学校代码：10286
分类号：TN492
密　级：公开
U D C：621.3
学　号：163804

東南大學
工程硕士学位论文

一种图像无损压缩算法 JPEG-LS 的 FPGA 实现

（学位论文形式：应用研究）

研究生姓名：**尤传亮**
导 师 姓 名：**徐平平 教授**
李　冰 教授
吕耀安 高工

申请学位类别 **工程硕士**　学位授予单位 **东 南 大 学**
工程领域名称 **集成电路工程**　论文答辩日期 **2019 年 5 月 24 日**
研究方向 **通信与信息处理集成电路**　学位授予日期 **20　年　月　日**
答辩委员会主席 **李祖华**　评　阅　人 **张萌**
李祖华

2019 年　月　日

万方数据

東南大學

工程硕士学位论文

一种图像无损压缩算法 JPEG-LS 的 FPGA 实现

专 业 名 称：集成电路工程

研究生姓名：尤传亮

导 师 姓 名：徐平平

李　冰

校 外 导 师：吕耀安

2019 年　　月　　日

万方数据

FPGA Implementation of a Lossless Image Compression Algorithm JPEG-LS

A Thesis Submitted to

Southeast University

For the Academic Degree of Master of Engineering

BY

YOU Chuanliang

Supervised by

Professor XU Pingping

and

Professor LI Bing

and

Senior Engineer LV Yaoan

School of Microelectronics

Southeast University

May 2019

万方数据

东南大学学位论文独创性声明

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

研究生签名：_____日期：_____

东南大学学位论文使用授权声明

东南大学、中国科学技术信息研究所、国家图书馆、《中国学术期刊（光盘版）》电子杂志社有限公司、万方数据电子出版社、北京万方数据股份有限公司有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括以电子信息形式刊登）论文的全部内容或中、英文摘要等部分内容。论文的公布（包括以电子信息形式刊登）授权东南大学研究生院办理。

研究生签名：_____导师签名：_____日期：_____

万方数据

摘要

对大量分辨率不断提高的图像进行实时处理，给系统的存储容量以及传输带宽带来极大的挑战。在图像压缩的国际标准中，JPEG-LS 具有无损压缩和近无损压缩两种功能，其中无损压缩在医学影像、生物特征、卫星遥感等领域具有重要的应用价值。虽然目前基于软件的压缩已有很多非常成熟的解决方案，但是普遍存在占用中央处理器资源过多、功耗较大以及处理速度已不能与接口速度相匹配等问题，JPEG-LS 无损压缩算法的硬件设计与实现成为解决此问题的关键。本文研究 JPEG-LS 无损压缩算法的 FPGA 实现具有工程价值。

传统的基于 FPGA 实现的 JPEG-LS 编码器存在时钟频率低、吞吐率小等问题，本文设计了一种改进的 JPEG-LS 编码器。整个 JPEG-LS 编码器的硬件设计包括上下文建模模块、正常编码模块、游程编码模块以及输出模块。对上下文因果模板的构建过程进行优化，一方面利用上一个已编码像素因果模板对本次待编码像素因果模板的构建过程进行优化，另一方面在上下文参数更新过程中完成下一个待编码像素因果模板的构建；利用二分查找法对 Golomb 编码参数 k 的计算过程进行优化。

本文所设计的 JPEG-LS 编码器通过 Modelsim 功能仿真，并基于 Xilinx FPGA 实现。实验结果表明，本文所设计的 JPEG-LS 编码器消耗 1521 Registers 和 3373 LUTs，能够实现对灰度图像的无损压缩，最大时钟频率达到 115.1MHz，吞吐率为 36.8Mbps，压缩比为 1.9:1，达到了预期设计指标。

关键词： JPEG-LS，无损压缩，上下文建模，Golomb 编码，现场可编程门阵列

万方数据

万方数据

Abstract

Real-time processing of a large number of images with increasing resolution poses great challenges to the storage capacity and transmission bandwidth of the system. In the international standard of image compression, JPEG-LS has two functions of lossless compression and near lossless compression. Lossless compression has important application value in medical imaging, biometrics, satellite remote sensing and other fields. Although there are many mature solutions for software-based compression, there are generally problems such as excessive CPU resources, high power consumption, and processing speed that cannot match the interface speed. JPEG-LS lossless compression algorithm Hardware design and implementation is the key to solving these problems. This thesis studies the FPGA implementation of JPEG-LS lossless compression algorithm with engineering value.

The traditional JPEG-LS encoder based on FPGA has low clock frequency and low throughput. In order to improve the clock frequency and increase the throughput rate, an improved JPEG-LS encoder is designed. The hardware design of the entire JPEG-LS encoder includes a context modeling module, a regular encoding module, a run-length encoding module, and an output module. The construction process of context causal template is optimized. On the one hand, the previous pixel causal template is used to optimize the construction process of the pixel causal template to be encoded, and on the other hand, the next pixel causal template to be encoded is constructed in the context parameter updating process. And the calculation process of the Golomb coding parameter k is optimized by a binary search method.

The JPEG-LS encoder designed in this thesis is simulated by Modelsim and implemented on Xilinx FPGA. The experimental results show that the JPEG-LS encoder designed in this thesis consumes 1521 Registers and 3373 LUTs, which can achieve lossless compression of grayscale images. The maximum clock frequency is 115.1MHz, and the throughput rate is 36.8Mbps and the compression ratio is 1.9:1, which have reached the expected targets.

Key words: JPEG-LS, Lossless Compression, Context Modeling, Golomb Encoding, FPGA

万方数据

万方数据

目录

万方数据

万方数据

插图目录

万方数据

万方数据

万方数据

万方数据

表格目录

万方数据

万方数据

缩略词

DCT	Discrete Cosine Transform	离散余弦变换
DWT	Discrete Wavelet Transform	离散小波变换
EBCOT	Embedded Block Coding with Optimized Truncation	嵌入式多分辨率编码
FFT	Fast Fourier Transformation	快速傅立叶变换
FPGA	Field Programmable Gate Array	现场可编程门阵列
HDL	Hard Description Language	硬件描述语言
IP	Intellectual Property	知识产权
ISO	International Organization for Standardization	国际标准化组织
ITU-T	ITU Telecommunication Standardization Sector	国际电联电信标准化部门
JPEG	Joint Photographic Experts Group	联合图像专家组
JPEG-LS	Joint Photographic Experts Group-Lossless	连续色调静止图像无损及近无损压缩
LOCO-I	LOw COmplexity LOssless COmpression for Images	图像低复杂度无损压缩
PSNR	Peak Signal to Noise Ratio	峰值信噪比
RAM	Random Access Memory	随机存取存储器
ROM	Read-Only Memory	只读存储器
RTL	Register Transfer Level	寄存器传输级
VLSI	Very Large Scale Integration	超大规模集成电路

万方数据

万方数据

第一章 绪论

1.1 课题背景与意义

随着图像传感器技术的快速发展，其分辨率不断提高，需要实时处理的图像数据量也随之越来越大，这将给整个系统的存储容量以及传输带宽造成极大的挑战[1]。虽然通过提高存储容量以及传输技术可以解决该问题，但是高容量的存储系统十分昂贵，同时数据在不同存储系统之间的传输也会消耗大量的时间[2]。因此，图像压缩成为人们研究的热点。为了满足人们对图像质量不断提高的要求，无损压缩在医学影像、生物特征、卫星遥感以及专业摄影等领域得到了广泛应用[3]，这也反过来促进了图像压缩技术的进一步发展。

数字图像通常包含的大量冗余信息为图像压缩提供了理论上的可能。与图像数据相关的冗余有3 种类型：编码冗余，像素间冗余和心理视觉冗余[2-3]。因此，通常会对图像进行压缩处理以减小冗余，从而降低图像数据对存储空间以及传输带宽的需求。图像压缩方法按照压缩后图像能否还原为源图像一般可分为两种，一种是基于有损压缩技术的方法，在压缩过程中有图像数据的丢失，压缩后的图像数据不能通过解码器还原为源图像，另一种是基于无损压缩技术的方法，在压缩过程中没有图像数据的丢失，压缩后图像数据能够通过解码器还原为源图像。由于自然图像、人物图像等图像容易获取，一般可对其进行有损压缩，而医学图像[4]、指纹图像[5]、卫星遥感图像[6]等图像十分珍贵，一般需要对其进行无损压缩。

JPEG-LS（Joint Photographic Experts Group-Lossless）是由国际标准化组织（International Organization for Standardization, ISO）于1999年12月发布的ISO-14495-1标准和国际电联电信标准化部门（ITU Telecommunication Standardization Sector, ITU-T）于1998年6月发布的ITU-T.87标准，是连续色调静止图像的无损及近无损压缩标准[7]。JPEG-LS算法基于惠普实验室开发的LOCO-I（LOw COmplexity LOssless COmpression for Images）算法[8]，提供了无损压缩和近无损压缩功能。由于该算法在复杂性和效率之间具有良好的平衡性，因此被选入JPEG-LS标准[9-10]。该算法没有诸如快速傅立叶变换（Fast Fourier Transformation, FFT）或离散余弦变换（Discrete Cosine Transform, DCT）的数据变换以及其他复杂的数据转换过程，只需实现数据的减法、移位以及其他类似的、简单的处理过程[11]。与JPEG（Joint Photographic Experts Group）、JPEG2000等算法相比，JPEG-LS算法具有复杂度低，易于硬件实现等特点[12-13]。

虽然目前基于软件的压缩已有很多非常成熟的解决方案，但是普遍存在占用中央处理器资源过多、功耗较大以及处理速度已不能与接口速度相匹配等问题。在逐渐提升的数据存储器访问速度和网络带宽的环境下，通过软件实现的图像无损压缩逐渐呈现出性能瓶颈状态。因此，JPEG-LS无损压缩算法的硬件设计与实现成为解决此问题的关键。由于现场可编程门阵列（Field Programmable Gate Array, FPGA）具有逻辑设计功能强大、开发周期短、投资成本低以及处理速度快等方面的优点，因此基于FPGA实现的JPEG-LS编码器成为人们研究的重点。但是，传统的基于FPGA实现的JPEG-LS编码器往往存在时钟频率低、吞吐率小等问题，而且为了降低硬件实现的复杂性以及减少所设计编

万方数据

码器硬件资源的消耗，舍弃JPEG-LS标准中的游程长度编码成为部分研究者的选择。为了提高时钟频率，增大吞吐率，保证JEPG-LS算法结构的完整性，本文基于FPGA平台对JPEG-LS无损压缩算法进行了优化与实现。

1.2 国内外研究现状

现如今，JPEG-LS 无损压缩在医学影像、生物特征、卫星遥感以及专业摄影等领域得到了越来越广泛的应用，成为当下图像压缩的热点，且主要集中于硬件实现方面。

在国外的研究成果中，2001年，Klimesh M、Stanton V和Watola D等人为了降低硬件实现的复杂性，对LOCO-I算法进行了修改，忽略了其中的游程编码模式[14]。FPGA实现的时钟频率达到12MHz，压缩速度可达1.33Mpixels/second。该算法的FPGA实现代表了符合空间要求的图像无损压缩硬件方面的显著进展。2009年，Merlino P和Abramo A等人提出了一种新的设计方案，在不修改原始压缩方案的情况下，该设计通过流水线结构来充分利用算法的顺序性[15]。但是，该编码器电路所获得的最大时钟频率约为21MHz，解码器电路所能获得的最大时钟频率也仅有16MHz，这就限制了整个压缩系统的图像处理速度。2011年，Daryanavard H、Abbasi O和Talebi R等人提出了一种全流水线结构，以减少存储容量，提高工作频率[16]。通过对JPEG-LS算法进行优化，将存储容量减少了24%；通过使用向前预测技术来避免所提出的全流水线结构中的竞争冒险，将电路的工作频率提高到155.2MHz，对于任何尺寸为512*512的图像的压缩都可以在1700μs内完成。2013年，Kim B S、Baek S和Kim D S等人提出采用高效的全流水线方案来设计JPEG-LS编码器[17]。编码器的处理速度达到120Mpixels/second。2015年，Mert Y M提出了一种基于FPGA实现的板载实时无损压缩的JPEG-LS编码器[18]。利用流水线数据向前预测技术优化预测误差计算，利用查找表优化游程长度编码，显著提高了编码器的处理速度，同时显著降低了FPGA硬件资源的占用。2016年，Nazar F和Murugan S等人提出了一种基于MATLAB实现的用于实时应用的JPEG-LS无损压缩算法[19]。2016年，Bhimani D、Chaurasiya P N和Sedani B等人专门针对使用图像压缩的小型应用程序对LOCO-I算法进行了优化[20]。该算法使用与JPEG-LS相同的上下文和因果模板进行预测，使得压缩比提高了5%-10%。虽然对于尺寸较小的图像来说压缩比的提高是较大的，但是对于尺寸较大的图像来说压缩比的提高就显得微不足道。2016年，Chen S L、Liu T Y和Shen C W等人提出了一种用于VLSI（Very Large Scale Integration）实现的基于JPEG-LS的新型近无损彩色滤镜阵列图像压缩算法[21]。该算法由像素恢复、预测、游程编码模式以及熵编码模块组成。在采用90nm CMOS工艺综合情况下，仅包含10.9K门数，面积为30625μm^2。相比以往的JPEG-LS设计，门数分别减少了44.1%和41.7%，非常适用于开发无线视频胶囊内窥镜系统。2018年，Jallouli S、Zouari S和Masmoudi N等人提出了一种基于自适应块的直方图包装方案，以提高稀疏和局部稀疏直方图图像的JPEG-LS压缩性能[22]。2018年，Haddad S、Coatrieux G和Cozic M等人提出了第一个用于保护医学图像的联合水印加密的JPEG-LS压缩方案[23]。保证系统在加密和压缩域中的可靠性，同时最小化图像的失真。

