 C[Q]值减 1。

而当 B[Q]大于 0 时，需要将 B[Q]的值减去 N[Q]的值，得到新的 B[Q]值，新的 B[Q]值若依然大于 0，则直接令 B[Q]等于 0。此时若 C[Q]的值小于

MAX_C，则将 C[Q]值加 1。参数 MIN_C 和 MAX_C 分别是对 C[Q]最小值和最大值的限制，一般取-127 和 128：

2.2.8 预测误差的编码

在对预测误差编码前，首先要将预测误差 Errval 映射为非负整数 MErrval，之后再对映射后的误差值 MErrval 进行限长 Golomb 编码。限长 Golomb 编码所需的编码参数 k 由上下文参数 A[Q]和 N[Q]计算 Golomb，k 的取值为能使 N[Q]左移 m 位大于等于 A[Q]成立的最小 m 值。得到编码参数 k 后便可以对 MErrval 进行限长 Golomb 编码，k 值是令 Golomb 编码码长最优的参数值。Golomb 编码的编码流程如图 2.4 所示。

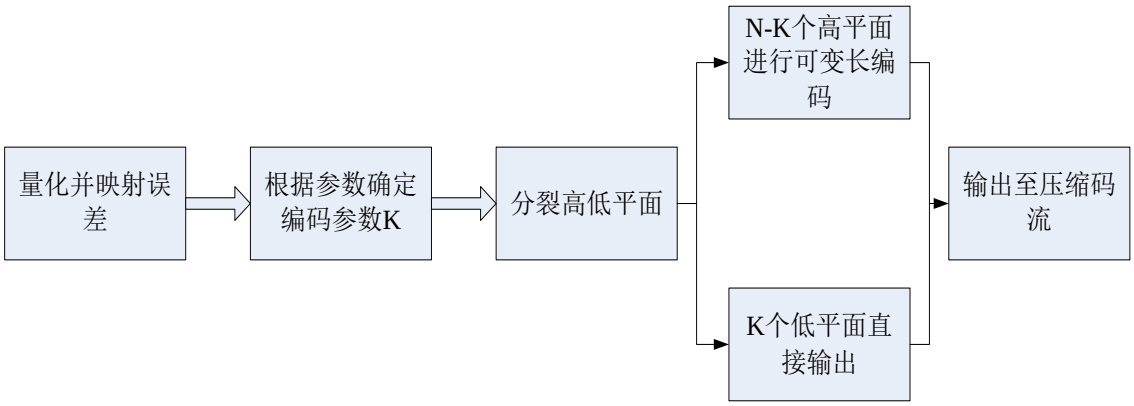

图 2.4　Golomb 编码流程

在对整数 n 进行 Golomb 编码时，首先将 n 除以参数 k 得到商和余数，商的部分采用一元码编码，之后余数的部分直接以二进制方式附加到码流中。

而在限长 Golomb 编码中，码字的最大长度被限制为 limit。若使用限长 Golomb 编码对最大位长为 qbpp 的整数 n 进行编码，则存在两种情况下的编码方式，此时 k 的取值一般不大于 qbpp。当 n < limit - qbpp – 1 时，编码方式与普通的 Golomb 编码方式无异；而当 n >= limit - qbpp – 1 时，首先对整数 limit-qbpp-1 进行一元码编码，一元码的码字长度为被编码整数的值加 1，所以此时的码长为 limit-qbpp，之后则将整数 n-1 用 qbpp 位二进制数进行表示并附加至码流中，这样编码后的码字长度恰好达到 limit。

在 JPEG-LS 算法标准中，对预测误差的校正、模减以及映射操作使其普遍

集中在 0 值附近，易于通过限长 Golomb 编码方式进行编码，并取得较短的码长。而限长 Golomb 编码也保证了个别较大的误差值不会使码字长度剧增，降低编码器的性能。

2.3 游长模式下的编码

在一幅图像中，或多或少存在着连续的像素值相同的区域，这些区域一般被称为“平坦”区域。平坦区域的出现意味着图像中存在着可供压缩的冗余信息，JPEG-LS 算法标准采用游长编码的方式对平坦区域进行编码，大幅度降低了压缩这类区域所使用的码字长度。当编码器进行无损压缩时，如果待编码像素点已出现的上下文具有相同的像素值时，会开始进行游长模式的编码[20]。此时会记录当前采样位置的像素值为参考值 a 并开始扫描，直至扫描到像素值不等于参考值 a 或行尾为止，退出游长模式并开始编码。前一种情况下，需要首先对游长模式扫描过的像素点数量，可以称之为游长长度，进行编码，而后对中断点像素值进行限长 Golomb 编码。后一种情况下，完成对游长长度的编码后，便可以开始对下一行像素的编码。

对游长长度编码时，一般先给定一组游长长度等级 rm，rm 一般为 2 的 rk 次幂。一个典型的具有 32 个 rk 值的表 J 如图 2.5 所示。

0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	5	5	6	6	7	7	8	9	10	11	12	13	14	15

图 2.5　一个典型的具有 32 个 rk 值的表 J

当游长长度大于等于长度等级 rm 或该游长段因扫描至行末尾而中断时，一个一比特的码字“1”会被加入压缩码流中被用于对该游长段进行编码。若当前游长段因遇到扫描行末尾而被终止命中且剩余游长长度大于 0 的情况下,将一个一比特的码字“1”加入压缩码流中；若不是，则将游长长度减去 rm，并开始判断下一个是否命中下一个长度等级 rm。而当剩余游长长度小于当前长度等级 rm 时，则先将一个一比特的码字“0”加入压缩码流，后用 rk 个比特位对剩余游长的实际长度进行编码。

在游长扫描遇到奇异像素点中断的情况下，程序将在对游长长度编码完成后再对导致中断的像素点的值进行 Golomb 编码。JPEG-LS 算法将中断的情况分为

两种，一种是中断点左侧的像素点 x_a 和上方的像素点 x_b 像素值相等，另一种则是这两个像素值不等。前一种情况下对待编码像素点的预测值为 x_a，而后一种情况的预测值为 x_b。之后再经过误差计算及映射、限长 Golomb 编码后完成编码流程。

2.4 解码流程

对经 JPEG-LS 算法压缩产生的码流进行解码的过程与 JPEG-LS 编码过程类似。首先，对待解码像素点的上下文进行与编码过程相同的梯度计算，以判断待解码像素点采用了何种编码方式进行编码。若待解码像素点是在普通模式下进行编码的，则根据待解码像素点的上下文进行预测得到预测值，并利用对应上下文模式的参数对预测值进行修正。通过参数还可以计算出 Golomb 编码的参数值 k，根据 k 值在码流截取相应的编码长度进行 Golomb 编码的解码得到误差值，由于该误差值是实际误差值经过映射后得到的，所以需要根据原误差值通过映射关系得到原误差值，将预测值与误差值相加即可得到原像素值。同时，对应上下文模式的参数会根据解码所得到误差值进行更新。若判断编码模式为游长模式，记录上一次解码得到的像素值为参考值，再根据接下来的码字判断游长模式中断的原因。当由于遇到行末尾而中断时，将目前待解码位置至行末尾的像素点全部填充为参考值；当由于遇到中断点导致游长模式中断时，则从码流中解码得出游长长度，将从待解码位置之后数量为游长长度的像素点均填充为参考值，再从码流中解码得到中断点的像素值，最后继续下一个像素点的解码。

2.5 图像质量的评价指标

2.5.1 码率

在图像压缩领域，图像压缩后的码率是指单位像素编码所需要的编码长度，一般单位为 bpp(Bit Per Pixel)。进行无损压缩时，若在压缩同一位深度的图像时取得的码率越低，则代表压缩性能越好。

2.5.2 峰值信噪比（PSNR）

峰值信噪比（PSNR）是评价图像质量的一种较为常见的指标。若要得到一幅重建图像的峰值信噪比的值，需要计算它与原图像之间的均方误差，并作处理后

得到[21]。具体如公式(2.8)所示：

$$\mathrm{MSE} = \frac{1}{H * W}\sum_{i=1}^{H}\sum_{j=1}^{W}(X(i,j) - Y(i,j))^2$$

$$\mathrm{PSNR} = 10\log_{10}(\frac{(2^n-1)^2}{MSE}) \tag{2.8}$$

峰值信噪比是评价重建图像质量时比较重要的指标，不过这一指标也存在着诸多局限。一些研究表明，峰值信噪比的数值可能与人眼对于图像的评价有一定的偏差[22]。这是因为人眼对图像某个区域的感知结果不单单取决于当前区域，在一定程度上取决于该区域的临近区域对人眼的影响。所以不能单单使用 PSNR 来评判图像质量的优劣，需要引入新的图像质量评价概念。

2.5.3 结构相似性（SSIM）

结构相似性是一种衡量两幅图像相似度的指标，该指标首先由德州大学奥斯丁分校的图像和视频工程实验室提出，给定两个图像 x 和 y, 则这两张图像的结构相似性可按照公式(2.9)的方式求出[23]：

$$\mathrm{SSIM(x,y)} = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \tag{2.9}$$

其中μ_x是 x 的平均值，μ_y是 y 的平均值，σ_x^2是 x 的方差，σ_y^2是 y 的方差，σ_{xy}是 x 和 y 的标准差。$\mathrm{c}_1 = (k_1L)^2$，$\mathrm{c}_2 = (k_2L)^2$是用来维持稳定的常数。L 是图像像素值的动态范围。$k_1 = 0.01, k_2 = 0.03$。结构相似性的范围为-1 到 1。当两张图像一模一样时，SSIM 的值等于 1。

2.6 本章小结

本章深入分析了 JPEG-LS 图像压缩标准，编码时主要分为普通编码模式和游长编码模式。其中普通模式基于上下文模型的预测和 Golomb 编码，而当图像出现大片“平坦”区域时则使用游长模式进行编码，具体采取何种方式进行编码取决于待编码像素点所处的上下文模式。JPEG-LS 图像压缩标准的解码流程也被给出，有助于更好地理解 JPEG-LS 图像压缩标准。同时介绍了一些图像质量的评价指标，为下文对码率控制算法的对比做基础。

第3章 码率控制算法的研究与改进

JPEG-LS 算法首先从图像源读出数据，对当前像素点已经出现的临近点进行梯度计算，无损模式下如上下文建模中上下文的梯度值均为 0，则进入游长模式编码，否则当前梯度映射到某一种上下文模式中，并对利用边界探测器得出的当前像素预测值进行自适应校正，再将像素真实值与校正后的值做差，最后对该差值进行 Golomb 编码。而编码器产生的预测值与实际值之间的误差并不能保持在一定的范围内，会根据图像的特性出现一定的波动。由于 Golomb 编码的特性，同一编码参数下待编码数值出现波动时，码字长度就会随之变化，就会产生码率不可控的问题。

3.1 现有码率控制算法分析

关于 JPEG-LS 的码率控制算法，国内外一直有相关的研究在进行，比较广泛采用的是基于预处理的码率控制方法以及基于调整误差量化参数 near 值的码率控制方法。基于预处理的码率控制方法中，比较典型的有基于对子图进行预处理的防码流溢出方法，该方法通过对预处理变换参数大于 1 的子图进行变换处理，以达到控制码流溢出的目的[24]。由于需要进行预处理以及变换操作，该方法编码器结构较为复杂，不利于硬件实现。而基于对 near 值调整的码率控制方法，一般通过反馈调节的方式使码率达到设定值[25-28]。在现有的码率控制算法中，比较有代表性的有一阶线性码率控制算法[29]和基于先验数据表的码率控制算法[30]，它们都在 near 值的调整与码率变化之间建立了简明有效的数学关系，编码器结构较为简单，利于硬件实现。

一阶线性码率控制算法认为 near 值较小时可视作与码率呈线性关系，该算法编码时将图像划分为 N 个 block，每压缩一个 block 就观察当前码率是否超过较高的码率阈值或小于较低的码率阈值，如果成立则增加或减少 near 的值，否则不变，继续压缩下一个 block，通过这种操作不断逼近目标码率。而基于先验数据表的码率控制算法则认为，在 Golomb-Rice 编码中，当失真量化参数发生变化时，像素的平均码率的变化量只与这调整前后的两个失真量化参数有关，和具体

的图像复杂度及其它因素基本无关，从而可以建立一个先验的存有相邻 near 值码率变化量的表，压缩过程中也不需要更新，只需要按照一阶线性码率控制算法中的动态调整方法，在压缩完一个 block 后根据码率调整表进行 near 值的增减。该方法码率收敛速度和码率精确度均优于一阶线性码率控制算法。

基于先验数据表的动态码率调整算法的理论基础是在 Golomb-Rice 编码中，当 near 值发生变化时，压缩码率的变化只与变化前后的两个 near 值有关，这是由大数据量下 Golomb-Rice 编码的特性决定的。在只考虑 Golomb-Rice 编码的情形下，这种分析是稳定而可靠的。然而 JPEG-LS 编码中还存在着游长编码部分，游长编码是针对图像“平坦”区域，即像素值变化量不超过容限值的区域，进行编码的方式。如若图像平坦区域较多，则该先验数据表所显示的压缩码率变化率是不准确的。下面我们将对该理论进行验证。

下面我们将对图 3.1 和图 3.2 的图像（尺寸为 512*512，位宽 8bit）在 near 值为 0、1、2 的情况下分别进行压缩，压缩后对其压缩码率、进入游长编码的次数分别进行统计，同时计算 near 值变化后压缩码率的差值。结果如表 3.1 和表 3.2 所示。

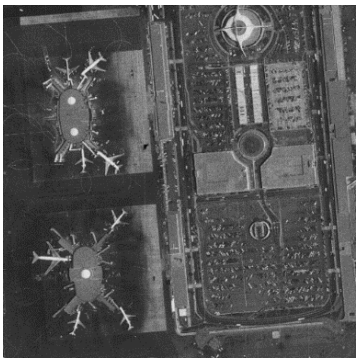

图 3.1　测试图像 lax

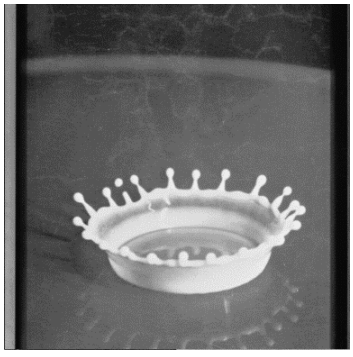

图 3.2　测试图像 milkdrop

表 3.1 测试图像 lax 在不同 near 值下的压缩指标

near	0	1	2
码率（bpp）	5.742	4.311	3.739
游长编码个数	0	86	476
码率差（bpp）	--	1.431	0.582

表 3.2 测试图像 milkdrop 在不同 near 值下的压缩指标

near	0	1	2
码率（bpp）	3.625	2.671	2.404
游长编码个数	2273	9192	14004
码率差（bpp）	--	0.954	0.267

从表 3.1 和表 3.2 的数据对比可以看出，如果一个图像在压缩过程中游长编码使用的次数越多，则同 near 值下压缩出来的码率就会比游长编码使用少的图像码率低。这符合我们的直观感受，因为游长编码使用的越多，说明图像“平坦区域”越多，也就更容易压缩。同时也可以看到，游长编码使用少的图像，near 值发生变化时码率的变化幅度也更大，反之则 near 值变化更小。所以在用先验码率调整表进行动态码率控制时会产生一些误差，使得调整幅度过大或过小，造成重构图像的不平滑。

若要对该方法进行改进，首先需要对先验数据表进行动态更新。这就意味着在每完成一个 block 的压缩后，不仅要对 near 值进行更新，也要计算该 block 压缩所用的 near 值相比上一个 block 所用的 near 值之间的码率差，用来对码率变化表进行修正。具体修正方法将由 3.2 节给出。

其次，通过表 3.1 和表 3.2 可以观察得到，near 值从 0 变化到 1 的时候码率变化幅度较之其他情况要大得多。为了使码率变化更为平滑，可在中间设置 near 值为 0.5 的调整节点。JPEG-LS 中误差量化如公式(3.1)所示。

$$QErrval = \frac{Errval+near}{2*near+1} \tag{3.1}$$

其中 QErrval 代表量化后的误差，Errval 代表原误差。当 near 取 0.5 时除数为 2，不会产生除数是非整数的问题。

3.2 基于动态码率调整表的码率控制算法

本文提出的新的码率调整算法是基于动态的码率调整表之上的，该方法将图像每 n 行划分为一个 block，每压缩完一个 block，根据码率调整表来调整 near 值压缩下一个 block，同时更新码率调整表。具体步骤如下：

设置每一个 block 的行数为 m，每行包含像素点数目为 p，目标码率为 b，near 初始值为 0.5，并根据经验得到初始码率调整表如表 3.3 所示。该表所示本算法的 near 值上限为 15。

表 3.3　初始码率调整表

失真量化参数	码率差（bpp）
0	--
0.5	1.0
1	0.5
2	0.5
3	0.3
4	0.3
5	0.2
6	0.2
7	0.1
8	0.1
9	0.1
10	0.1
11	0.05
12	0.05
13	0.05
14	0.05
15	0.05

每当压缩至第 n（n>1）个 block，计算当前 block 的码率为b_1，图像当前总压缩码率为b_{sum}，记录当前 near 值为$near_1$，第 n-1 个 block 的 near 值为$near_2$，上一个 block 的码率为b_2。若$near_1$与$near_2$不相等，则更新码率调整表。

更新码率调整表的步骤为：假设$near_2$>$near_1$，首先计算上一 block 的码率b_2与当前 block 的码率b_1的差值，由于 near 值调整带来的码率调整幅度不是均匀变化的，可进行加权计算得出$near_1$至$near_2$范围内相邻 near 值变化导致的码率差。根据式(3.2)计算如何根据第 n-1 个 blcok 到第 n 个 block 的码率变化情况，对 near 值为 i（$near_1 \le i < near_2$）时的码率调整表进行调整。其中 near_dis 代表由当前码率调整表得出的$near_2$变化至$near_1$时的码率差之和，Δbpp_{neari}代表当前码率调

整表下 near 值为 i 时由 i 变化为下一个 near 值时的码率差。

$$\Delta bpp'_{neari} = \frac{(\frac{b_2-b_1}{|near\_dis|}*\Delta bpp_{neari}+\Delta bpp_{neari})}{2} \tag{3.2}$$

压缩完第 n+1 个 block 后目标码字数为$(n+1)*m*p*b$，当前即已经压缩完的 n 个 block 的码字数则为$n*m*p*b_{sum}$，设第 n+1 个 block 的目标码率为b_{next}，根据式(3.3)计算下一个 block 所需的码率b_{next}。

$$b_{next} = \frac{(n+1)*m*p*b-n*m*p*bsum}{m*p} \tag{3.3}$$

化简后可得式(3.4)。

$$b_{next} = (n+1)*b-n*b_{sum} \tag{3.4}$$

再计算b_{next}与当前 block 的码率b_1的差值，根据当前 near 值$near_1$在码率调整表中的位置，找到能够使码率差最符合要求的新 near 值来满足条件，使新 near 值作用于下一个 block 的压缩过程中。

3.3 算法性能评估及结果分析

算法的性能评价标准采用码率收敛步数以及压缩后恢复图像的峰值信噪比（PSNR）和结构相似性（SSIM）等三个数值。我们认为当图像压缩完第 n 个 block 后压缩率达到目标压缩率±0.05 范围内，且直至压缩结束时压缩率不会超出该范围，则该图像压缩过程需 n 步完成收敛。下面将对图 3.3 所示分辨率为 512*512 的 8bit 太阳图像同时采用两种码率控制算法进行压缩，并对比两种算法在上述标准下的优劣。

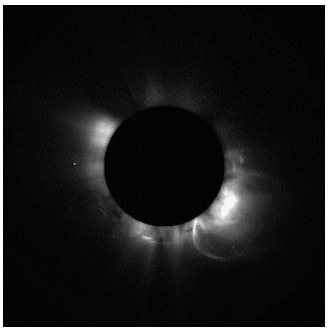

图 3.3　测试图像 sun(8bit)

取每一个 block 的行数为 16，目标码率为 2bpp，即目标压缩率为 4，可得出两种算法的码率调整曲线如图 3.4 所示。

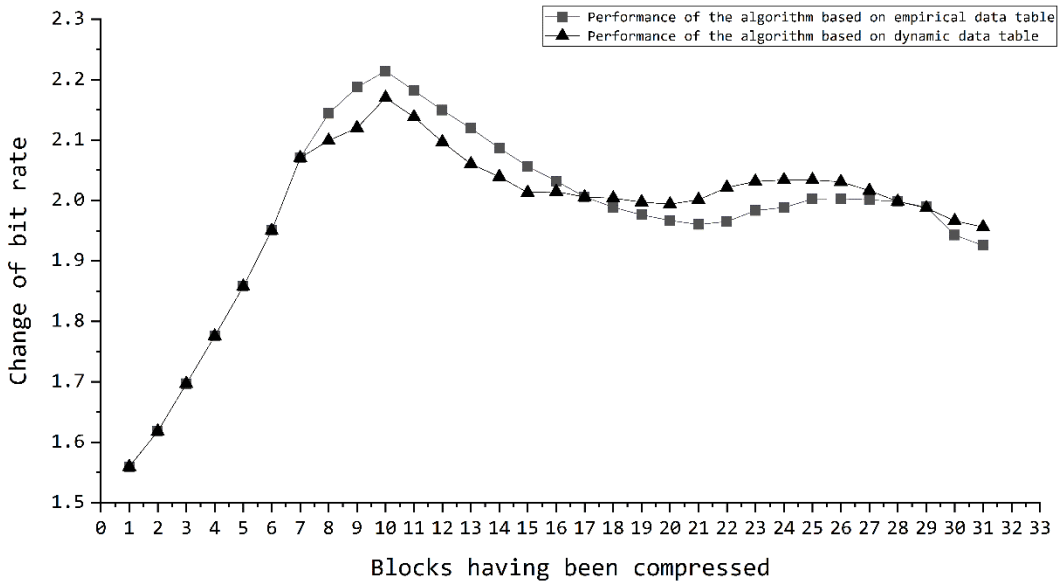

图 3.4　两种算法的码率调整曲线

由图 3.4 我们可以看出，两种算法刚开始对码率的调整效果相同，而当接近目标码率时，基于动态数据表的码率控制算法可以更快地收敛，且码率波动更小，最终结果更接近目标码率。同时基于动态数据表的码率控制算法所得到的重建图像，PSNR（峰值信噪比）值为 34.8821，SSIM（结构相似度）值为 0.8405；而基于先验数据表的动态码率控制算法得到的重建图像 PSNR 值为 33.8258，SSIM 值为 0.7727。