在国内的研究成果中，2006 年，刘波和姜宏旭等人在对 JPEG-LS 算法进行改进的基础上，提出了一种由去相关处理、熵编码及压缩位率控制等三部分组成的设计方案[24]。该方案能够实现对输入

万方数据

速度为 8MB/s 的源图像数据进行实时压缩。2010 年，韩俊萍、程永强和戴鑫等人基于 Spartan 3S500E 完成了 JPEG-LS 图像无损压缩标准 IP 核的实现，硬件资源占用少[25]。2011 年，陈军、王怀超和顾晓东等人提出采用流水线和并行技术完成基于 LOCO-I 算法的高速星载系统的无损压缩核的 FPGA 实现，50MHz 时钟频率下吞吐率达到 400Mbps[26]。2013 年，王舒瑶提出采用四级反馈和预测参数的方法来解决由上下文更新所引入的数据相关性，使其能够采用全流水结构进行设计，同时硬件资源消耗较少[27]。但是所提出的编码器的最大时钟频率也仅为 77.1MHz，而且不能完全按照 JPEG-LS 的规定来处理连续出现的游程区域。2014 年，张毅提出了一种改进的 JPEG-LS 控制算法，将图像分块压缩，根据当前码率偏差和先验数据表来调整阈值参数 NEAR，并以此来控制下一个码块的输出码率，具有码率控制精度高、码率收敛速度快等优点[28]。2016 年，聂永康、雷杰和李云松等人提出了一种全新的 JPEG-LS 近无损编码器 VLSI 结构，具有编码速度快、不需要片外存储器、功耗低等优点[29]。2016 年，范文晶、王召利和王惠娟等人提出了一种基于 FPGA 全流水线结构实现的 JPEG-LS 无损压缩算法[30]。通过采用多级流水线来降低每一级运算的延迟，最大时钟频率可达 120MHz。2017 年，蒙红英、柴昱洲和韩宇等人针对游程模式在 Near 值调整中造成恢复数据的“水平线状纹理”，提出了一种基于非均匀划分机制的字典稀疏方案[31]。该方案易于硬件实现，有效改善了“线状纹理”现象。2018 年，陈聪、张学全和周盛雨等人提出了一种新的基于动态码率表的 JPEG-LS 码率控制算法，控制精度高，压缩后恢复图像的信噪比高，最高时钟频率可达 60MHz[32]。2018 年，李长兴提出对待编码像素因果模板中位置 a 上的重建值的存储进行优化，并且使用向前预测的方式对变量 C[Q] 的更新过程进行优化，减少了逻辑资源的占用，使编码器的时钟频率达到 41MHz[33]。但是所设计的编码器没能实现游程编码设计，这或多或少会影响到编码器的性能，尤其是对于平坦区域较多的图像来说，游程编码将提高图像的压缩比，减少图像的压缩时间。

综上所述，随着人们不断地对JPEG-LS算法的软硬件实现进行研究，其压缩性能有了一定的提高，在压缩比、压缩速度、应用范围以及面积功耗等方面都取得了不同程度的提高。但是，随着高速接口设备的普及以及人们对JPEG-LS算法压缩性能不断提高的要求，JPEG-LS的压缩速度仍然是研究中无需累赘的重点之一，因此研究更高压缩性能的JPEG-LS算法及其硬件实现很有意义。

1.3 研究内容与设计指标

1.3.1 研究内容

在阅读国内外有关JPEG-LS编码器相关文献的基础上，进行JPEG-LS编码器的硬件设计，包括上下文建模模块、正常编码模块、游程编码模块以及输出模块，并通过Modelsim功能仿真以及FPGA验证，实现对比特深度为8的灰度图像的无损压缩。主要研究内容如下。

（1）对JPEG-LS无损压缩算法进行研究与优化，使JPEG-LS算法的处理过程适合硬件电路的并行化处理，并确定JPEG-LS编码器硬件设计方案。

（2）使用 Verilog 硬件描述语言（Hard Description Language, HDL）进行 JPEG-LS 编码器上下文建模模块、正常编码模块、游程编码模块以及输出模块等的寄存器传输级（Register Transfer Level, RTL）设计，并将其集成为完整的 JPEG-LS 编码器。为提高所设计 JPEG-LS 编码器的时钟频率以及

万方数据

吞吐率，在硬件设计与实现过程中采用了多种方法对其进行优化。对上下文因果模板的构建过程进行优化，一方面利用上一个像素因果模板对本次待编码像素因果模板的构建过程进行优化，另一方面在上下文参数更新过程中完成下一个待编码像素因果模板的构建；利用二分查找法对 Golomb 编码参数 k 的计算过程进行优化。

（3）通过Modelsim对本文设计的JPEG-LS编码器及其各个模块进行功能仿真，在ISE 14.7开发环境下对设计进行综合，给出资源利用报告和时序报告，并对其进行结果分析。

（4）本文设计的JPEG-LS编码器在FPGA平台上实现，并对其进行结果分析。

1.3.2 设计指标

本文进行 JPEG-LS 无损压缩算法的硬件设计与 FPGA 实现，预期达到如下的功能指标与性能指标。

功能指标：

（1）通过设计上下文建模模块、正常编码模块、游程编码模块以及输出模块等，完成 JPEG-LS 无损压缩算法的 FPGA 实现；

（2）能够对比特深度为 8 的灰度图像进行无损压缩，产生编码码流，并且能够通过 JPEG-LS 软件解压。

性能指标：

（1）所设计的JPEG-LS编码器在FPGA平台上实现，时钟频率达到80MHz以上，吞吐率达到25Mbps以上；

（2）所设计的JPEG-LS编码器的压缩比达到2:1，与JPEG-LS软件压缩比一致。

1.4 论文组织

本论文共分为六个章节进行撰写，各章节的内容如下。

第一章是绪论。分析本文的选题背景与意义、国内外研究现状、研究内容与设计指标以及论文的组织情况。

第二章是相关图像压缩理论分析。首先分析图像压缩基本原理，然后分析 JPEG-LS 基本原理，然后分析三种常见的连续色调静止图像压缩标准及其算法，最后分析图像压缩的评价指标。

第三章是 JPEG-LS 算法分析与方案设计。分析 JPEG-LS 算法的整体结构及其各个模块，并确定 JPEG-LS 编码器的整体结构及其各个模块的设计方案。

第四章是 JPEG-LS 无损压缩算法的 FPGA 实现。对 JPEG-LS 编码器整体及其各个子模块进行详细设计，包括架构设计、时序控制状态机设计、逻辑设计等，并完成上下文建模模块、正常编码模块、游程编码模块以及输出模块等的 RTL 级设计。

第五章是仿真与验证。对本文设计的编码器进行功能仿真和 FPGA 板级验证，并对其进行结果分析。

第六章是总结与展望。总结本文的研究内容，并根据研究结果对下一步的工作进行展望。

万方数据

第二章 相关图像压缩理论分析

本章首先从信息的度量以及数据冗余与编码两个方面分析图像压缩原理，然后分析 JPEG-LS 算法各部分的基本原理，包括上下文建模、正常编码以及游程编码，然后分析三种常见的连续色调静止图像的压缩标准及算法，包括 JPEG、JPEG-LS 以及 JPEG2000，最后分析压缩图像质量以及图像压缩性能评价标准。

2.1 图像压缩原理

数据压缩的理论基础是信息论。从信息论的角度来看，压缩的目的是尽可能去掉信息中的冗余，即保留不确定的信息，而去除确定的信息。图像作为一个信源，包括大量的冗余信息，比如编码冗余、像素间冗余以及心理视觉冗余。

2.1.1 信息的度量

虽然一个信源发送出的符号通常是不确定的，但是可以根据其出现的概率来衡量它。概率大，出现机会多，不确定性小；反之，出现机会少，不确定性大。如果一个信源只发送一种符号，即发送内容是确定的，概率为 100%，那么接收方无法从接收信号中获得任何信息，信息量为 0。如果发送方和接收方约定好符号 1 代表二进制数 0，符号 2 代表二进制数 1，那么接收方可以通过所接收到的信源符号来获取一定的信息，信息量非 0。

1948 年，香农提出了“信息熵”的概念，用来定量衡量信息的大小[34]。所谓信息熵是指将原始信息中的冗余消除后的平均信息量。假设一个信源发出的信息含有 n 个信源符号X_i，其中 i 为 1 到 n 的整数。那么对于其中任何一个单独的信源X_i，其所表示的信息量如式（2.1）所示。

$$I(X_i)=\log_a(1/P(X_i))=-\log_a P(X_i) \qquad i=1,2,3...n \tag{2.1}$$

那么，这个含有 n 个信源符号的信源所发出的信息的总的信息量如式（2.2）所示。

$$H(X)=\sum_{i=1}^{n}P(X_i)I(X_i)=-\sum_{i=1}^{n}P(X_i)\log_a P(X_i) \tag{2.2}$$

其中，$P(X_i)$为信源符号出现的概率，$H(X)$为信源的信息熵。当 a=2 时，信息量的单位为比特，本文中采用比特作为信息量的单位。信息熵不仅定量地衡量了信息的大小，而且为信息编码提供了理论上的最优值：编码的平均码长的理论下限就是信息熵，即信息熵为数据压缩的极限。

2.1.2 图像数据冗余与编码

图像之所以能够被压缩，是因为图像像素之间并非完全独立，相邻像素之间有一定的相关性，通常包含大量的冗余信息。图像压缩的目的是尽可能地减少甚至消除冗余信息。

与图像数据相关的冗余有 3 种类型：编码冗余，像素间冗余和心理视觉冗余。编码冗余也叫信息熵冗余，是指所表示图像中像素强度的位数通常比实际需要的多。由于图像是以二进制编码形式进行存储和传输的，不同的编码方法对同一幅图像编码将产生不同的编码长度，相同的编码方法对不同图像编码也将产生不同的编码长度，因此产生了编码冗余，可以使用熵编码来减少冗余信息。

万方数据

像素间冗余是指图像的帧内像素或者帧间像素高度相关，信息不必要地重复，可以通过当前待编码像素的邻近像素来推估当前待编码像素的预测值，通常对待编码像素与其预测值的差值进行编码来减少冗余信息。心理视觉冗余是指人类视觉系统不可能一次对所有颜色都敏感，可以在编码过程中使用量化技术来减少冗余信息。这些冗余信息可以用特定概率分布的统计模型来表示，然后通过编码器来减少图像的数据量。因此，通常会对图像进行压缩处理以减小甚至消除冗余信息，从而降低图像对存储空间以及传输带宽的需求。

图像压缩方法按照压缩后的图像数据能否还原为源图像一般可分为两种方法，一种是基于有损压缩技术的方法，在压缩过程中有图像数据的丢失，信息熵减少，压缩后的图像数据不能通过解码器还原为源图像[35]，如对 DCT 或 FFT 处理后的数据进行编码，另一种是基于无损压缩技术的方法，在压缩过程中没有图像数据的丢失，信息熵保持不变，压缩后的图像数据能够通过解码器还原为源图像[36]，如对预测误差进行编码。常用的图像编码方法主要包括预测编码、统计编码以及变换编码三种[37-40]。

（1）预测编码

图像的邻近像素间具有高度相关性。预测编码就是利用图像像素在空间以及时间上的相关性进行编码。首先通过当前待编码像素的邻近像素来推知当前待编码像素的预测值，然后计算当前待编码像素与其预测值的差值，最后对预测差值进行编码。预测编码易于硬件实现，压缩性能良好。比如 JPEG-LS 就使用预测编码来对待编码像素与其预测值的差值进行编码，以提高编码的效率。