再选择两组 16bit 太阳局部图片，一组分辨率为 512*512，另一组分辨率为 1024*1024，取每一个 block 的行数为 16，目标码率为 5.33bpp 即目标压缩率为 3，进行算法性能测试测试。所得到的收敛步数如图 3.5、图 3.6 所示。

从这两幅图表的对比我们可以看出，所有被压缩图像在基于动态数据表的算法下达到码率收敛所需的收敛步数明显小于基于先验数据表的算法。

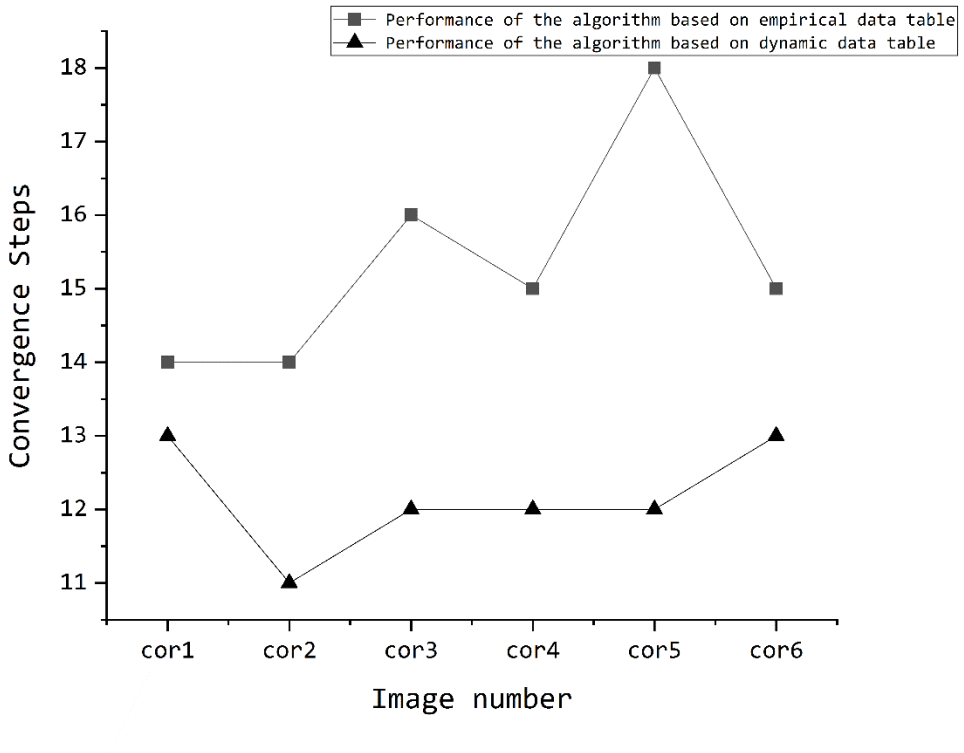

图 3.5　图像系列 cor（分辨率 512*512）收敛步数对比

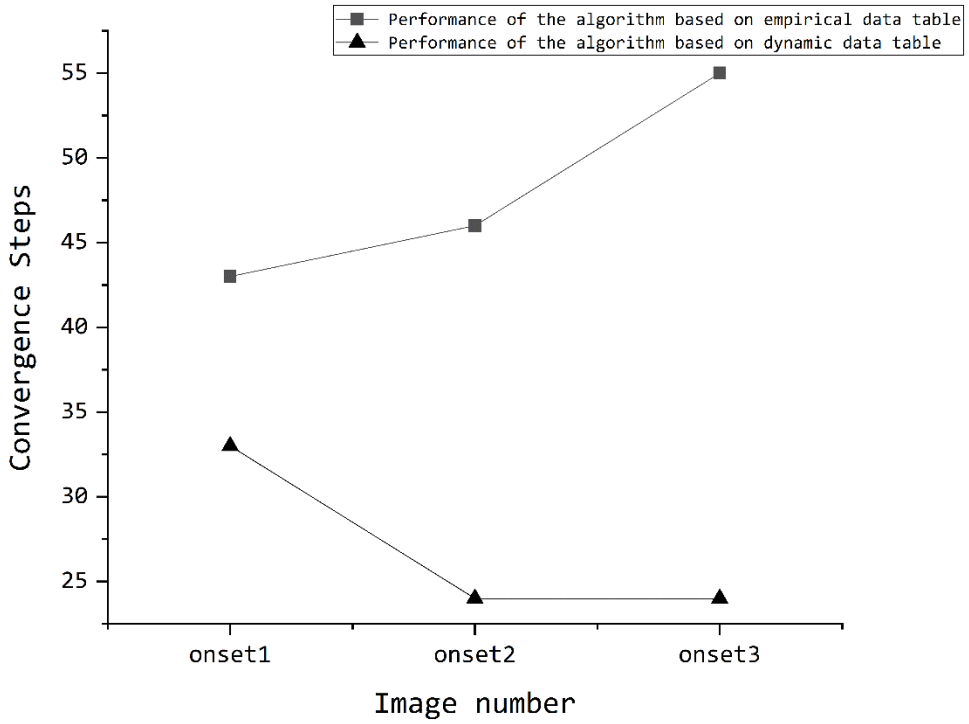

图 3.6　图像系列 onset（分辨率 1024*1024）收敛步数对比

两组图像经两种算法压缩后解码所得重建图像的 PSNR 值以及 SSIM 值对比如表 3.4 和表 3.5 所示。可以看出基于动态数据表算法得到的重建图像质量明显由于基于先验数据表的算法所得的重建图像，且图像尺寸越大，差距越明显。这是由于在同样的目标码率下，两种算法都是先增大 near 的值使压缩码率快速接近目标码率，再对 near 值进行微调让码率收敛至目标码率。基于动态数据表的算法能更快更准确地通过调整 near 值使压缩码率收敛至目标码率，而近无损压缩模式下，像素 x 经过误差量化后重建得到的像素值都会处于[x-near,x+near]的范围内，near 值调整得更快更准确，有利于减小重建图像与原图像的均方误差，从而使得图像质量更优。

表 3.4　图像系列 cor（分辨率 512*512）和 onset（分辨率 1024*1024）重建图像 PSNR 值对比

图像名称	动态数据表算法	先验数据表算法
cor1	40.1527	39.9352
cor2	46.7350	46.2295
cor3	43.9284	43.5900
cor4	47.2992	46.6795
cor5	44.8926	44.3121
cor6	44.9488	44.4505
onset1	45.0393	43.3560
onset2	45.3974	43.5766
onset3	45.3304	43.5421

表 3.5　图像系列 cor（分辨率 512*512）和 onset（分辨率 1024*1024）重建图像 SSIM 值对比

图像名称	动态数据表算法	先验数据表算法
cor1	0.9200	0.9067
cor2	0.9675	0.9570
cor3	0.9538	0.9422
cor4	0.9702	0.9594
cor5	0.9581	0.9455
cor6	0.9588	0.9470
onset1	0.9713	0.9552
onset2	0.9726	0.9564
onset3	0.9728	0.9567

3.4 本章小结

本章分析了 JPEG-LS 算法与 near 值的关系，结合之前关于动态码率控制方法的研究，找出其中的不足之处，改进并提出了一种新的码率控制方法。实验结果表明：该算法在码率控制精度以及码率收敛速度等主要性能指标上均优于现有的码率控制算法。

第4章 JPEG-LS 近无损压缩编码器的硬件实现

4.1 星载图像压缩硬件平台简介

星载硬件平台规模较小，功耗低，专用性、可靠性强，图像压缩系统作为星载数据处理系统的子系统，一般也采用专用硬件平台实现。目前广泛采用的图像压缩系统硬件平台有：图像压缩 ASIC（Application Specific Integrated Circuit），DSP（Digital Signal Processor），以及 FPGA（Field Programmable Gate Array）。

4.1.1 ASIC

ASIC 是基于定制方式的专用集成电路，一般根据用户特定的需求进行设计和制造。ASIC 电路一般分为数字、模拟和数模混合三种类型，其中数字 ASIC 电路有全定制和半定制两种实现方法。全定制方式需要设计者在半导体管级，使用设计工具完成全部电路设计，因此开发效率较为低下。半定制方式则允许设计者使用标准逻辑单元库中的逻辑单元，可以使设计更可靠，方便设计者完成系统设计。

ASIC 在批量生产时可以提高可靠性以及降低成本，但在小规模应用情境下，开发周期过长，成本也偏高，且由于 ASIC 电路的不可修改性，会带来比较高的风险。

4.1.2 DSP

DSP 即数字信号处理器，内部一般都集成多个处理单元，如硬件乘法器(MUL)、累加器(ACC)以及其他算术单元等，可以快速实现各种数字信号处理算法，目前已广泛应用到控制、通信等多个领域[31]。

DSP 的缺点是缺乏并行处理机制，对于图像压缩此类需要并行处理的情境，DSP 若想取得较高的处理速率只能通过提高输入时钟频率的方法，然而这样又会带来功耗过大的问题，这使得 DSP 的应用受到限制。

4.1.3 FPGA

FPGA 是一种可编程器件，设计者可以根据需求通过设计方法在 FPGA 上实现各种功能[32]。在对 FPGA 的开发中可以以硬件描述语言（Verilog 或 VHDL）完成电路设计，在简单的综合和布局布线后，即可烧录在在 FPGA 器件上完成测

试。

FPGA 开发周期短，成本低，具有更好的可靠性和可维护性，同时可以进行并行运算，不需要较高的时钟频率即可取得较好的数据处理速率。所以在星载图像压缩领域，FPGA 与 ASIC 和 DSP 相比更有竞争力。

4.2 FPGA 设计开发技术

完整的 FPGA 设计流程包括设计、功能仿真、综合、布局布线、时序仿真以及加载配置和板级验证与仿真等。

4.2.1 设计输入

设计输入即将设计者要实现的电路功能输入到 EDA 软件中，一般通过硬件描述语言(HDL)或原理图设计的方式进行。虽然原理图设计的方式更加直观，但随着电路规模的扩大，这种方式也变得越来越难以维护，而硬件描述语言可以很方便地进行模块复用和设计的升级维护，所以被广泛采用。比较流行的硬件描述语言有 Verilog HDL 和 VHDL。

4.2.2 功能仿真

在初步完成电路设计后，需要对设计进行功能仿真，以验证设计是否能实现预期功能。这一步主要是通过编写仿真文件 Testbench 生成种类足够丰富的激励，输入到设计中并检查结果是否与预期一致。

功能仿真能及时发现并定位设计中存在的逻辑错误，方便进行自下而上的设计。

4.2.3 综合优化

综合过程是将设计者输入的设计编译成由门电路、RAM 存储单元、触发器等基本逻辑单元组成的电路。在综合过程中存在着一些约束条件，可以使综合过程按照设计者的意愿对综合后生成的电路进行优化，最常用的约束条件有 I/O 管脚位置约束和电平幅度约束。

综合优化后 EDA 软件会输出网表（Netlist）文件，网表文件中存储了基本逻辑单元之间的连接关系，这一文件可供 EDA 软件进行实现。

4.2.4 实现与布局布线

实现是在生成网表文件后，EDA 软件会根据设计者所选用的 FPGA 器件型号，将网表文件对该 FPGA 芯片进行适配。布局将网表文件中的硬件原语和底层单元合理地配置到芯片内部的固有硬件结构上。

Xilinx ISE 开发套件对实现过程分为以下三个步骤进行：

1. 转换（Translate）：将综合过程中生成的网表文件翻译成硬件原语和底层单元。

2. 映射（Map）：将网表文件与具体型号的 FPGA 芯片进行适配。

3. 布局布线（Place＆Route）：通过布局布线器，设计将在 FPGA 芯片上被布局并根据网表文件中逻辑单元之间的连接关系合理正确地进行布线，整个过程会遵循用户以及 EDA 软件施加的各类约束。