（2）统计编码

统计编码是一种根据信源概率分布的可变长码，用较短的码字表示出现概率较大的信源，用较长的码字表示出现概率较小的信源，使平均的码字长度尽可能短，以达到压缩的目的。比如 JPEG 就使用 Huffman 编码来对待编码像素与其预测值的差值进行编码，JPEG-LS 就使用游程长度编码来对图像水平方向灰度值连续相等的像素进行编码，以提高编码的效率。

（3）变换编码

变换编码是将通常在空间域描述的图像信号变换到另外的向量空间中进行描述，以消除图像像素间的相关性，然后再根据图像在此向量空间中系数的特点以及人的视觉特性进行编码，以达到压缩的目的。比如 JPEG2000 就使用变换编码对经过离散小波变换(Discrete Wavelet Transform, DWT)处理过的图像数据进行编码，以提高编码的效率。

如图 2-1 所示，为图像编码的一般过程。首先，对源图像像素重新建立一个数学模型，完成源图像像素表现形式到其他表现形式的映射，以降低源图像像素之间的相关性；其次，对建模后的模型参数进行量化，设法用更简洁的表现形式来表示建模后的模型参数；最后，对量化后的模型参数进行符号的分配，尽可能地减少冗余信息，以达到压缩源图像目的。

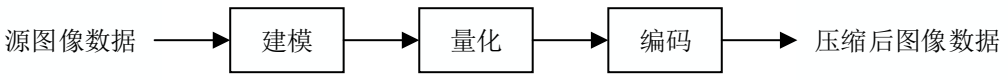


图 2-1 图像编码的一般过程

其中，量化过程是区别有损压缩与无损压缩的关键步骤，若量化过程中有数据丢失，信息熵减

万方数据

少，则为有损压缩，若量化过程中无数据丢失，信息熵保持不变，则为无损压缩。编码结束后，对压缩后数据进行解码，即可得到源图像或近源图像。

2.2 JPEG-LS 压缩原理

JPEG-LS 算法包括正常编码模式、游程编码模式。正常编码模式对非平坦区域的像素进行 Golomb 编码，游程编码模式对平坦区域的像素进行游程长度编码、对差异像素进行 Golomb 编码。

（1）上下文建模

由于图像的邻近像素间具有一定的相关性，可以通过邻近像素推估当前待编码像素。根据随机过程以及信息论可知，对马尔可夫图像信源序列x_n，x_{n-1}，…，x_1进行熵编码的最小信息量如式（2.3）所示。

$$H(x_m, x_{m-1}, ..., x_1) = \sum_{i=1}^{n} \sum_{x^{i-1}} p(x^{i-1}) \{ \sum_{x_i} p(x_i \mid x^{i-1}) \log_2 p(x_i \mid x^{i-1}) \} \tag{2.3}$$

其中，x^{i-1}=$x_{i-1}, x_{i-2}, \ldots, x_1$为符号$x_i$的上下文。也就是说，无损压缩的建模可以看作是一个归纳推理的过程，在对当前待编码像素编码时需要先扫描过去已编码的像素，通过给当前待编码像素分配一个条件概率 P，来推断出当前待编码的像素值，当前待编码像素的平均码长为$-\log 2^{p}$。

（2）预测误差

JPEG-LS 中正常编码模式通过上下文建模完成局部梯度值的计算，然后进行边缘检测，进而获得预测误差，然后通过上下文参数对预测误差进行校正，以减少系统偏差的影响，使预测误差更为准确。

在不考虑系统偏差的情况下，预测误差可以用如图 2-2 所示的一个近似对称的几何分布来表示，如式（2.4）所示。

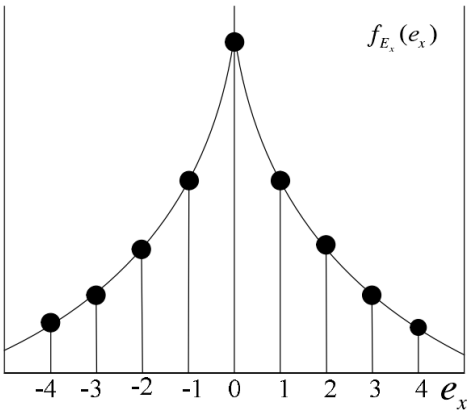


图 2-2 无偏预测误差分布

$$f_{E_x}(e_x) = \frac{1-\rho}{1-\rho} \rho^{|e_x|} \tag{2.4}$$

JPEG-LS 中预测误差的计算如式（2.5）所示。

$$Errval = sign(x - P_x) - \beta_x \tag{2.5}$$

其中，*Errval* 为预测误差；*sign* 为取值为 1 或-1 的校正符号；*x* 为当前待编码像素；P_x为当前待

万方数据

编码像素的预测值；β_x为偏差的修正值，不仅可以自适应补偿由非预测因子带来的偏差，而且还可以产生一个可控偏差，进而使得对预测误差的编码更有效。当$\beta_x \in \mathbb{Z}$时，通过自适应调整，使得预测误差的数学期望满足式（2.6）。

$$-1 \le E[E_x] \le 0 \tag{2.6}$$

（3）Golomb-Rice 编码

JPEG-LS 中使用 Golomb-Rice 编码对预测误差进行编码。Golomb 编码是 S.W.Golomb 于 1966 年提出的，依赖于预测误差的分布满足以 0 为中心、两边对称、指数衰减。因为上述所得预测误差期望为[-1,0]，所以将预测误差映射为非负数。通过式（2.7）对预测误差e_x进行映射。

$$e_x = \begin{cases} 2e_x & e_x \ge 0 \\ -2e_x - 1 & e_x < 0 \end{cases} \tag{2.7}$$

虽然经过映射后的e_x不能完全服从双边对称的几何分布，但是在舍弃其所包含的一个或多个最低有效位后，可以使其满足近似的几何分布。因此，对于 k≥1 所对应的 Golomb 参数，编码是被证明是有效可行的。

满足几何分布的预测误差e_x的概率函数由式（2.8）所示。

$$f_{E_x}(e_x) = (1-\rho)\rho^{e_x},\ 0 < \rho < 1 \tag{2.8}$$

当$\rho \le (1/2)$时，预测误差的一元码为最佳前缀码，因此需要将$\rho > (1/2)$转换至$\rho \le (1/2)$的几何分布。可以将$\rho > (1/2)$下的e_x表示为$e_x = e_{xq}m + e_{xr}$，其中e_{xq}是e_x除以整数 m 的商，e_{xr}是其余数，分别由式（2.9）、（2.10）所示。

$$e_{xq} = \left\lfloor \frac{e_x}{m} \right\rfloor \tag{2.9}$$

$$e_{xr} = e_x \bmod m \tag{2.10}$$

若设E_{xq}为随机变量，则其概率函数如式（2.11）所示。

$$f_{E_{xq}}(E_{xq}) = \rho m e_{xq}(1-\rho)\sum_{i=0}^{m-1} f_{E_x}(i) \tag{2.11}$$

显然，E_{xq}服从参数为ρ^m的几何分布。e_{xr}可进行优化可变长编码，通过将e_{xq}的码字和e_{xr}的码字连接起来，即可生成该几何分布的信源的可变长编码。

（4）游程长度编码

游程长度编码属于统计编码的一种，是无损压缩编码。对于灰度图像的游程长度编码来说，如果水平方向上有 F 个连续相等的像素 G，那么在对其进行游程长度进行编码后，用(G,F)即可表示这些连续相等的像素，可大量减少数据量，达到压缩的目的。这些连续相等的像素即为游程。显然，游程长度越长，游程长度编码的效率越高，因此游程长度编码特别适用于灰度等级少、灰度值变化小的二值图像。

JPEG-LS 中采用游程长度编码对游程长度进行编码。通过将所扫描的游程长度和入口向量表所指向的游程长度进行对比，按照游程长度来获得编码码字，进一步提高了编码的效率。

万方数据

2.3 常见静止图像压缩标准与算法

为了更好地进行图像信息的交流，国际标准化组织 ISO 以及国际电联电信标准化部门 ITU-T 的图像专家组开始进行图像压缩标准的制定，以提供统一、高效的压缩算法以及统一的压缩数据格式。针对连续色调静止图像相继提出了 JPEG、JPEG-LS、JPEG2000 等标准。

（1）JPEG

JPEG 标准是联合图像专家组于 1991 年发布的第一个连续色调静止图像压缩的国际标准[41]。JPEG 是分别基于预测及变换的无损及有损压缩算法，采用线性预测、离散余弦变换以及 Huffman 编码算法实现对图像的无损压缩以及有损压缩。如图 2-3 所示，为 JPEG 无损压缩的结构框图。JPEG 编码器首先对源图像数据进行线性预测，得到预测值，然后得到预测误差，最后对预测误差进行熵编码，编码结束后输出编码码流。

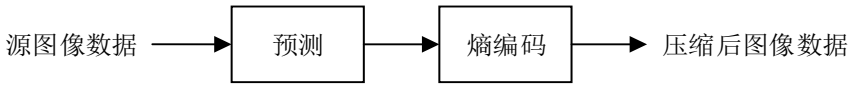


图 2-3 JPEG 无损压缩的结构框图

其中，预测模块通过当前待编码像素的 3 个邻近像素完成对当前待编码像素预测值的推估，而 JPEG-LS 通过当前待编码像素的 4 个邻近像素完成对当前待编码像素预测值的推估。相比之下，JPEG-LS 得到的预测值更接近于源图像像素值。熵编码模块可以通过 Huffman 编码完成对预测误差的编码，而 JPEG-LS 通过 Golomb 编码完成对预测误差的编码。相比之下，Golomb 编码的硬件实现更简单。虽然 JPEG 支持无损压缩，但是对于中等复杂度的彩色图像的压缩比不到 2:1，且压缩性能一般，因此被 JPEG-LS 代替[39]。

（2）JPEG2000

JPEG2000 标准是联合图像专家组于 2000 年 12 月发布的一个新的静止图像压缩的国际标准，通常被认为是取代 JPEG 标准的新一代图像压缩标准[42-43]。JPEG2000 是基于变换的无损及有损压缩算法，采用离散小波变换以及嵌入式多分辨率编码（Embedded Block Coding with Optimized Truncation, EBCOT）算法实现对图像的无损压缩以及有损压缩。如图 2-4 所示，为 JPEG2000 无损压缩的结构框图。JPEG2000 编码器首先对源图像数据进行预处理，然后对预处理后的图像数据进行离散小波变换，将图像时域信号变换到频域信号，最后对其进行嵌入式块编码，编码结束后输出编码码流。

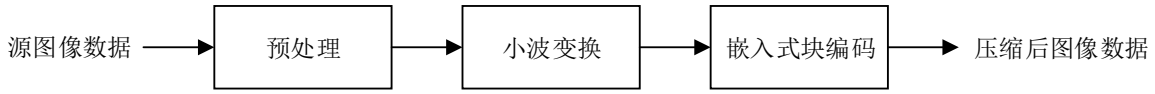


图 2-4 JPEG2000 无损压缩的结构框图

其中，预处理模块包括区域划分、直流偏移以及分量变换，嵌入式块编码包括算术编码以及位平面编码。与 JPEG-LS 相比，JPEG2000 算法复杂度高，不利于硬件实现，而且由于其编码核心部分的众多演进算法存在专利风险，开发出免授权的编码器难度很高。

（3）JPEG-LS

JPEG-LS 标准是由国际标准化组织 ISO 于 1999 年 12 月发布的 ISO-14495-1 标准和国际电信联

万方数据

盟 ITU 于 1998 年 6 月发布的 ITU-T.87 标准。JPEG-LS 是基于预测的无损及近无损压缩算法，采用固定预测、Golomb 编码以及游程长度编码实现对图像的无损压缩以及近无损压缩。如图 2-5 所示，为 JPEG-LS 无损压缩的结构框图。JPEG-LS 编码器首先对源图像数据进行上下文建模，然后根据待编码像素的局部梯度值进行编码模式选择，若局部梯度值全部为零，则对其进行游程编码，否则对其进行正常编码，编码结束后输出编码码流。

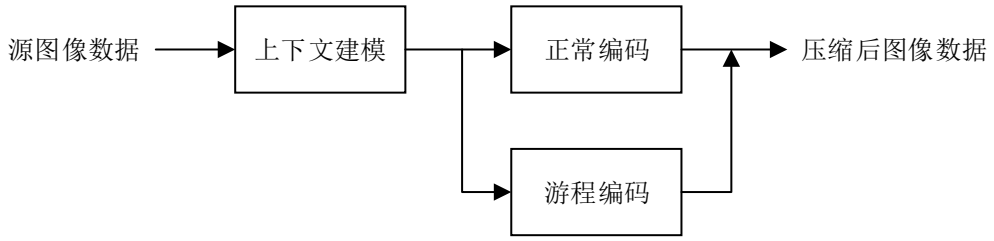


图 2-5 JPEG-LS 无损压缩的结构框图

其中，正常编码模块采用 Golomb 编码对预测误差进行编码，游程编码模块采用游程长度编码对游程长度进行编码，采用 Golomb 编码对差异像素的预测误差进行编码。