布局布线过程完成后，在时序约束的限定下，需要对设计进行检查以判断设计是否满足设计者对其的时序要求，这一过程叫做静态时序分析。静态时序分析不需要输入激励信息，即可生成一个包含所有时序信息的报告，如关键路径、扇入扇出等。通过分析这一报告，能够发现设计中存在的毛刺、延迟路径和时钟偏移等问题。

静态时序分析运行速度快，且可以对设计进行全面的时序检查，在检查过程中能够发现最大的延时路径并对其进行优化，现已被广泛应用于数字集成电路验证之中。

4.2.5 时序仿真

在设计布局布线完成以后可以提供一个时序仿真模型，可以利用该模型对设计做时序仿真。

EDA 软件会在布局布线后给出一个包含时序信息的设计模型，时序仿真可以利用与功能仿真相似的 testbench，利用该模型对设计中存在的时序问题进行分析，而时序仿真与功能仿真的不同在于前者会将最长的布局布线延时到仿真器的设计，并且显示在仿真结果波形图中。

4.2.6 板级仿真与验证

在将设计实现并布局布线后，一个设计信息的二进制文件会被生成，可把该文件下载至 FPGA 中，进行对设计进行板级仿真与验证，Xilinx 器件一般通过 iMPACT 工具实现。在板级验证阶段，可以采用 Chipscope 或示波器的方式进行

调试。Chipscope 可以将 FPGA 内部的信号和数据总线接入并传到上位机进行显示，示波器则可以接入开发板的管脚，直接将管教点评数据读出并显示。

4.3 编码器概述及难点分析

本文基于 JPEG-LS 算法标准，结合 FPGA 的特性和设计原则，设计了一种 JPEG-LS 近无损压缩编码器。这个编码器旨在通过流水线结构来高效率的实现 JPEG-LS 近无损压缩方法，并可以控制码率。编码器结构框图如图 4.1 所示。

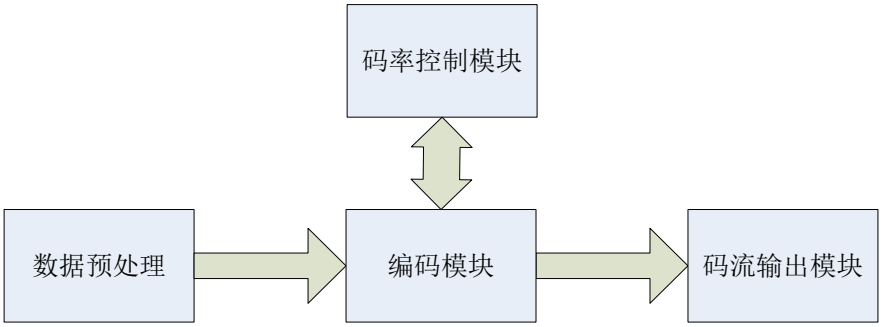

图 4.1　JPEG-LS 图像压缩编码器框图

在整个编码器中，外部图像数据经过数据接口模块处理后输入至编码模块，最后经由码流输出模块进行整合后输出。整个 JPEG-LS 图像压缩系统的核心即编码模块。

编码模块的工作流程如图 4.2 所示：上下文以及待编码像素点的像素值数据首先经过梯度及 Q 值计算模块的处理，计算梯度后判断是否进入游长编码模块。若进入游长编码模块，编码器进入游长扫描模式，直至中断后经过游长编码后将压缩码流数据直接输入码流输出模块。若进入普通模式编码，则对梯度值进行量化并预测 Q 值，再经过边缘检测模块得出预测值，在误差预测及量化模块与原像素值、C[Q]值分别进行除法运算并融合以得出误差值，然后误差值映射后进行 Golomb 编码，期间进行参数更新，最后 Golomb 编码的结果输出至码流输出模块。像素重建值于梯度及 Q 值计算模块进行预测，在误差映射及参数更新模块进行选择正确的像素重建值并输出，具体机制在下文给出。编码过程中所用到的上下文数据从图像重建值缓存模块读取，经过编码后重建的像素数据也会返回图像缓存模块进行存储。

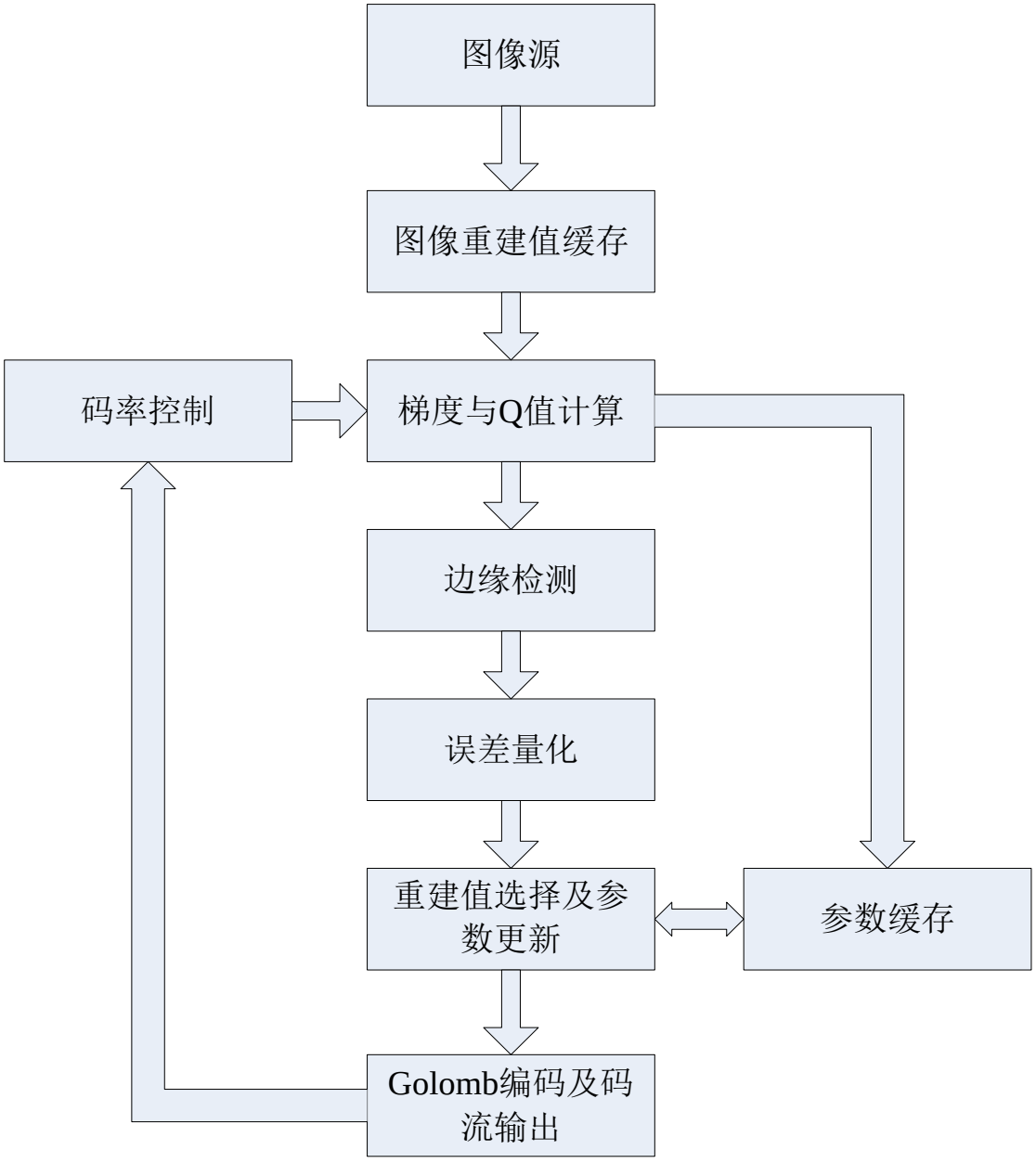

图 4.2　编码模块工作流程图

4.3.1 重建值的预测

在 JPEG-LS 图像压缩算法中，梯度和 Q 值的计算以及边缘检测依赖于待编码像素已出现的上下文。而在近无损压缩模式中，Q 值的计算需要用到位于待编码像素左侧的像素重建值 Ra，该值需要通过上一次压缩量化后的误差重建得到，这使得像素重建值无法及时反馈到梯度和 Q 值计算以及边缘检测模块。为了实现高效的流水线处理，本系统采用对像素重建值进行预测，并在编码前对像素重

建值进行选择的机制，此机制不需要等待上一次压缩结束后再进行本次压缩，从而大大地提高了压缩的效率。

本系统中，由于采用的是近无损压缩模式，梯度计算后量化为索引值 Q 时采用的量化方式如公式(4.1)所示，D_i代表梯度，Q_i代表梯度量化后对应的索引值。可以发现当前量化公式与无损模式下量化公式的区别在于，近无损模式下当梯度值小于等于 near 时即可被量化为 0，而无损模式下仅当梯度值为 0 时量化值才为 0。

$$Q_i = sign(D_i) * \begin{cases} 0, |D_i| \le near \\ 1, near < |D_i| < T_1 \\ 2, T_1 \le |D_i| < T_2 \\ 3, T_2 \le |D_i| < T_3 \\ 4, |D_i| \ge T_3 \end{cases} \tag{4.1}$$

本系统对参数T_1，T_2，T_3的设置如式(4.2)所示，其中 bpp 代表图像像素值的位宽，一般为大于等于 8 小于等于 16 的整数。

$$\begin{cases} T_1 = (bpp - 7) * 3 + near \\ T_2 = (bpp - 7) * 7 + 2 * near \\ T_3 = (bpp - 7) * 21 + 3 * near \end{cases} \tag{4.2}$$

Q 值的计算公式如式(4.3)所示。

$$\mathrm{Q} = 81 * q_1 + 9 * q_2 + q_3 \tag{4.3}$$

q_1 与 q_2 的值由前行像素重建值产生，在当前对当前像素点进行压缩前均已得到。而 q_3 的取值涉及到待编码像素点左侧的像素重建值 Ra，该值在本轮压缩开始时并不会得到，所以 Q 值预测只涉及到 q_3 的取值。由于近无损压缩模式下，像素点 x 经过误差量化后重建得到的像素值都会处于[x-near,x+near]的范围内。观察公式(4.1)与(4.2)可知，Q 值量化之中每两个阈值之间的间隔都大于 near，即使像素点 x 的重建值与原像素值之间的误差达到最大，量化至 q_3 时至多仅产生不大于 1 的误差。所以在梯度量化后只会产生三个可能的量化后梯度值 q_3，q_3-1 和 q_3+1，其中 q_3 代表由点 a 的原像素值进行梯度计算并量化后得到的索引值。

经过以上操作后最终产生三个索引值 Q_1，Q_2 和 Q_3，需要设置三个参数缓存模块，每个模块中含有一组 4 个 RAM 存储四种上下文参数值。在一次压缩中，产生上述三个索引值后分别在三个参数缓存模块中取出上下文参数，在误差量化后，计算得到重建值选择出正确的参数更新结果，并将更新后的参数值同时写入三组 RAM 中，像素重建值也会写入图像重建值缓存模块中。