JPEG-LS 标准并不是对 JPEG 标准的修正与补充，而是一种采用全新架构、全新算法的全新标准[34]。JPEG 标准下的无损压缩采用的是线性预测以及 Huffman 编码，有损压缩采用的是离散余弦变换、量化以及 Huffman 编码；而 JPEG-LS 标准无论无损压缩还是近无损压缩均采用固定预测、Golomb 编码以及游程编码，与无损压缩不同的仅仅在于近无损压缩会对预测误差进行量化处理。与 JPEG 算法、JPEG2000 算法相比，JPEG-LS 算法没有诸如离散余弦变换、离散小波变换等复杂的数据变换过程，只需实现数据的加法、减法、移位以及其他类似的、简单的数据处理过程。如表 2.1 所示，为 JPEG、JPEG2000 以及 JPEG-LS 三种压缩算法的比较。

表 2.1 三种压缩算法的比较

算法	获得难度	低复杂性	无损压缩性能
JPEG	-	++++	+
JPEG2000	++	++	+++
JPEG-LS	-	+++++	++++

由表 2.1 可知，与 JPEG、JPEG2000 等算法相比，JPEG-LS 算法具有可获得难度低，复杂度低，并且在同等复杂度的情况下具有最好的压缩性能[7][11][44]。同时，由于 JPEG-LS 无损压缩算法只需实现数据的加法、减法、移位以及其他类似的、简单的数据处理过程，硬件实现复杂度低，且占用硬件资源少。因此，本文选择研究 JPEG-LS 无损压缩算法的 FPGA 实现。

2.4 图像压缩评价标准

2.4.1 压缩图像质量评价标准

用于压缩图像质量的评价标准主要包括主观评价与客观评价两种。

（1）主观评价

主观评价通过挑选一定数量的观察者对压缩图像的质量进行评价，评价结果可以用很差、较差、较好以及很好等词语来描述。由于主观评价没有统一的衡量指标，主要取决于人的心理视觉系统，

万方数据

其感知的结果受许多因素的影响而产生变化，比如图像的空间频率、亮度、邻近区域等，因此主观评价主要用来衡量图像的有损压缩。

（2）客观评价

客观评价主要采用峰值信噪比(Peak Signal to Noise Ratio, PSNR)来衡量[45]。*PSNR* 通过均方差来定义，均方差如式（2.12）所示。

$$\delta_e^2 = \frac{1}{MN}\sum_{i=1}\sum_{j=1}[S(i,j)-S^{'}(i,j)]^2 \tag{2.12}$$

其中，*M* 和 *N* 分别为图像垂直方向和水平方向上像素的个数，也就是图像的高和宽；$S(i,j)$和$S^{'}(i,j)$分别为点(i,j)上源图像的像素值和解压后重建的像素值，*i* 和 *j* 分别表示像素所处图像的行和列。

通过均方差来定义峰值信噪比，峰值信噪比如式（2.13）所示。

$$PSNR=10\log\frac{S_{p-p}^2}{\delta_e^2}(dB) \tag{2.13}$$

其中，δ_e^2为源图像的均方差，$S_{p\text{-}p}$为源图像的峰峰值。*PSNR* 越高，还原后图像的失真越低，越和源图像接近。理论上，因为对源图像进行的是无损压缩，解压后重建的像素值等于源图像的像素值，所以 *PSNR* 无穷大。但是由于人的心理视觉系统对误差的敏感度并不是绝对的，因此 *PSNR* 并不能很好地反映人的主观感受。由于本文研究的是 JPEG-LS 的无损压缩，因此在对本文所设计的 JPEG-LS 编码器进行功能测试时不使用该评价标准。

2.4.2 图像压缩性能评价标准

图像压缩性能的评价标准主要包括压缩比和编码效率。

（1）压缩比

压缩比是指源图像的信息量与压缩后图像的信息量之比。在不减少图像信息熵的基础上，压缩比越大越好。图像的压缩比如式（2.14）所示。

$$c=\frac{n_1}{n_2} \tag{2.14}$$

其中，*c* 为图像的压缩比，n_1为源图像的信息量，n_2为压缩后图像的信息量。

（2）编码效率

编码效率是指源图像的信息熵与压缩后图像数据的平均码长之比。在编码过程中图像信息不丢失的基础上，平均码长越低，编码效率越高。图像的编码效率如式（2.15）所示。

$$\begin{aligned}\eta &= \frac{H(X)}{L}\\ L &= \sum_{i=1}^{n}P(X_i)l(X_i)\end{aligned} \tag{2.15}$$

其中，η为编码效率，*L* 为压缩后图像数据的平均码长，$l(X_i)$为信源X_i码字的长度。编码后图像的冗余度如式（2.16）所示。

$$R=1-\eta \tag{2.16}$$

其中，*R* 为编码后图像的冗余度。由于本文研究的是 JPEG-LS 的无损压缩，其所采用编码算法是 Golomb 编码以及游程长度编码，其平均码长接近于信息熵，编码效率接近于 1，因此在对本文所

万方数据

设计的 JPEG-LS 编码器进行性能测试时不使用该评价标准。

2.5 本章小结

本章主要阐述图像压缩的理论基础，首先分析了信息熵以及图像数据冗余与编码，阐述信息熵在图像压缩中的意义、图像压缩的原因以及压缩的方法；然后研究了 JPEG-LS 中上下文建模、正常编码以及游程长度编码等的基本原理，为 JPEG-LS 压缩算法的实现提供了理论基础；然后分析了三种常见的连续色调静止图像压缩标准与算法，分析结果表明 JPEG-LS 相比 JPEG 和 JPEG2000 易于硬件实现，占用硬件资源少，并且在同等复杂度的情况下能够获得最好的压缩性能；最后从压缩图像质量的评价标准以及图像压缩性能的评价标准两个方面分析了图像压缩的评价标准。

万方数据

第三章 JPEG-LS 算法分析与方案设计

本章分析 JPEG-LS 算法的整体框架及其各个模块，并确定 JPEG-LS 编码器硬件设计的整体结构以及各个模块的设计方案，包括上下文建模模块、正常编码模块、游程编码模块以及码流拼接模块。针对传统的上下文因果模板构建过程影响编码器吞吐率的问题，提出一种改进的因果模板构建的设计方案；针对传统的 Golomb 编码参数 k 计算过程影响编码器时钟频率的问题，提出采用二分查找法对其计算过程进行优化的设计方案。

3.1 JPEG-LS 算法整体框架

JPEG-LS 算法的核心是惠普实验室开发的 LOCO-I 算法，包括正常编码模式及游程编码模式两种模式，主要由上下文建模、正常编码、游程编码等部分组成，提供无损压缩以及近无损压缩两种功能。当无损压缩与近无损压缩的阈值 *NAER* 等于 0 时，JPEG-LS 进行的是无损压缩，否则进行的是近无损压缩。由于本文只研究 JPEG-LS 算法的无损压缩部分，因此 *NEAR* 取值为 0。如图 3-1 所示，为 JPEG-LS 算法的整体框架。

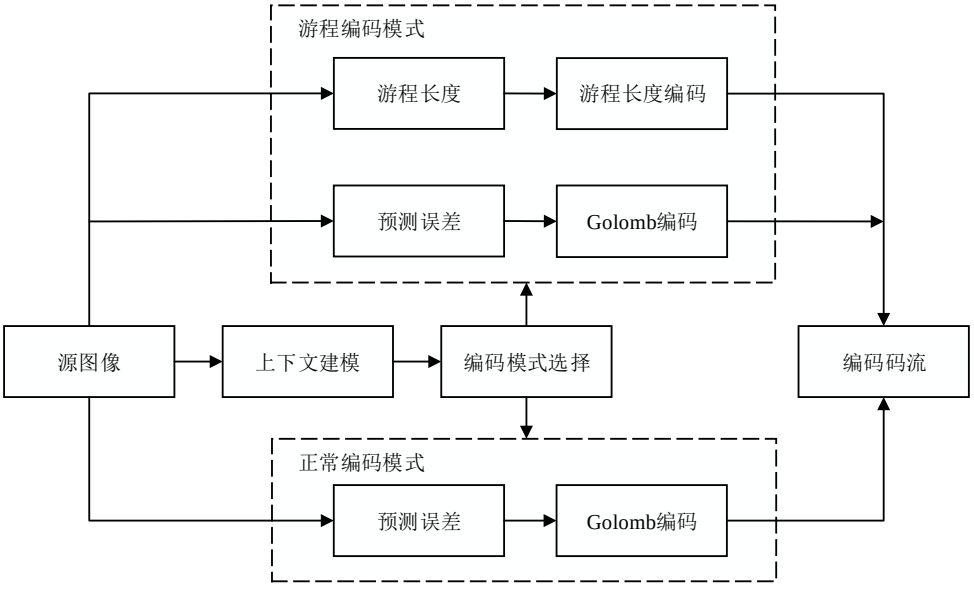


图 3-1 JPEG-LS 算法整体框架

JPEG-LS 编码过程如下：

（1）按照光栅扫描顺序读入源图像数据，完成对当前待编码像素因果模板的构建；

（2）计算当前待编码像素的局部梯度值，并根据局部梯度值进行编码模式的选择，若局部梯度值的绝对值全部小于等于 *NEAR*，则进入步骤（3），否则进入步骤（4）；

（3）当前待编码像素进入游程编码模式，得到游程编码码流，进入步骤（6）；

（4）对局部梯度值进行量化、符号校正、合并，得到上下文参数 *Q*；

（5）当前待编码像素进入正常编码模式，得到正常编码码流，进入步骤（6）；

（6）对上次拼接剩余码流与本次编码码流进行拼接、输出，然后进入步骤（1），直到整幅图像编码结束。

万方数据

根据 JPEG-LS 编码算法的整体架构，提出一个由上下文建模模块、正常编码模块、游程编码模块以及输出模块组成的 JPEG-LS 编码器整体结构，如图 3-2 所示。整个编码器的编码过程通过状态机来控制。

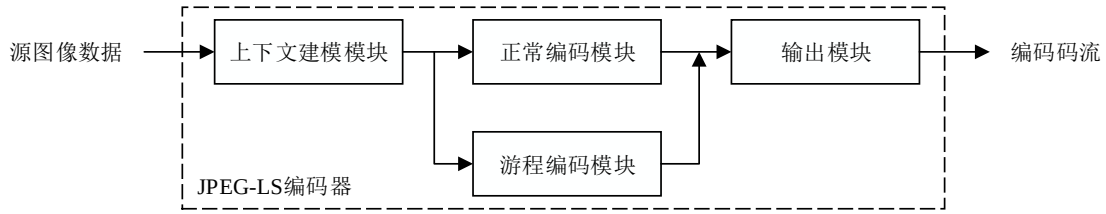


图 3-2 JPEG-LS 编码器的整体结构框图

其中，上下文建模模块的功能是完成上下文参数的计算以及编码模式的选择，正常编码模块的功能是对进入正常编码模式的待编码像素进行编码，游程编码模块的功能是对进入游程编码模式的待编码像素进行编码，输出模块的功能是对上次拼接剩余码流与正常编码模块的编码码流以及游程编码模块的编码码流进行拼接、输出。

3.2 上下文建模

所谓上下文建模就是利用当前待编码像素的邻近像素与当前待编码像素之间的相关性对其进行建模[46]。如图 3-3 所示，为上下文建模模块的结构框图。

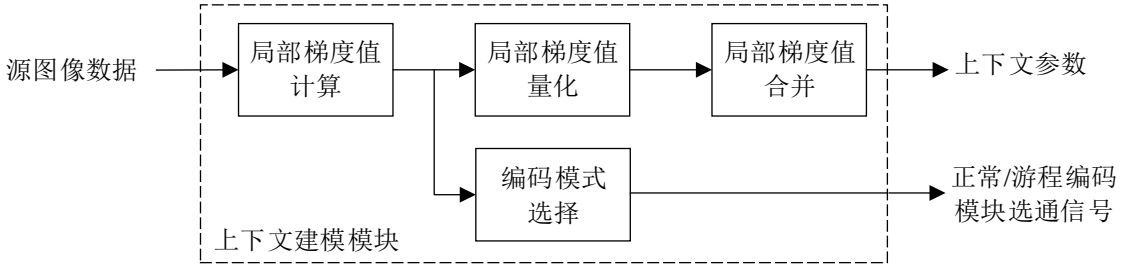


图 3-3 上下文建模模块的结构框图

其中，局部梯度值计算模块的功能是读入源图像数据，并完成当前待编码像素因果模板的构建，计算因果模板中邻近位置像素的差值，得到局部梯度值；局部梯度值量化模块的功能是将局部梯度值量化为区间[-4,4]内的整数；局部梯度值合并模块的功能是对量化后的局部梯度值进行符号的校正，然后将校正后的局部梯度值映射为区间[0,364]内的整数，得到上下文参数；编码模式选择模块的功能是根据局部梯度值计算模块的结果进行编码模式的选择，若结果全为 0，则选通游程编码模块，否则选通正常编码模块。