4.3.2 误差量化中的除法运算

近无损压缩模式下的误差量化公式如式(2.6)所示，这意味着每一次压缩都需要进行除法运算。文献[33]中采用了一种基于查表的除法运算，将所有可能的除法运算结果分为商和余数都保存在片内 ROM 中。将与误差相关的预测值、原像素值以及 C[Q]值分别进行该种除法运算再进行商和余数的合成，便可得到最终误差量化的结果。

由于文献[33]中输入图像最大为 10bit，near 值上限为 10，所以最大只需储存 1024*11 个 10bit 运算结果。本文所述系统输入图像上限为 16bit，若采用此方法则需储存 65536*11 个 16bit 运算结果，耗费资源巨大。本文在此基础上进行改进，将 16bit 除法的分为高 8bit 和低 8bit 进行运算，高 8bit 数和低 8bit 数的计算结果分别缓存与 RAM 中。先取 16bit 数中的高 8bit 进行运算后产生商和余数，余数与低 8bit 相加后在进行除法运算，产生的商与高 8bit 所产生的商相加为最终的商，产生的余数为最终的余数。此方法比原方法多耗时一个时钟周期，并需要少量额外的逻辑对两步除法结果进行融合,但它的优点就在于只需要在 RAM 中储存 256*11+(256+20)*11=5852 个运算结果，大幅优化了原有算法。

4.3.3 码率表的动态调整

编码模块的最后将设置一个码率控制模块连接码流输出模块和编码模块的梯度计算及量化模块之间，动态码率表缓存于该模块中。编码模块每压缩完一个 block 便会将该 block 压缩后的码流长度反馈至码率控制模块，该模块将首先根据码流长度计算得到该 block 的码率；然后码率控制模块便会依据 3.2 节所述方法更新码率调整表，其中码率表的调整所用除法也采用查表的方法进行，除法精确到小数点后两位。最后根据新的码率调整表来修改梯度计算及量化模块处的 near 值，由于码率的计算以及码率表的调整需要多个时钟周期，所以调整后的 near 值将作用于计算完成时的下一个 block 压缩。码率调整的算法流程如图 4.3 所示。

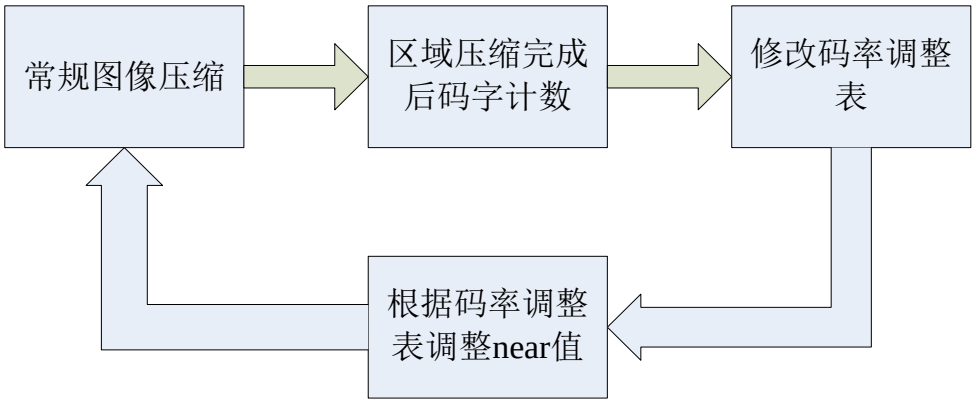

图 4.3　码率调整算法流程图

4.4 最终实现方案硬件结构

4.4.1 图像重建值缓存模块

预处理模块主要负责接收外部传来的图像数据，然后在下一个时钟上升沿将图像数据传输给编码模块，同时完成上下文模型中前行上下文的缓存。

模块内设置一双端口 RAM 用来缓存压缩过程中用到的前行数据，编码模块计算出像素重建值 Rx 后输入该 RAM 对应地址中存储并覆盖之前的数据，等待下一行数据压缩时输出。每输出一个图像数据只需输出对应的一个前行像素重建值 Rd，编码模块会将上一个 Rd 值传递给 Rb，Rb 值传递给 Rc，完成新的上下文模型的建立。

4.4.2 梯度与 Q 值计算模块

上下文像素进入该模块中，首先进行梯度的计算，计算公式如公式(2.3)所示。此时由于 a 点的像素重建值 Ra 并未计算出，所以将用 a 点的实际像素值 Ia 替代进行梯度计算。

梯度计算完成后需要对梯度进行量化，量化的方式如公式(4.1)与公式(4.2)所示。量化后得到 q_1、q_2、q_3 的值，正如本文在 4.3.1 节所分析的，Ra 会分布在[Ia-near,Ia+near]的区间内，根据本系统的阈值设置情况，正确的 q_3 值会在 Ia 与 Rb 之间梯度差量化产生的 q_3 值的±1 范围内，可以将 q_3、q_3+1、q_3-1 所产生的三种 Q 值分别进行同样的后续压缩过程，在最后得到准确的 a 点重建像素值 Ra 后，选择准确的 Q 值所产生的误差值进行 Golomb 编码。在本阶段将会产生 2 个准确的梯度量化值 q_1、q_2 以及 3 个梯度的预测值 q_3、q_3+1、q_3-1。该步骤占用一个时

钟周期。

在下一个时钟周期将进行 Q 值的计算。通过公式(4.3)所示的方法，分别产生三个 Q 值 Q_1、Q_2、Q_3，分别对应 q_3、q_3+1、q_3-1。在实现的过程中可以对 Q 值计算中的乘法运算进行优化，转换成移位运算和加法运算，优化方法如公式(4.4)所示。

$$Q = q_1 \gg 6 + q_1 \gg 4 + q_2 \gg 3 + q_1 + q_2 + q_3 \tag{4.4}$$

计算得出的三个 Q 值，将分别送入三个相同的后续压缩模块中的参数模块中，查询该上下文情况下的参数并反馈至压缩模块中。

4.4.3 边缘检测模块

在 Q 值计算的同一个时钟周期将同时进行边缘检测，用来产生预测值 Px。Px 的产生于 Ra、Rb、Rc 有关，由于 Ra 分布在[Ia-near,Ia+near]的区间内，所以在 Ra 未确定的情况下，理论上最多会产生 2*near+1 个预测值 Px。后续在进行误差量化时，会对 Ix、Px 以及偏差值 C[Q]分别进行除法运算得到商和余数并进行合并处理，而在误差量化的除法运算中，分母为 2*near+1，大于任何两个可能的 Px 值之间的差。这种情况下我们可以先行计算其中一个 Px 值量化后的商和余数，在最后获得 Ra 准确值的时候选择正确的 Px 值，从而对商和余数进行对应的加法和逻辑判断，获得准确的 Px 量化值。

在本系统中，将直接采用 a 点的准确值 Ia 进行边缘检测。边缘检测的方法如公式(4.5)所示。

$$P_x = \begin{cases} \max(I_a, R_b) & if\ R_c < min(I_a, R_b) \\ \min(I_a, R_b) & if\ R_c > max(I_a, R_b) \\ I_a + R_b - R_c & else \end{cases} \tag{4.5}$$

4.4.4 误差量化模块

在得出待编码像素点的预测值后，将开始对像素实际值、预测值以及偏差值 C[Q]的量化。由于图像像素值的最大位宽为 16 位，根据 4.3.2 节的分析，将采取将待量化值分高位和低位分别进行量化的方式进行查表法除法运算。具体步骤如图 4.4 所示。

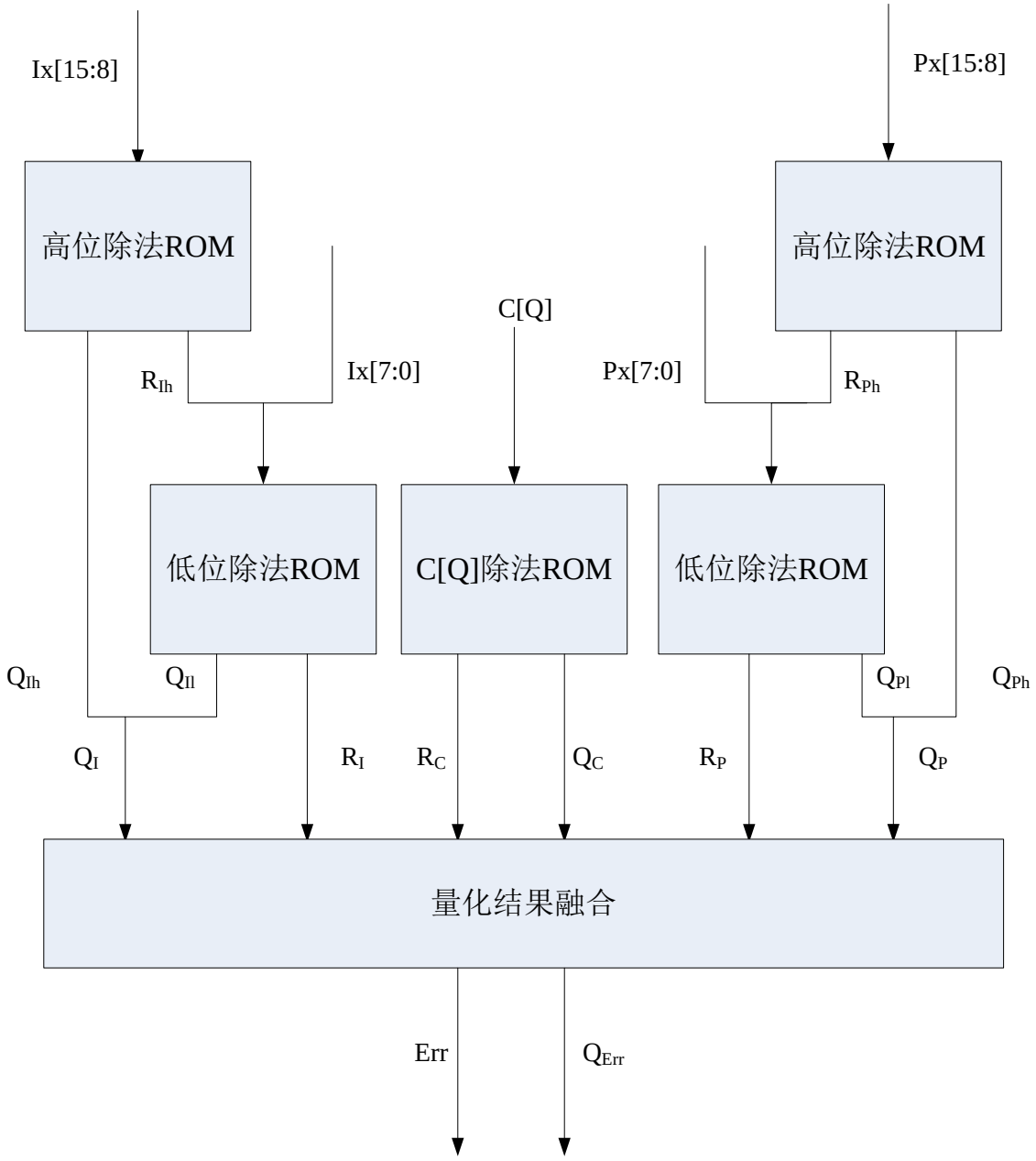

图 4.4　误差量化模块硬件结构示意图

在预测值得出后的第一个时钟周期，首先提取像素实际值 Ix 和预测值 Px 的高八位值，送入存有高八位数值除法结果的 ROM 中。由于失真量化参数 near 的值为 0 到 10，所以 ROM 的深度为 256*11，得到实际值量化后的商 Q_{Ih} 和余数 R_{Ih}，以及预测值量化后的商 Q_{Ph} 和余数 R_{Ph}。