因果模板的构建是进行局部梯度值计算的条件。通常需要读 5 次源图像存储器才能完成当前待编码像素因果模板的构建，通过利用无损压缩情况下重建值等于像素值这一特性，本文提出一种改进的当前待编码像素因果模板构建过程，通过利用上一个已编码像素的因果模板，只需读 2 次源图像存储器即可完成当前待编码像素因果模板的构建，有效提高了编码器的吞吐率。通常在当前待编码像素编码结束后才开始下一个待编码像素因果模板的构建，本文提出一种改进的下一个待编码像素因果模板的构建过程，通过在当前待编码像素上下文参数更新过程中完成下一个像素因果模板中待编码像素 *Ix* 以及重建值 *Rd* 的读取，进一步提高了编码器的吞吐率。改进的因果模板构建过程的

万方数据

具体实现将在第四章中给出。

如图 3-4 所示，为 JPEG-LS 算法中上下文建模的流程图。首先按照从左到右、从上到下的顺序扫描待编码像及其 4 个已编码的邻近像素，完成因果模板的构建；然后完成局部梯度值的计算；然后根据局部梯度值完成编码模式的选择，若局部梯度值不全为 0，则对局部梯度值进行量化，否则结束本次上下文建模，进入游程编码模式；然后计算校正符号；最后对局部梯度值进行符号校正、合并，合并完成后结束本次上下文建模，进入正常编码模式。

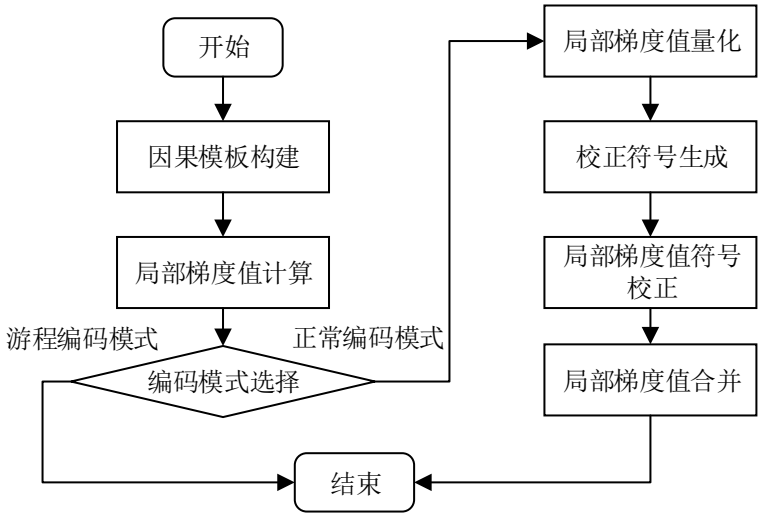


图 3-4 上下文建模流程图

（1）局部梯度值计算

因为图像像素之间存在相关性，所以会利用当前待编码像素的 4 个邻近像素来完成当前待编码像素因果模板的构建。如图 3-5 所示，为当前待编码像素的因果模板。其中，x 为当前待编码像素所在位置，对应像素值 *Ix*；a、b、c、d 为当前待编码像素的 4 个相邻位置，分别对应 4 个重建值 *Ra*、*Rb*、*Rc*、*Rd*。

c	b	d
a	x	

图 3-5 当前待编码像素因果模板

如图 3-6 所示，为处于图像边界时当前待编码像素的因果模板。若当前待编码像素处于图像的第一行，则位置 b、c、d 上的重建值*Rb*=*Rc*=*Rd*=0；若当前待编码像素处于图像的第一列，则位置 a 上所对应的重建值*Ra*=*Rb*；若当前待编码像素处于图像的最后一列，则位置 d 上所对应的重建值*Rd*=*Rb*。在无损压缩情况下，重建值就等于该位置上所对应的源图像的像素值。

万方数据

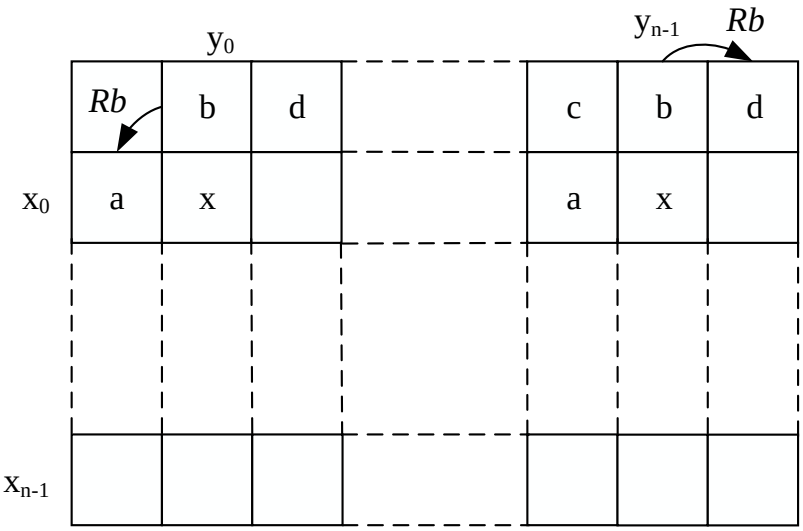


图 3-6 处于图像边界时当前待编码像素的因果模板

其中，x_0、y_0、x_{n-1}、y_{n-1}分别指图像的第一行、图像的第一列、图像的最后一行以及图像的最后一列。

在当前待编码像素的因果模板构建完成后，开始对当前待编码像素的局部梯度值进行计算。局部梯度值的计算如式（3.1）所示。局部梯度值 $D1$、$D2$、$D3$ 表示的是当前待编码像素邻近像素的活动水平，比如平滑性、边界性，被用来估计当前待编码像素预测误差的统计行为。当 $D1$、$D2$、$D3$ 的绝对值小于等于 *NEAR* 时，当前待编码像素与其邻近像素构成平坦区域；当 $D1$、$D2$、$D3$ 的绝对值大于 *NEAR* 时，当前待编码像素与其邻近像素构成非平坦区域。在无损压缩情况下，平坦区域是指灰度值相等的区域，非平坦区域是指灰度值不相等的区域。

$$\begin{aligned} D1 &= Rd - Rb \\ D2 &= Rb - Rc \\ D3 &= Rc - Ra \end{aligned} \tag{3.1}$$

在局部梯度值 $D1$、$D2$、$D3$ 计算完成后进行编码模式的选择。若局部梯度值的绝对值全部小于等于 *NEAR* 时，则当前待编码像素进入游程编码模式，否则当前待编码像素进入正常编码模式。

（2）局部梯度值量化

如果在同一个上下文中对少量的待编码像素进行编码，通常就无法获得足够多的上下文统计信息；如果上下文数量过多，编码处理过程中存储容量的需求就会随之增大。为了避免产生大量的上下文，需要对局部梯度值作进一步的处理。因此，具有相似特征的局部梯度值被量化、合并。因为图像水平方向的像素与垂直方向的像素对局部梯度值量化的影响是相同的，所以 $D1$、$D2$、$D3$ 对局部梯度值量化过程的影响是相同的，并且 $D1$、$D2$、$D3$ 都被量化到少量近似相等的区域。JPEG-LS 算法中局部梯度值的量化如式（3.2）所示，局部梯度值被量化为 9 个整数，有效减少了上下文的数量，便于运算与统计。如果量化后上下文参数Qi过多，那么便失去了量化的意义；如果量化后上下文参数Qi过少，那么会使一个Qi对应大量的像素，无法获得足够多的上下文统计信息。

万方数据

$$Qi=\begin{cases}-4 & Di <= -T3\\ -3 & Di <= -T2\\ -2 & Di <= -T1\\ -1 & Di < -NEAR\\ 0 & Di <= NEAR \quad i=1,2,3\\ 1 & Di < T3\\ 2 & Di < T2\\ 3 & Di < T1\\ 4 & otherwise\end{cases} \tag{3.2}$$

其中，$T1$、$T2$、$T3$ 为局部梯度值量化的阈值。当对一幅比特深度为 8 的灰度图像进行无损压缩时，$T1$、$T2$、$T3$ 的值分别为 3、7、21。

（3）局部梯度值合并

因为局部梯度取 V 值的概率与取-V 值的概率相同，所以局部梯度值的量化器是关于 0 差分对称的。因此，通过合并量化后符号相反的局部梯度值($Q1$,$Q2$,$Q3$)，可以进一步减少上下文的数量。当满足以下任意条件：（1）$Q1<0$；（2）$Q1=0$， $Q2<0$；（3）$Q1=Q2=0$，$Q3<0$，将校正符号 *SIGN* 赋值为-1，对矢量($Q1$,$Q2$,$Q3$)进行符号变换，变换后的矢量为*(-Q1,-Q2,-Q3)*，否则将 *SIGN* 赋值为 1，矢量($Q1$,$Q2$,$Q3$)保持不变。局部梯度值符号变换后就将最多可能的9*9*9=729个矢量减少为365个矢量。

为了便于上下文参数的运算与统计，在矢量($Q1$,$Q2$,$Q3$)符号变换完成后，将其以一种一对一的方式映射为区间[0,364]内的整数 Q，用以表示当前待编码像素的上下文参数。JPEG-LS 标准中并未规定将矢量($Q1$,$Q2$,$Q3$)映射为 Q 的函数，本文采用的合并函数如式（3.3）所示。通过该式，不仅可以将矢量($Q1$,$Q2$,$Q3$)唯一的映射为整数 Q，保证了编码码字的独特可译性，而且使得 Q 可能取到的最大值不超过 364，最小值不低于 0。

$$Q=(Q1*9+Q2)*9+Q3 \tag{3.3}$$

3.3 正常编码模式

如图 3-7 所示，为正常编码模块的结构框图。采用并行处理方式，对正常编码模块结构进行优化，即在预测误差模减完成后，就开始对上下文参数进行更新。

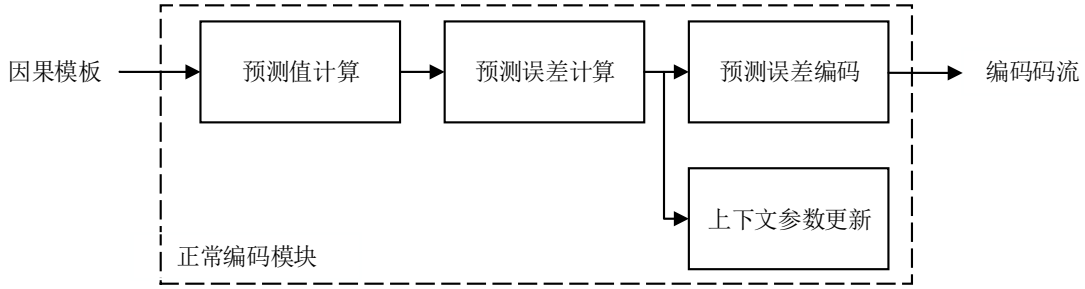


图 3-7 正常编码模块的结构框图

其中，预测值计算模块的功能是根据因果模板完成当前待编码像素预测值的计算，根据校正符号完成预测值的符号校正，并将校正后的预测值映射到区间[0,*MAXVAL*]内；预测误差计算模块的功能是完成预测值与当前待编码像素差值的计算，并对得到的预测误差进行符号校正以及模减，将预测误差映射到区间(-*RANGE*/2,⌈*RANGE*/2⌉-1]内；上下文参数更新模块的功能是对变量 $A[Q]$、$B[Q]$、

万方数据

C[*Q*]以及 *N*[*Q*]进行更新；预测误差编码模块的功能是对映射后的预测误差进行 Golomb 编码。

如图 3-8 所示，为 JPEG-LS 算法中正常编码模式的流程图。在当前待编码像素进入正常编码模式后，首先对当前待编码像素进行边缘检测，得到当前待编码像素的预测值，然后对预测值进行校正、规整，然后计算当前待编码像素的预测误差，并对预测误差进行校正、模减，然后计算 Golomb 编码参数，然后对预测误差进行映射，然后对映射后的预测误差进行 Golomb 编码，最后进行上下文参数的更新。