在下一个时钟周期，就会进行 Ix 和 Px 低位的量化除法运算。存有低八位数

除法结果 ROM 的输入为 Ix 和 Px 的低八位，分别加上对应高位值量化后所得的余数 R_{Ih} 和 R_{Ph}，由于 near 最大值为 10，可知在量化时最大的除数为 2*10+1=21，所以该 ROM 的深度为(256+20)*11。在经过一个时钟周期后将会得到低位量化后的商 Q_{Il} 和 Q_{Pl}，以及余数 R_I 和 R_P。将高位和低位量化后的商结果进行混合，得到实际像素值 Ix 量化后的商 Q_I 和余数 R_I，以及预测值 Px 量化后的商 Q_P 和余数 R_P。

同时在这个时钟周期可以获得 Q 值查询参数后获得的结果 C[Q]，因为在梯度及 Q 值计算模块产生了三个 Q 值 Q_1、Q_2、Q_3，所以也会产生三个 C[Q]值分别输入三个相同的存有 C[Q]除法结果的 ROM，下个时钟周期便可以读出三组量化后的商 Q_C 和余数 R_C，这将在下个模块中重建值选择后进行修正。由于 C[Q]的最小值设置为-128，最大值设置为 127，包括 256 个值，所以 ROM 的深度为 256*11。

最后在预测值得出后的第三个时钟周期时，将对像素实际值 Ix、预测值 Px 以及 C[Q]的量化值进行融合，融合后将会得到量化误差初步结果 Err 和余数 R_{Err}。此时需要对余数 R_{Err} 进行逻辑判断，如果余数小于 0，需要将 Err 减去 1，余数 R_{Err} 加 2*near+1；而当余数大于等于 2*near+1 时，需要将 Err 加 1，余数 R_{Err} 减 2*near+1；其他情况下则维持不变。经过该判断后，得到新的误差量化结果 Err 和余数 R_{Err}。

4.4.5 重建值选择及参数更新模块

重建值选择以及参数更新模块的工作将在同一个时钟周期内完成。首先要完成预测值 Px 以及上下文参数的选择。根据上一时钟周期得到的 a 点像素重建值 Ra，重新计算预测值 Px，同时计算准确的索引值 Q，选择准确的偏差值 C[Q]及其对应的量化结果，之后与原来的 Px 作差，作差的结果将修正上一时钟周期得到的余数 RErr.。再对 RErr 进行 4.4.4 节中所述的逻辑判断，最后得到准确的误差量化结果 Err 和余数 RErr。

在得到误差量化结果后，需要计算待编码像素点的重建值 Rx。首先分析本系统中误差量化结果 Err 得到的方式，Ix 大于$(Px + SIGN * C[Q])$的值时，可以用公式(4.6)总结。Ix 小于等于$(Px + SIGN * C[Q])$的值时，则将该公式右侧取反即可。

$$\mathrm{Err} = \frac{Ix-(Px+SIGN*C[Q])+near-\mathrm{R_{Err}}}{2*near+1} \tag{4.6}$$

而 Rx 的计算方式如公式(2.8)所示，化简可得到 Rx 的计算方式如公式(4.7)所示。

$$\mathrm{Rx} = \begin{cases} Ix + \mathrm{R_{Err}} - near & if\ Ix > (Px + SIGN * C[Q]) \\ Ix - \mathrm{R_{Err}} + near & if\ Ix \le (Px + SIGN * C[Q]) \end{cases} \tag{4.7}$$

得到的误差量化结果 Err 将经过模减和映射的处理，模减的操作过程如 2.1.2.4 所述，将误差量化结果模减后再进行映射，如公式(4.8)所示。映射后的结果 MErr 将送入 Golomb 编码模块进行编码。

$$\mathrm{MErr} = \begin{cases} 2 * Err & if\ Err \ge 0 \\ -2 * Err - 1 & if\ Err < 0 \end{cases} \tag{4.8}$$

同时进行的还有 Golomb 编码中参数 K 的计算，计算过程需调用参数 A[Q]和 N[Q]。K 的取值即为使得公式(4.9)成立的最小整数。

$$(\mathrm{N[Q]} \ll \mathrm{K}) \ge \mathrm{A[Q]} \tag{4.9}$$

最后将进行上下文参数 A[Q]、B[Q]、C[Q]、N[Q]的更新。更新过程与 2.2.7 节所述过程一致。

4.4.6 Golomb 编码及码流输出模块

Golomb 编码模块接收误差量化、模减并映射后的非负整值 MErr，对其进行限长 Golomb 编码。编码过程将每 n 行分割为一个区域，压缩每个区域时将对该区域压缩码流的长度进行计数，压缩完一个区域后将长度传输至码率控制模块并重新开始计数。

每当压缩完成一个像素后，由于每个像素压缩完后的码流长度不为定值，所以需要进行组帧操作。图 4.5 展示了组帧的操作流程。

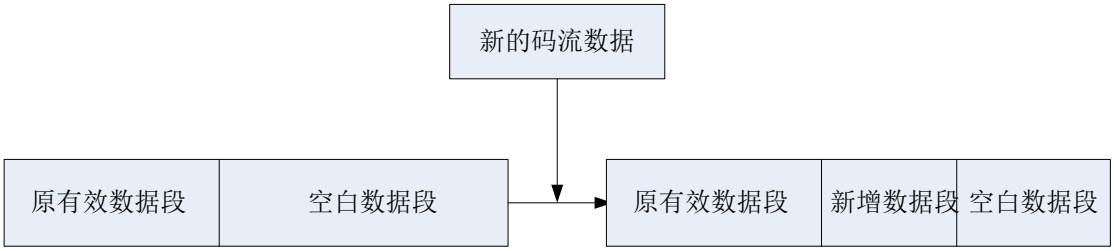

图 4.5　组帧操作

首先设置一个缓冲寄存器 buff，在本系统中限长 Golomb 编码产生的最大码

字长度为 64 位，所以 buff 设置为 64 位宽，初始数据为全 0。每当有新的压缩码流到来时，首先判断原有效数据长度 len 与新码流数据长度 newlen 之和是否大于 64。若否，则根据 buff 中原有数据长度，将新的码流数据左移 64-len 位后与 buff 进行与操作，从而将新的码流数据写入缓冲寄存器 buff，len 与 newlen 之和为新的 len 值。若是，则首先将新码流数据右移 len+newlen-64 位，与 buff 与操作后，将 buff 中的数据写入 FIFO，完成一帧码流数据的发送，之后将 buff 寄存器清空，并将新码流数据的低 len+newlen-64 位数据写入 buff 并左移 128-(len+newlen)位，新的 len 值为 len+newlen-64。

4.4.7 码率控制模块

码率控制模块负责更新图像压缩时需要用的 near 值，同时存储并更新动态码率表。其硬件结构如图 4.6 所示。在码率控制模块中，根据基于动态码率表的算法，首先在模块内设置一个 RAM 中存储一个码率调整表，为了方便计算，表内存储的是相邻 near 值调整所带来的每行像素压缩后码流长度的调整值。near 值范围为 0 到 10，所以 RAM 中共存有 10 个值。

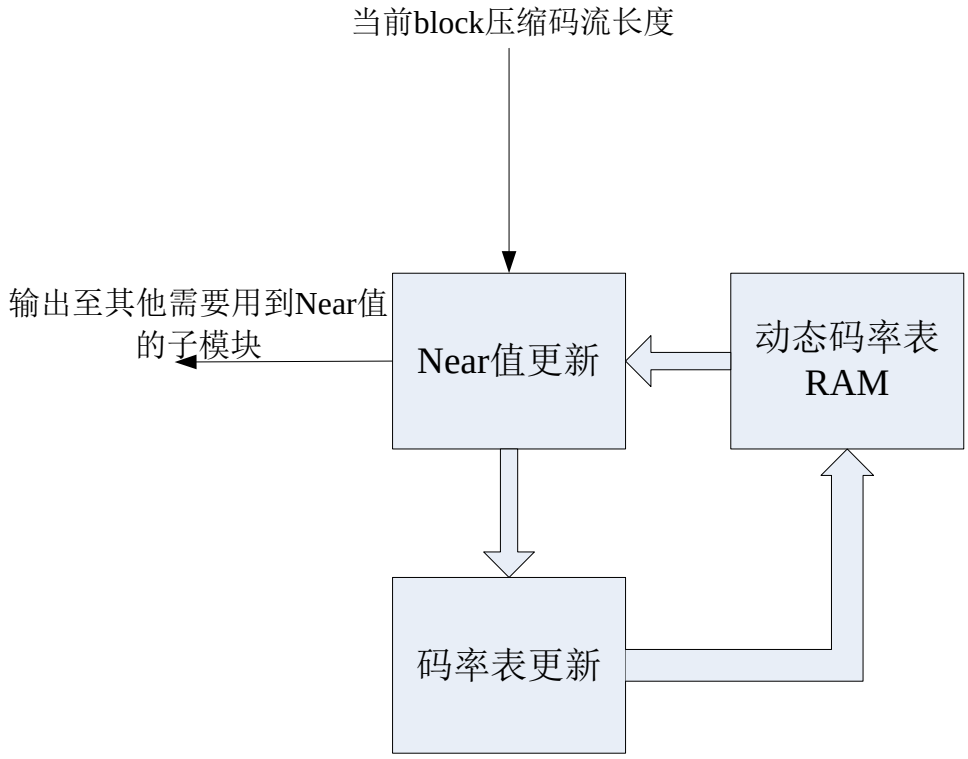

图 4.6　码率控制模块硬件结构

本系统将图像中每 n 行像素设置为一个区域，在 Golomb 编码模块完成 n 行

数据的压缩之后，对该 n 行编码所产生的码流长度进行统计并送入 Golomb 编码模块，n 一般设置为 2 的 n 次幂，经过简单的移位运算，得到当前区域每行像素压缩后的码流长度。为了保证下一区域能够及时获得新的 near 值，本系统将首先进行 near 值的调整。将目标值与当前区域每行像素压缩后的码流长度进行对比，若目标值比当前值更小，则需要逐步增大 near 值，并依次从取出 RAM 中对应 near 值带来的码率调整，从当前值中减去，直至小于 0 时停止，此时的 near 值便是下一区域所需要用到的 near 值，反之亦然。

在得到下一区域所需的 near 值后，再进行码率表的更新，此过程不需要在较少的时钟周期内完成。首先将当前区域每行像素压缩后的码流长度与上个区域的值进行比较，得到由上一个 near 值调整至当前 near 值后，所带来的码率调整。之后便根据 3.2 节所述基于动态码率调整表的算法进行码率表的更新。由于算法中需要用到除法，而码率表的更新并不需要在较短的时钟周期内快速得出，所以本文将在该模块中例化一个除法器进行计算，之后对码率表进行更新。更新完成后，将等待新一轮压缩完成后再进行下一轮的 near 值调整和码率表更新。

4.5 硬件性能分析与验证

4.5.1 Virtex4 系列器件结构简介

本文所使用 FPGA 型号为 XC4VSX35，属于 Xilinx Virtex-4 系列。Xilinx 的 Virtex-4 器件采用 90 nm 铜工艺，使用 300 mm（12 英寸）晶片技术生产。主要的片内资源有可配置逻辑块（CLB）、可编程输入输出单元(IOB)、片内块存储器(BlRAM)、数字时钟管理单元(DCM)、XtremeDSP Slice 以及丰富的布线资源。

4.5.1.1 可配置逻辑块（CLB）

可配置逻辑块是 FPGA 内的基本逻辑单元，每个可配置逻辑块由 4 个 Slice 和附加逻辑构成，通过对可配置逻辑块进行配置可以实现组合逻辑、移位寄存器或 RAM。在本型号的 FPGA 中， Slice 是由两个 4 输入函数发生器及一些逻辑单位组成。

CLB 中的 Slice 分为两种，SliceM 和 SliceL，它们的区别在于：SliceM 的查找表（LUT）结构具有 RAM 和 ROM 的功能，而 SliceL 的则没有，所以 SliceM 可以实现分布式 RAM（Distributed RAM）和移位寄存器等结构，而 SliceL 只能

实现逻辑功能。

4.5.1.2 可编程输入输出单元（IOB）

可编程输入输出单元是芯片与外界电路的接口部分，用以完成不同电气特性下对输入和输出信号的驱动与匹配要求。FPGA 的 IOB 被划分为若干个组（Bank），每组只能有一种接口工作电压，但不同组的接口电压可以不同，能够独立地支持不同的 I/O 标准，在实际设计中，EDA 软件会根据设计的实际需要来配置每组 IOB。

4.5.1.3 数字时钟管理单元（DCM）

DCM 是 FPGA 芯片中集成的用以消除时钟偏斜和进行时钟相位调整的时钟管理单元。利用 DCM 可以轻松地实现全局时钟的设计，并可以方便地对时钟进行倍频、分频、相移等操作。

4.5.1.4 片内块存储器（BRAM）

片内块存储器是 FPGA 内部的专用存储器模块。BRAM 的存在大大拓展了 FPGA 的应用范围及灵活性。本型号芯片中每块 BRAM 大小为 18Kbit，该片内块存储器可以配置成单端口 RAM、双端口 RAM、内容地址存储器 CAM 以及 FIFO 等常用存储结构。

CLB 中的 SliceM 实际上也可以综合为以上存储结构，通常称为分布式 RAM（Distributed RAM），当 BRAM 的数量不足时，可以通过将逻辑资源 SliceM 综合为分布式 RAM 进行补充。

4.5.1.5 XtremeDSP Silce

Xilinx 器件的独有结构，每个 XtremeDSP Slice 包含一个 18 x 18 位乘法器、一个加法器和一个累加器 XtremeDSP Slice 可通过 IP 核的形式方便地调用。同时 XtremeDSP Slice 中每个乘法器或累加器也能够独立使用。

4.5.2 性能分析

误差量化模块是本系统的核心模块，首先需要对该模块进行功能仿真，以验证其逻辑功能是否正确。其功能仿真结果如图 4.7 所示。

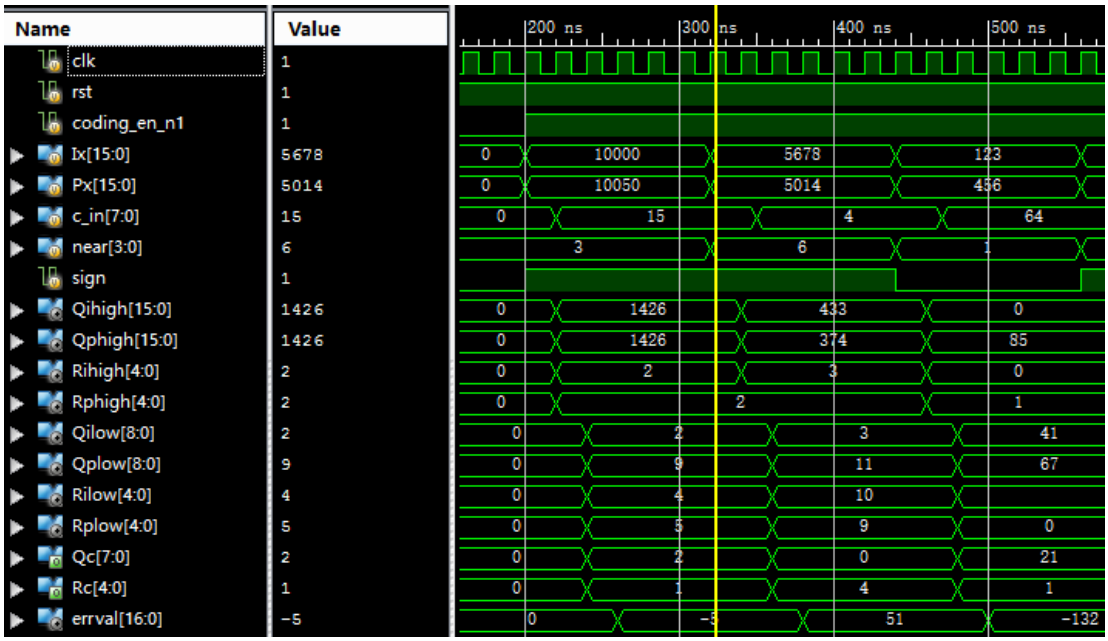

图 4.7　误差量化模块功能仿真结果图

本文通过 testbench 文件向模块中注入 Ix，Px，c_in，near 以及 sign 的值，观察模块所输出的误差量化结果 errval 是否正确来判断该模块是否正常工作。根据以上仿真结果可以看出，高位的量化结果于输入数据注入后第 1 个时钟周期给出，低位以及 c_in 的误差量化结果于第 2 个时钟周期给出,最后在输入数据注入的第 3 个时钟周期，得到输入数据所对应的误差量化值 errval，符合本系统的设计初衷。

经综合后，该系统的资源占用情况如表 4.1 所示，综合后该图像压缩系统对图像的处理速度可达 60M Pixel/s，每秒可处理 14 帧分辨率为 2048*2048 的 8~16bit 图像。

表 4.1　JPEG-LS 图像压缩 FPGA 系统资源占用情况

资源	使用数量	可用数量	利用率
Slice	7187	15,360	46%
LUT	12094	30,720	39%
FIFO16/RAMB16s	85	192	44%

4.5.3 测试平台的搭建

在完成设计步骤之后，本文通过搭建如图 4.8 所示的测试平台，将生成的二进制文件烧录入 FPGA，对设计进行板级仿真和验证。

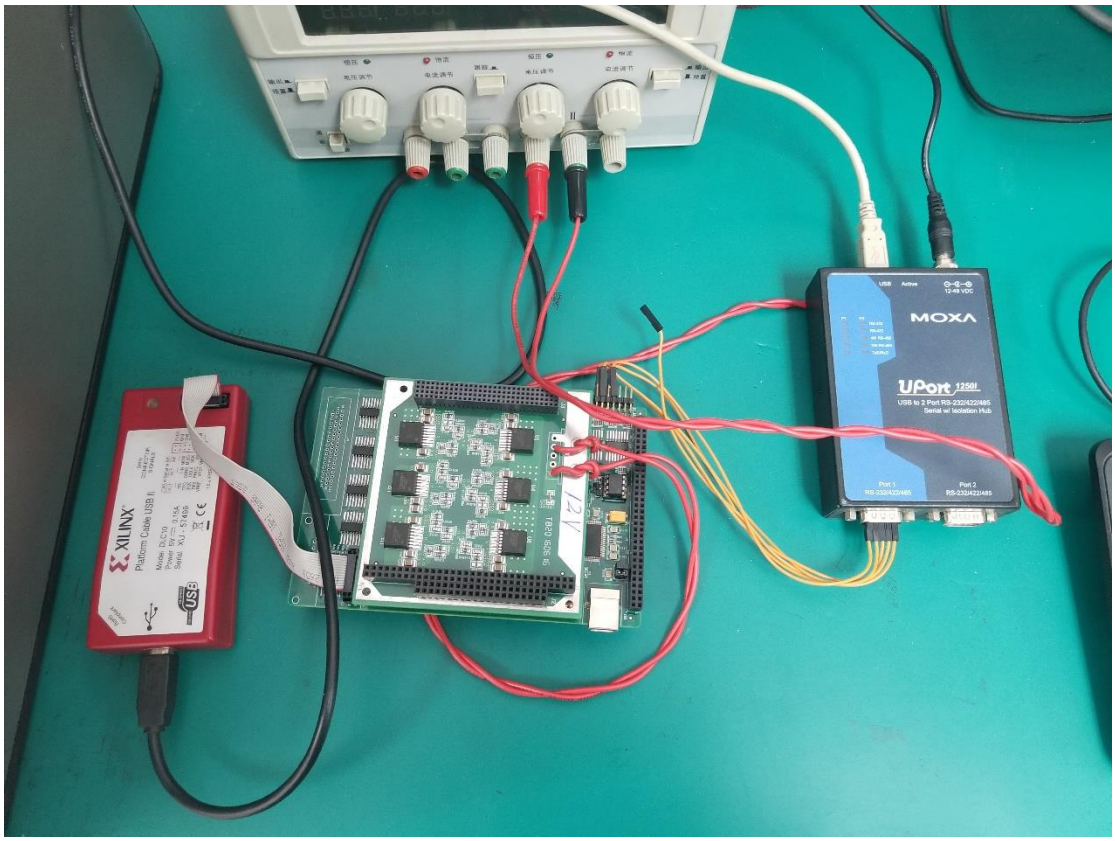

图 4.8　验证平台

本文于 JPEG-LS 近无损图像压缩编码器之外，编写了用于数据传输的 UART 串口通信模块并与 JPEG-LS 编码器进行连接，使上位机可以将待压缩的图像文件通过 RS-422 串口传输至 FPGA 芯片的片内 RAM 中，再由编码器从 RAM 中读出图像数据并进行压缩，压缩完成之后的码流输出至 FIFO，最后由 FIFO 通过 UART 串口通信模块将压缩码流传回上位机，由上位机存储为二进制文件。码流文件可以使用上位机解码软件进行解码并与原图像进行比对，完成验证过程。本文采用串口调试助手实现图像文件的发送以及压缩码流的接收功能，示例如图 4.9 所示。