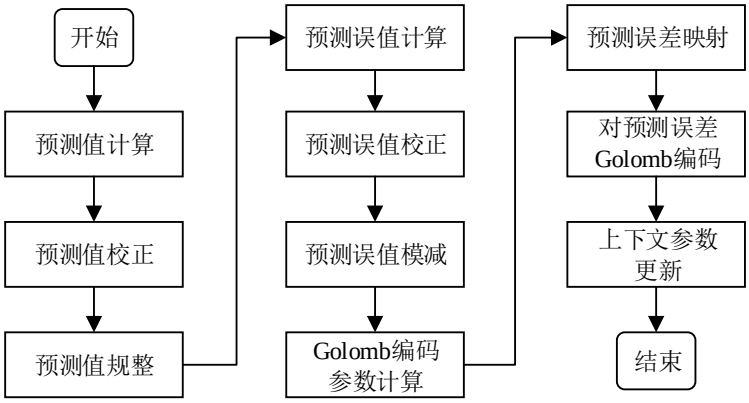


图 3-8 JPEG-LS 算法正常编码模式的流程图

（1）预测值计算

为了补偿局部梯度值量化过程中信息的丢失，在上下文参数确定之后，首先要对当前待编码像素进行边缘检测，即通过当前待编码像素 4 个邻近位置上的重建值来得到当前待编码像素的预测值。为了得到当前待编码像素的预测值，需要检测当前待编码像素的垂直边界或水平边界。待编码像素的边缘检测如式（3.4）所示。如果*Rc*≥max*(Ra,Rb)*，并且*Ra*≥*Rb*，那么待编码像素的左侧存在垂直边界，此时预测值*Px*=*Rb*；如果*Rc*≥max*(Ra,Rb)*，并且*Ra*<*Rb*，那么待编码像素的上方存在水平边界，此时预测值*Px*=*Ra*；如果*Rc*≤min*(Ra,Rb)*，并且*Ra*≥*Rb*，那么待编码像素的上方存在水平边界，此时预测值*Px*=*Ra*；如果*Rc*≤min*(Ra,Rb)*，并且*Ra*<*Rb*，那么待编码像素的左侧存在垂直边界，此时预测值*Px*=*Rb*；如果没有检测到垂直边界或水平边界，那么预测值*Px*=*Ra*+*Rb*-*Rc*。

$$Px=\begin{cases}\min(Ra,Rb) & Rc\geq\max(Ra,Rb)\\ \max(Ra,Rb) & Rc\leq\min(Ra,Rb)\\ Ra+Rb-Rc & otherwise\end{cases} \tag{3.4}$$

由于在基于上下文的模型中，依赖于上下文的预测误差的系统偏差并不少见，因此为了减小系统偏差的影响，当边缘检测完成后，需要对预测值进行校正。通过*C*[*Q*]以及 *SIGN* 对预测值 *Px* 进行校正，若*SIGN*=1，则*Px*=*Px*+*C*[*Q*]，否则*Px*=*Px*-*C*[*Q*]，使 *Px* 满足对称中心在-1 和 0 之间的双边几何分布。当预测值的校正完成后，对校正后的预测值 *Px* 进行规整，若*Px*>*MAXVAL*，则*Px*=*MAXVAL*，否则若*Px*<0，则*Px*=0，否则 *Px* 保持不变，将 *Px* 映射到区间[0,*MAXVAL*]内。其中，*C*[*Q*]为上下文参数 *Q* 所对应的预测误差的校正值，*MAXVAL* 为所扫描的源图像最大的像素值，*MAXVAL* 的计算如式（3.5）所示。

$$MAXVAL=2^{p}-1 \tag{3.5}$$

其中，*p* 为源图像的比特深度，本文中源图像的比特深度为 8。对于比特深度为 8 的灰度图像来

万方数据

说，*MAXVAL*=127。

（2）预测误差计算

当预测值校正完成后，计算预测误差 *Errval*。预测误差的计算如式（3.6）所示。预测误差计算完成后，对预测误差进行符号校正。若*SIGN*=-1，则*Errval=-Errval*，否则 *Errval* 保持不变。在 JPEG-LS 近无损压缩情况下，需要对预测误差进行量化处理以及当前待编码像素重建值 *Rx* 的计算。因为在预测误差的量化过程中有信息的丢失，信息熵减少，所以这也是无损压缩和近无损压缩的区别所在。由于本文研究的是无损压缩，因此不需要对预测误差进行量化处理，也就不对该量化过程做进一步分析。当预测误差符号校正完成后，对预测误差进行模减，若*Errval*<(-*RANGE*/2)，则*Errval=Errval+RANGE*，否则若*Errval*≥(1+*RANGE*/2)，则*Errval=Errval-RANGE*，否则 *Errval* 保持不变，将其映射到区间(-*RANGE*/2,⌈*RANGE*/2⌉-1]内。其中，*RANGE* 为预测误差的范围，其计算如式（3.7）所示。对于比特深度为 8 的灰度图像来说，*RANGE*=128。

$$Errval = Ix - Px \tag{3.6}$$

$$RANGE = MAXVAL + 1 \tag{3.7}$$

（3）预测误差编码

预测误差编码模块主要完成 Golomb 编码参数 k 的计算、预测误差的映射以及对映射后预测误差的 Golomb 编码。如图 3-9 所示，为预测误差编码模块的结构框图。

图 3-9 预测误差编码模块的结构框图

其中，Golomb 编码参数 k 计算模块的功能是通过变量 *A*[*Q*]、*N*[*Q*]完成编码参数 k 的计算，使得 Golomb 编码能够用最短的码字来表示映射后的预测误差；预测误差映射模块的功能是将满足双边几何分布的预测误差映射为满足单边几何分布的预测误差，进一步提高 Golomb 编码的效率；Golomb 编码模块的功能是对映射后的预测误差进行编码。JPEG-LS 算法中预测误差编码的过程如下。

首先，通过比较(*N*[*Q*]<<k)和 *A*[*Q*]的大小得到 Golomb 编码参数 k，使得条件(*N*[*Q*]<<k)<*A*[*Q*]不成立时的最小 k 值即为 Golomb 编码参数 k。其中，*A*[*Q*]为上下文参数 *Q* 所对应的预测误差绝对值的和，*N*[*Q*]为上下文参数 *Q* 出现的次数。对于一幅比特深度为 8 的灰度图像来说，*N*[*Q*]的最小值为 1，*A*[*Q*]的最大值为 128，所以 k 的最大值为 7，最小值为 0。由于 k 是根据变量 *A*[*Q*]和 *N*[*Q*]自适应调整的，因此 Golomb 编码能够用最短的码字来对映射后的预测误差进行编码，使得编码码字的平均长度最短。

常用的采用顺序比较的方法来计算 Golomb 编码参数 k 的值，最多需要比较 7 次，则硬件实现时最多有 7 级比较器延迟以及 7 级移位寄存器延迟，本文提出使用二分查找法来计算 k 的值，最多需要比较 3 次，则硬件实现时最多有 3 级比较器延迟以及 3 级移位寄存器延迟，能够有效提高所设计编码器的时钟频率。使用二分查找法来完成 Golomb 编码参数 k 计算的具体实现将在第四章中给出。

万方数据

然后，通过 *NEAR*、k、*B*[*Q*]、*N*[*Q*]将预测误差映射为一个非负整数 MErrval。由于校正后的预测值满足双边几何分布，所以预测误差仍然满足双边几何分布。为了进一步提高 Golomb 编码效率，将预测误差映射为非负整数，映射后的预测误差满足单边几何分布。之所以没有像 JPEG 算法中采用 Huffman 编码，是因为 Golomb 编码比 Huffman 编码更适用于硬件实现；之所以使预测误差满足几何分布，是因为 Golomb 编码对满足几何分布的预测误差有等同于 Huffman 编码的压缩效果[47]。

最后，对映射后的预测误差 MErrval 进行 Golomb 编码。在编码之前要对 MErrval 进行右移 k 位操作，得到 MErrval 除以 k 的商 *q*。如果*q*<*LIMIT-qbpp*-1，那么先对 *q* 进行一元编码，即先将 *q* 个 1 比特 0 附加到编码码流中，然后再将 1 比特 1 附加到编码码流中，最后将 MErrval 的低 k 位以二进制形式附加到编码码流中，高位在前，低位在后。如果*q*≥*LIMIT-qbpp*-1，那么先对*LIMIT-qbpp*-1进行一元编码，即先将*LIMIT-qbpp*-1个 1 比特 0 附加到编码码流中，然后再将 1 比特 1 附加到编码码流中，最后将MErrval-1的低 *qbpp* 位以二进制形式附加到编码码流中，高位在前，低位在后。其中，*LIMIT* 为 Golomb 编码码字的最大长度，*qbpp* 为表示映射预测误差所需要的比特数。*LIMIT* 和 *qbpp* 计算如式（3.8）所示。

$$\begin{aligned} LIMIT &= 2*(bpp+\max(8,bpp)) \\ bpp &= \max(2,\lceil \log(MAXVAL+1)\rceil) \\ qbpp &= \lceil \log RANGE\rceil \end{aligned} \tag{3.8}$$

其中，*bpp* 为表示 *MAXVAL* 所需要的比特数。对于比特深度为 8 的灰度图像来说，*bpp*=8，*qbpp*=8，*LIMIT*=32。

（4）上下文参数更新

因为 Golomb-Rice 编码非常依赖于预测误差的分布，所以应使预测误差满足以 0 为对称中心、呈指数衰减的双边几何分布。因为依赖于上下文模型的系统偏差并不少见，同时系统偏差的存在会严重恶化 Golomb-Rice 编码器的压缩性能，所以需要对变量 *A*[*Q*]、*B*[*Q*]、*C*[*Q*]、*N*[*Q*]进行更新以减少系统偏差的影响。

JPEG-LS 算法中对当前待编码像素进行编码的最后一步是变量 *A*[*Q*]、*B*[*Q*]、*C*[*Q*]、*N*[*Q*]的更新。在对 MErrval 的编码完成后，通过当前待编码像素的预测误差 *Errval* 完成变量 *A*[*Q*]和 *B*[*Q*]的更新。为了增强对局部图像变化以及图像统计一般非平稳性的适应，当 *N*[*Q*]达到预设的阈值时，对 *A*[*Q*]和 *N*[*Q*]进行取半处理。本文中 *N*[*Q*]取值为 64。每次迭代过程中，通过 *B*[*Q*]控制 *C*[*Q*]进行加 1 或者减 1 运算，反过来 *B*[*Q*]也会被调整以反映 *C*[*Q*]的变化。同时，对变量加以钳制来限制其可能值的范围，有效提高了运算与统计的效率。对于一幅比特深度为 8 的灰度图像来说，*A*[*Q*]、*B*[*Q*]、*C*[*Q*]、*N*[*Q*]分别被初始化为 4、0、0、1。

3.4 游程编码模式

为了提高对图像平坦区域的编码性能，JPEG-LS 算法中使用游程编码模式对该区域中的像素进行编码。如图 3-10 所示，为游程编码模块的结构框图。

万方数据

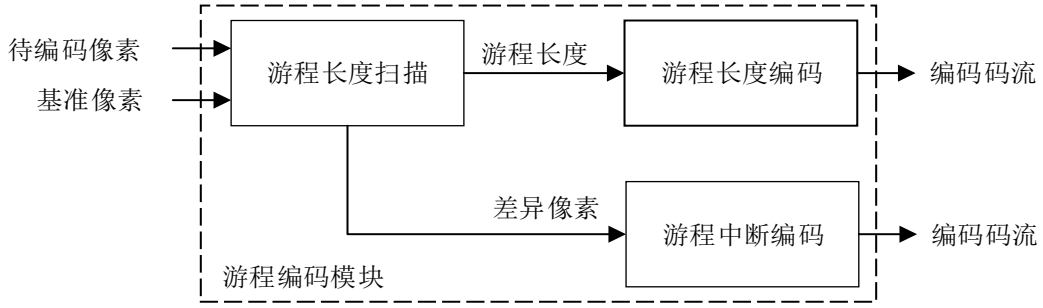


图 3-10 游程编码模块的结构框图

其中，游程长度扫描模块的功能是通过比较待编码像素与基准像素的大小完成游程长度的统计，并判断游程长度扫描终止的原因；游程长度编码模块的功能是对扫描的游程长度进行游程长度编码；游程中断编码模块的功能是对差异像素的预测误差进行 Golomb 编码。

如图 3-11 所示，为 JPEG-LS 算法中游程编码模式的流程图。在当前待编码像素进入游程编码模式后，首先对待编码像素进行游程长度的扫描；然后判断游程长度扫描终止的原因，若是由于扫描到行末像素，则对游程长度编码后退出游程编码模式，否则在游程长度编码后对差异像素进行游程中断编码；然后计算差异像素的预测值；然后进行预测误差的计算、校正以及模减；然后计算上下文参数、映射系数以及 Golomb 编码参数；然后对预测误差进行映射；然后对映射后的预测误差进行 Golomb 编码；最后进行上下文参数的更新，变量更新完成后退出游程编码模式。

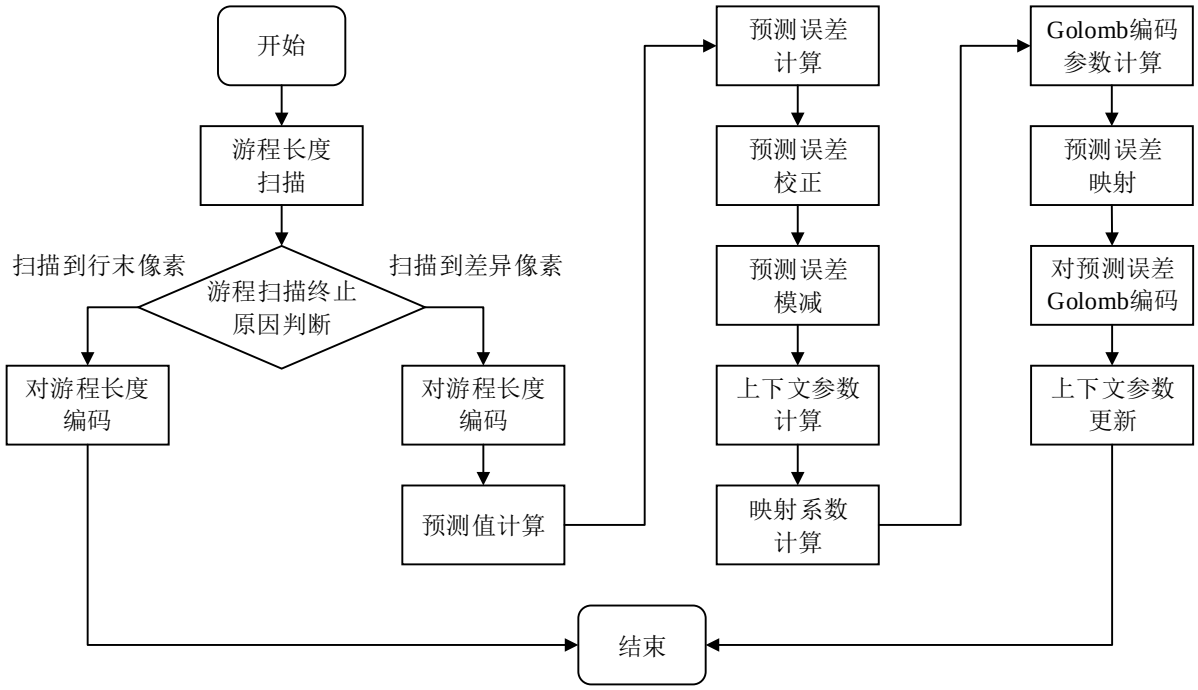


图 3-11 游程编码模式流程图

（1）游程长度扫描与编码

游程编码模式首先通过比较读入的待编码像素 *Ix* 与基准像素 *Ra* 是否相等，来进行游程长度 RUNcnt 的统计。当游程长度扫描结束后，对 RUNcnt 进行编码。通过比较所扫描的游程长度 RUNcnt 与入口向量表 J 所指向的游程长度 rg 的大小，来完成对游程长度的编码。如图 3-12 所示，为入口向量表 J。入口向量表 J 内的值为 rk 而不是 rg，$rg=2^{J[RUNindex]}$。对于每一个游程长度 rg 的编码，1 比

万方数据

特 1 应当被附加到编码码流中。每次得到一个游程长度 rg 时，入口向量表 J 的索引值 RUNindex 应当进行加 1 运算，直到其最大值为 31。如果游程长度扫描是由于扫描到行末像素而终止，即EOLine=1，那么在连续减去 rg 后剩余的游程长度大于 0 的情况下，一个额外的 1 比特 1 将被附加到编码码流中；如果游程长度扫描是由于扫描到差异像素而终止，即扫描到与基准像素 *Ra* 不相等的像素，那么先将 1 比特 0 附加到编码码流中，然后再将剩余的游程长度以 rk 位二进制形式附加到编码码流中，同时在 RUNindex 不小于 0 的情况下，对 RUNindex 进行减 1 运算，最后对差异像素进行游程中断编码。

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

rk	4	4	5	5	6	6	7	7	8	9	10	11	12	13	14	15
RUNindex	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

图 3-12 入口向量表 J

（2）游程中断编码

游程中断编码模块完成对差异像素预测误差的 Golomb 编码，其结构如图 3-13 所示。采用并行处理方式，对游程中断编码模块结构进行优化，即在预测误差模减完成后，就开始对上下文参数进行更新。

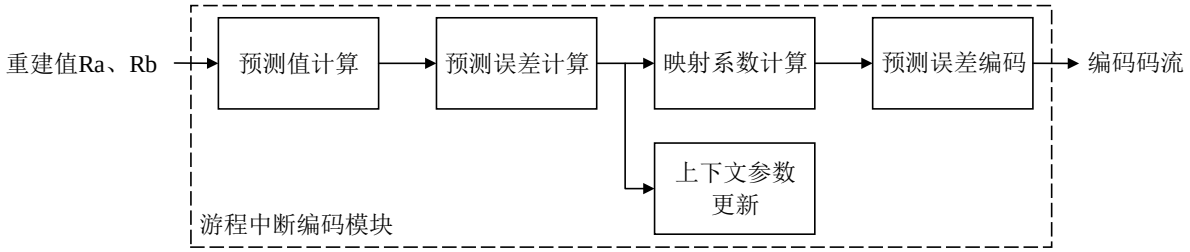


图 3-13 游程中断编码模块的结构框图

其中，预测值计算模块的功能是通过比较当前待编码像素邻近像素 *Ra*、*Rb* 的大小来计算预测值；预测误差计算模块的功能是完成预测值与当前待编码像素差值的计算，并对得到的预测误差进行符号校正以及模减，将预测误差映射到区间(-*RANGE*/2,⌈*RANGE*/2⌉-1]内；映射系数计算模块的功能是计算预测误差的映射系数，该映射系数能够自适应调整预测误差的映射过程，使得到的映射后的预测误差满足单边几何分布；上下文参数更新模块的功能是完成变量 *A*[*Q*]、*N*[*Q*]的更新；预测误差编码模块的功能是对映射后的预测误差进行 Golomb 编码。

JPEG-LS 算法中游程中断编码的过程为：首先计算当前待编码像素的预测值，然后进行预测误差的计算、校正以及模减，然后进行 Golomb 编码参数 k 以及映射系数的计算，然后进行预测误差的映射，然后对映射后的预测误差进行 Golomb 编码，最后进行上下文参数的更新。由于游程中断编码与正常编码基本一致，因此不对其做进一步的分析。

万方数据

3.5 码流拼接

码流拼接模块的功能是完成上次拼接剩余码流和本次编码码流的拼接，并且以 32 位位宽为单位进行编码码流的拼接输出。如图 3-14 所示，为码流拼接模块的结构框图。

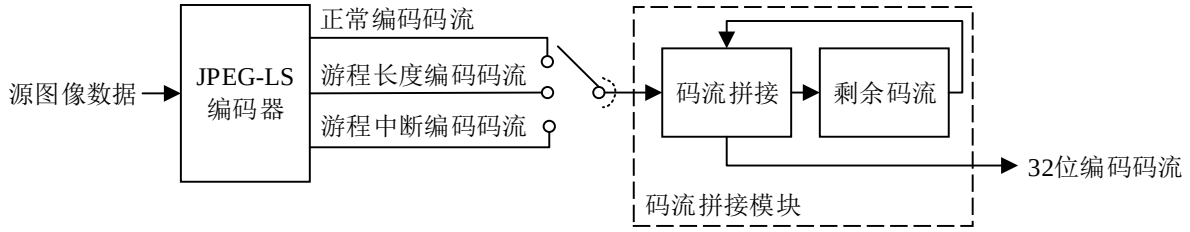


图 3-14 码流拼接的结构框图

其中，码流拼接模块的功能是将上次拼接剩余编码码流与输入的正常编码码流、游程长度编码码流、游程中断编码码流进行码流的拼接，对拼接完成后的编码码流以 32 比特为单位进行输出，对不足 32 比特的编码码流进行暂存。之所以将编码码流以 32 比特为单位进行拼接输出，是因为每次编码产生的编码码流位宽不超过 32 比特。

如图 3-15 所示，为 JPEG-LS 算法中码字输出的流程图。当编码完成后，开始进行码字的输出，首先判断编码码字的长度是否小于 8，若小于 8，则将编码码字保存，结束本次码字的拼接输出，否则将本次编码产生的码字和上次输出不足 8 比特编码码字组合成新的 8 比特码字，并输出，然后将编码码字的长度减去 8，并判断此时的编码码字长度是否小于 8，若小于 8，则将编码码字保存，结束本次编码码字的拼接输出，否则继续上述过程，直到编码码字的长度小于 8，保存此时的编码码字，结束本次码字输出。

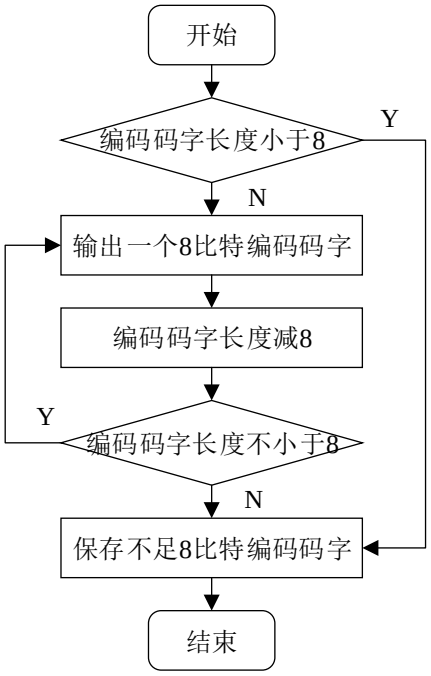


图 3-15 码字输出流程图

万方数据

3.6 本章小结

本章分析了 JPEG-LS 整体框架及其各个模块的算法，包括上下文建模、正常编码模式、游程编码模式以及码流拼接等，为 JPEG-LS 编码器的硬件设计提供基础；完成了 JPEG-LS 编码器及其各个模块的方案设计，包括上下文建模模块、正常编码模块、游程编码模块以及码流拼接模块等，为编码器的 FPGA 实现提供基础。通过采用流水线并行化处理方式，提高编码效率；通过和传统的因果模板的构建过程进行比较，完成改进的因果模板构建过程的方案设计，能够有效提高所设计编码器的吞吐率；通过和传统的 Golomb 编码参数 k 计算方法进行比较，提出使用二分查找法来完成 Golomb 编码参数 k 计算的方案设计，能够有效提高所设计编码器的时钟频率。

万方数据

第四章 JPEG-LS 无损压缩算法的 FPGA 实现

根据第三章的方案，设计 JPEG-LS 编码器的整体架构，并对其各个模块进行硬件电路的设计，包括各个模块的结构框图、模块时序控制的有限状态机、状态转换图，并对上下文建模模块、正常编码模块、游程编码模块、输出模块及其优化方案的 FPGA 实现进行详细阐述。

4.1 JPEG-LS 编码器整体架构

根据第三章的 JPEG-LS 算法分析与方案设计，本文设计的 JPEG-LS 编码器的整体硬件架构如图 4-1 所示。整个 JPEG-LS 编码器主要包括上下文建模模块、正常编码模块、游程编码模块以及输出模块等。时序控制模块控制整个 JPEG-LS 编码器编码过程。如表 4.1 所示，为 JPEG-LS 编码器的基本信号列表。