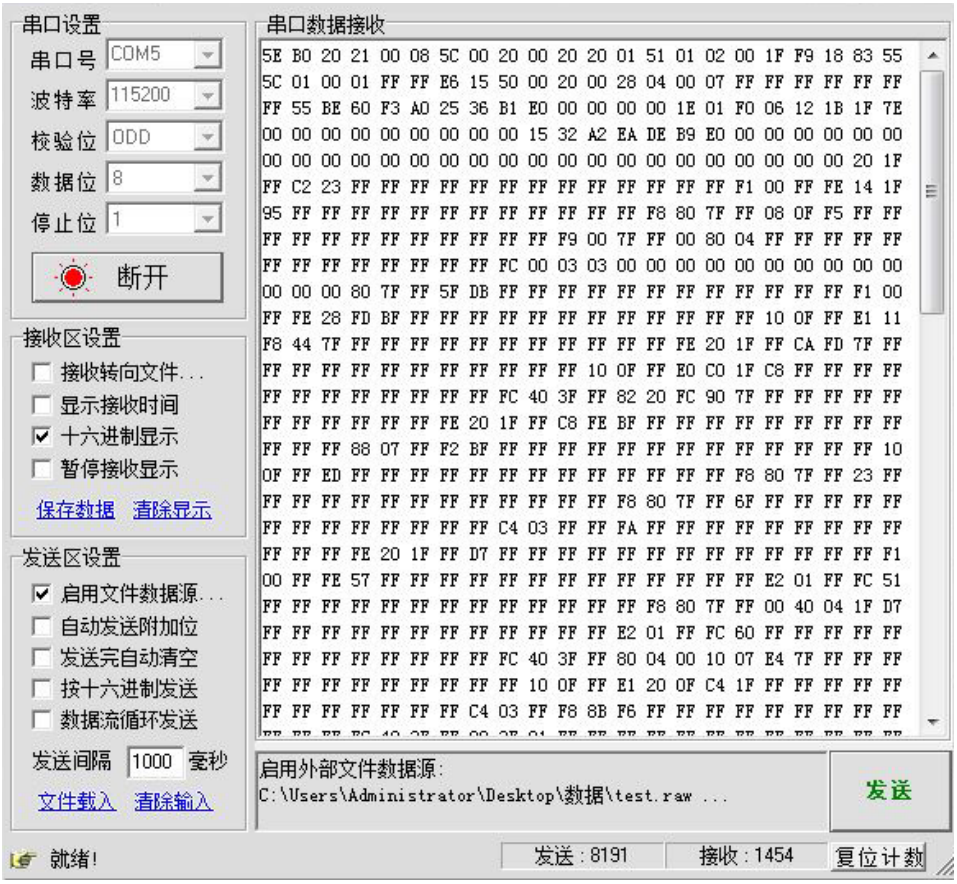

图 4.9　上位机与 FPGA 编码器通过串口通信示例

整个验证流程的数据流如图 4.10 所示。

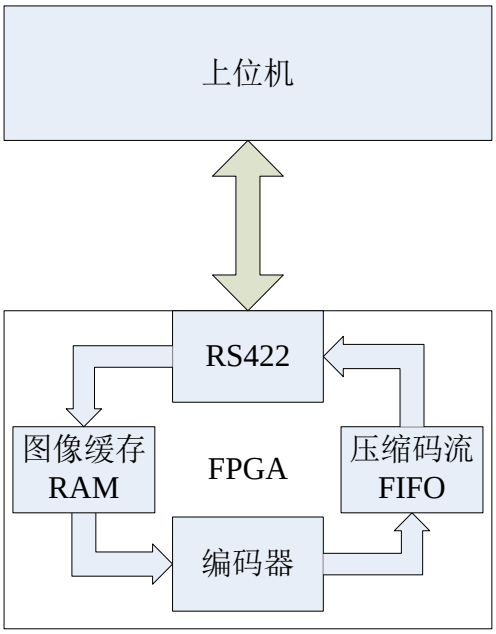

图 4.10　编码器板级验证流程图

4.6 本章小结

本章中首先介绍了之所以选择 FPGA 作为星载图像压缩平台的理由，并介绍了 FPGA 设计的流程。之后，针对前一章所述的 JPEG-LS 码率控制算法，设计了一套基于 FPGA 的码率可控 JPEG-LS 图像近无损压缩系统。最后对该系统进行了验证，并对系统性能进行了分析。

第5章 总结与展望

随着我国航天事业的蓬勃发展，卫星发射越来越密集，通过卫星所获得的图像数据量也随之越来越庞大。如何利用有限的星载系统硬件资源和传输带宽将遥感图像回传至地面，成为了亟待解决的问题。在这一情境下，设计一套高效易用的星载图像压缩系统变得愈发重要。

JPEG-LS 图像压缩算法具有硬件实现简单、压缩效果较好的特点，适用于无损及近无损图像压缩，但也有码率不可控的缺点。本文基于该标准，在研究了过往的 JPEG-LS 码率控制算法后，提出了一种基于动态码率表的码率控制算法，并进行了硬件实现。

本文主要工作和创新点为：

1.详细分析了 JPEG-LS 图像压缩标准和分析了过往的码率控制算法，在此基础上提出一种基于动态码率表的码率控制算法。使用 8bit 和 16bit 图像分别就新旧算法进行测试。测试结果表明，在同样的目标码率下，基于动态码率表的码率控制算法能取得比以往的码率控制算法更快的收敛速度，同时压缩后重建图像的质量更好。

2.根据基于动态码率表的码率控制算法，采用 FPGA 器件，使用 Verilog HDL 语言对其进行了硬件实现。在硬件实现的过程中，就误差量化中的 16bit 除法提出了新的结构，并就图像重建值的预测与选择、码率动态控制等在硬件实现中遇到的问题进行了优化，最终在 FPGA 上实现了码率可控的 JPEG-LS 近无损图像压缩算法编码器。

同时，由于本文研究时间限制和作者的水平局限，对 JPEG-LS 图像压缩算法及其码率控制算法的认识还有很多不足，在以下两个方面还有更多探索和改进的空间：

1. JPEG-LS 算法是基于预测方式的图像压缩算法，在解码时都要依赖已解码的像素，若一个像素的解码出错，将导致整个图像重建失败。如果能研究出一种行之有效的抗误码算法，将大幅提高 JPEG-LS 的性能和可用性。

2.在第 3 章中，通过不同图像压缩的参数可知，游长模式所编码像素的数量

与图像压缩率之间存在着某种数学关系，如果能建立一个数学模型来较为精确地描述这种关系，将对 JPEG-LS 的码率控制更有帮助。

参考文献

[1]吴乐南.静止图象压缩标准新进展[J]. 电脑应用技术, 1998(42):8-11.

[2]张兆亮.基于 CCSDS 算法的星载图像压缩系统的 FPGA 实现[D].西安电子科技大学，2010.

[3]王伊洛.EBCOT 中 T2 编码器的 VLSI 设计[D].西安电子科技大学，2009.

[4]王春秋.基于 CCSDS 标准的遥感图像压缩平台硬件系统设计[D].西安电子科技大学，2009.

[5]房鹤.卫星图像实时压缩设备中的关键技术研究[D].国防科学技术大学，2006.

[6]张春田，苏玉挺，张静.数字图像压缩编码[M]. 清华大学出版社, 2006, 1

[7]Weinberger M J. From LOCO-I to the JPEG-LS Standard[M]. From LOGO-I to the JPEG-LS standard. 1999:68-72 vol.4.

[8]Information Technology—Lossless and Near-Lossless Compression of Continuous-Tone Still Images[M]. ISO/IEC14495-1, ITU Recommendation T.87, 1999.

[9]徐欣锋，黄廉卿，徐抒岩，等.高空间分辨率遥感图像实时压缩进展[J]. 光学精密工程, 2004, 12: 266-271.

[10]Sweldens W. The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets[J]. Applied & Computational Harmonic Analysis, 1996, 3(2):186-200.

[11]Sweldens W. The lifting scheme: a construction of second generation wavelets[J]. Siam J.math.anal, 1997, 29(2):511-546.

[12]Daubechies I, Sweldens W. Factoring wavelet transforms into lifting steps[J]. Journal of Fourier Analysis & Applications, 1998, 4(3):247-269.

[13]Shi Q Y. Biorthogonal wavelet theory and techniques for image coding[C]// International Symposium on Multispectral Image Processing. International Society for Optics and Photonics, 1998:24-32.

[14]Taubman D S, Marcellin M W. JPEG2000: standard for interactive imaging[J]. Proceedings of the IEEE, 2002, 90(8):1336-1357.

[15]Rabbani M, Joshi R. An overview of the JPEG 2000 still image compression standard[J]. Signal Processing Image Communication, 2002, 17(1):3-48.

[16]Itu-T I. Information technology—JPEG 2000 image coding system: Core coding system[J].

ITU-T Recommendation T.800 and ISO/IEC International Standard, 2002, 21(1):27-51. [17] S．W．Golomb．Run-length encodings[J]，IEEE Trans．Inform．Theory，1966，6(12)：399~401．

[18]戴鑫.JPEG-LS 图像无损压缩 IP 核的 FPGA 设计[D].太原理工大学,2010.

[19]Meyr H, Rosdolsky H, Huang T. Optimum Run Length Codes[J]. Communications IEEE Transactions on, 1974, 22(6):826-835.

[20]Weinberger M J, Seroussi G, Sapiro G. LOCO-I: a low complexity, context-based, lossless image compression algorithm[C]// Data Compression Conference, 1996. DCC '96. Proceedings. IEEE, 2002:140.

[21]吕雪莹.基于 Curvelet 变换的图像去噪算法研究[D].哈尔滨工业大学,2015.

[22]李春.基于小波分解的图像多分辨率非局部去噪方法的研究[D].天津大学,2009.

[23]Wang Z, Bovik A C, Sheikh H R, et al. Image quality assessment: from error visibility to structural similarity[J]. IEEE Transactions on Image Processing, 2004, 13(4):600-612.

[24]侯淑维，董刚，蒙红英.用于 JPEG-LS 无损压缩算法防码流溢出的图像预处理方法及装置：中国，CN201711158299.3[P].2018-04-06.

[25]Jiang J, Yang S. Rate-controlled near-lossless image codec based on visual perception and content adaptability[J]. Proceedings of SPIE - The International Society for Optical Engineering, 2001, 4209:262-272.

[26]Tsai T H, Kao S C, Lee Y X. The segment-based rate control algorithm in JPEG-LS for bandwidth-efficiency applications[C]// IEEE International Conference on Multimedia and Expo. IEEE, 2008:793-796.

[27]Jiang J, Reddy M. Open-loop rate control for JPEC-LS near lossless image compression[J]. Electronics Letters, 1999, 35(6):465-466.

[28]Chou C H, Liu K C. A perceptually optimized JPEG-LS coder for color images[C]// Iasted International Conference on Signal Processing, Pattern Recognition, and Applications. ACTA Press, 2007:26-32.

[29]徐燕凌，刘蓓.JPEG-LS 图像压缩动态码率控制策略[J]. 计算机工程,2008,34(7):238-239.

[30]张毅，雷杰，李云松.一种新的基于先验数据表的 JPEG-LS 动态码率控制算法[J]. 电子与信息学报，2014,36(4):823-827.

[31]宋正勋，胡贞，许红梅.DSP 器件的原理及应用[J].长春光学精密机械学院学

报,1999(02):62-67.

[32]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.

[33]张毅.码率可控的 JPEG-LS 近无损图像压缩编码器硬件实现[D].西安电子科技大学,2014.

致谢

近三年的硕士学习生涯如白驹过隙，入学报到时的场景还恍如昨日，历历在目。一路走来，无论是雁栖湖时的课程学习，还是中关村的科研生活，都让我受益匪浅。我相信，这三年将是我人生中最值得回忆的时光之一。

感谢我的导师周盛雨老师，为我提供了优越的科研环境，不仅在科研上指导我进步，同时也在生活中关心我的成长，解答我学习、科研以及求职时的疑惑。老师的学识、为人都值得我学习，是我今后工作和生活的榜样。

感谢张学全老师在繁忙的科研工作中抽出宝贵的时间解答我学习以及科研中的疑惑，与张老师的交谈让我感到如沐春风，使我对科研的理解与认识更上一层楼，在此向张老师致以真诚的谢意。

感谢安军社老师、张忠伟老师的关心和支持，以及孙建伟同学、郑铁同学的帮助，同时也要感谢电子室各位老师和职工对我的帮助。

感谢和我同一间实验室所有的同学，和他们在同一个屋檐下，我经历了两年难忘的时光，感谢他们在我学习和生活上的帮助。感谢我的两位舍友，感谢他们在三年的宿舍生活中给予我的帮助和支持。

感谢我的父母对我的养育之恩，是他们抚养我成人，他们无条件支持着我的求学，在关键事情上尊重我的选择和决定。在今后的日子里我将努力工作，报答他们。

感谢所有帮助过、关心过我的人，感谢曾经那个毛躁、自卑，但是从未放弃过的自己。