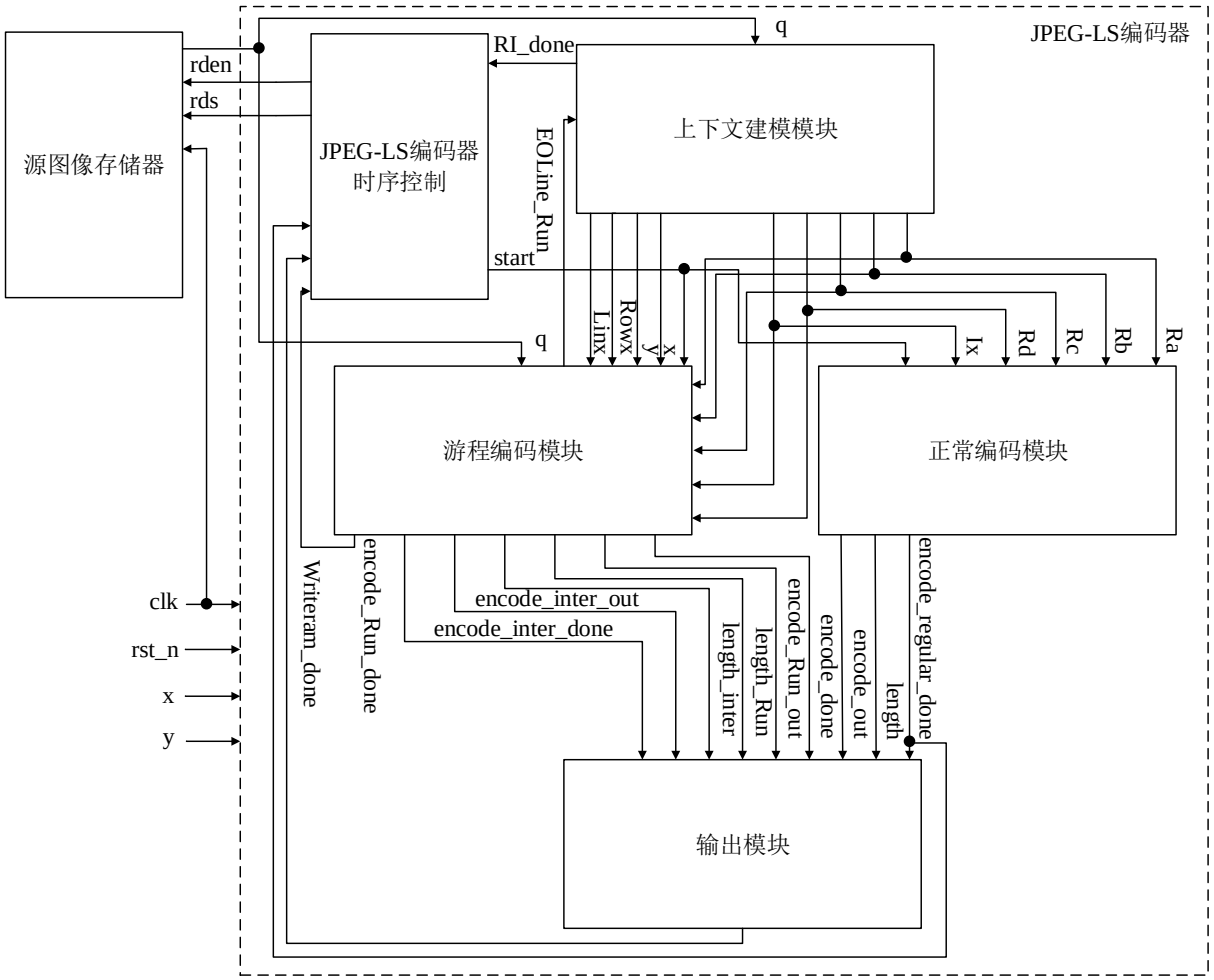


图 4-1 JPEG-LS 编码器整体硬件架构

时序控制模块控制着整个 JPEG-LS 编码器的编码过程，是最关键的模块。根据 JPEG-LS 算法的编码流程，采用状态机来完成 JPEG-LS 编码器的时序控制，设计了如图 4-2 所示的 JPEG-LS 编码器时序控制的状态转换图。JPEG-LS 编码器的时序控制模块检测编码使能信号，开始编码过程，控制着各个模块顺序执行，给出各个模块的初始化数据和使能信号，检测编码完成的标志位，对编码码流进行拼接、输出，直到整幅图像编码结束，拉低编码器的使能信号，结束一次编码过程。

万方数据

表 4.1 JPEG-LS 编码器基本信号列表

信号名	位宽	接口状态	备注
clk	1	输入	时钟信号，上升沿触发
rst_n	1	输入	复位信号，低电平有效
x	9	输入	源图像的高
y	9	输入	源图像的宽
rds	19	输出	源图像存储器读地址信号
rden	1	输出	源图像存储器读使能信号
start	1	中间变量	正常/游程编码模块选通信号
encode_Run_out	32	中间变量	游程长度编码码流
encode_inter_out	32	中间变量	游程中断编码码流
encode_out	32	中间变量	正常编码码流
encode_done	1	中间变量	游程长度编码完成标志位
encode_inter_done	1	中间变量	游程中断编码完成标志位
encode_Regular_done	1	中间变量	正常编码完成标志位
length_Run	6	中间变量	游程长度编码码流长度
length_inter	6	中间变量	游程中断编码码流长度
length	6	中间变量	正常编码码流长度
Writeram_done	1	中间变量	编码码流写入存储器标志位

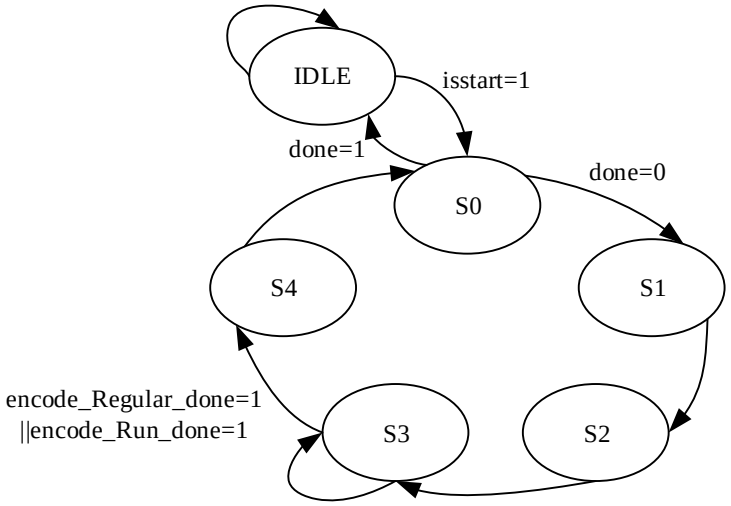


图 4-2 JPEG-LS 编码器时序控制的状态转换图

状态 IDLE 检测编码器的编码使能信号，若编码器接收到源图像的宽和高信息，即该使能信号为高电平，则开始对源图像进行编码处理，进入状态 S0，否则一直处于此状态。

状态 S0 判断整幅图像编码是否结束，若结束，则跳转到状态 IDLE，否则跳转到状态 S1。

状态 S1 构建当前待编码像素的因果模板，因果模板构建完成后跳转到状态 S2。

状态 S2 根据 S1 状态构建的因果模板进行当前待编码像素局部梯度值的计算以及编码模式的选择，若局部梯度值全为 0，则进入游程编码模块，否则进入正常编码模块，编码模式选择完成后跳转到状态 S3。

状态 S3 检测正常编码模块以及游程编码模块的编码完成标志位，若该标志位为高电平，则跳转到状态 S4，否则一直处于此状态；

状态 S4 更新下一个待编码像素的读地址信号以及行末像素的标志位，更新完成后跳转到状态 S0。

万方数据

4.2 上下文建模模块

上下文建模模块主要由局部梯度值计算、编码模式选择、局部梯度值量化与合并等子模块组成。如图 4-3 所示，为上下文建模模块的结构框图，其中时序控制模块控制整个上下文建模模块的处理过程。如表 4.2 所示，为上下文建模模块的基本信号列表。如图 4-4 所示，为上下文建模模块的时序图。

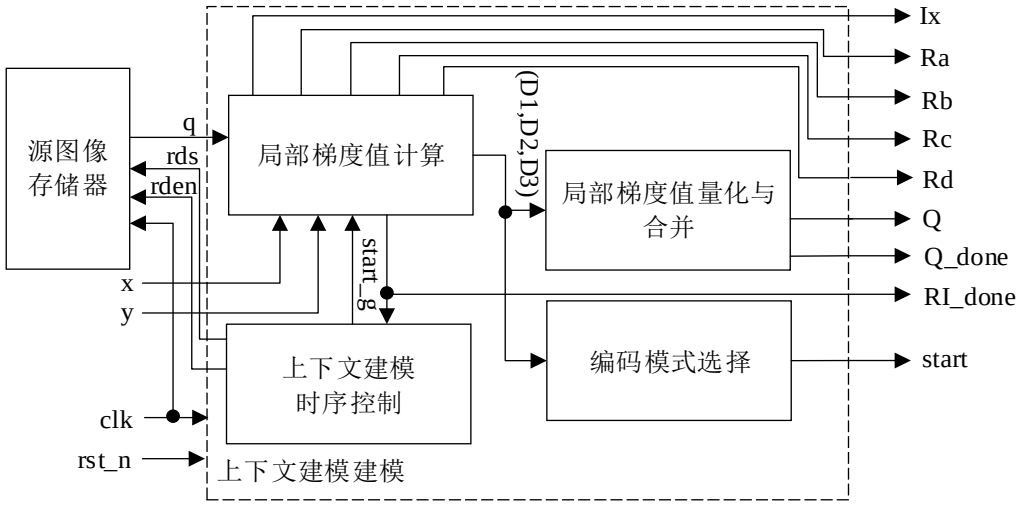


图 4-3 上下文建模建模的结构框图

表 4.2 上下文建模模块基本信号列表

信号名	位宽	接口状态	备注
Ix	8	输出	因果模板中位置 x 上的像素
Ra	8	输出	因果模板中位置 a 上的像素
Rb	8	输出	因果模板中位置 b 上的像素
Rc	8	输出	因果模板中位置 c 上的像素
Rd	8	输出	因果模板中位置 d 上的像素
Q	9	输出	上下文参数
Q_done	1	输出	上下文建模完成标志位
RI_done	1	输出	因果模板构建完成标志位

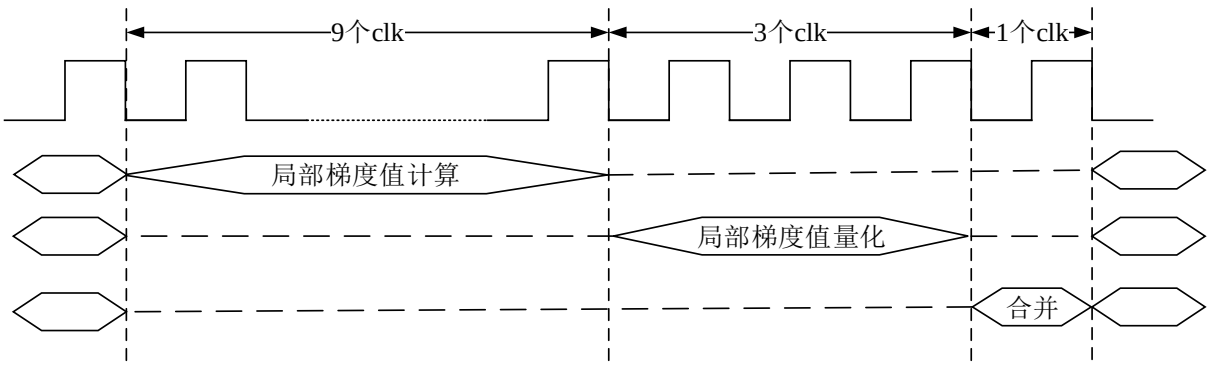


图 4-4 上下文建模模块的时序图

万方数据

4.2.1 局部梯度值计算

如图 4-5 所示，为局部梯度值计算的结构框图。当编码器检测到编码开始信号时，开始局部梯度值的计算。首先通过时序控制模块来控制待编码像素因果模板的构建过程，构建完成后将标志位 RI_done 赋值为 1，然后将 Ra、Rb、Rc 和 Rd 送入加法器，得到局部梯度值 D1、D2、D3。

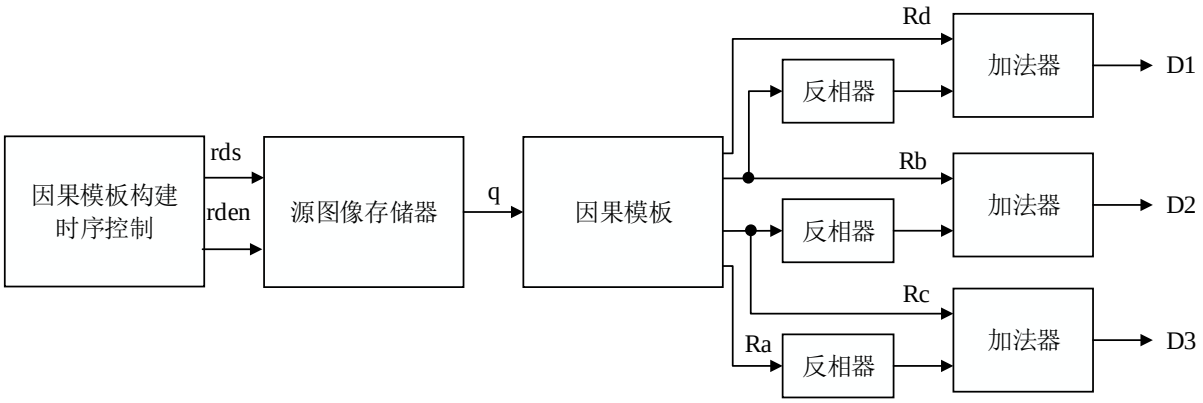


图 4-5 局部梯度值计算的结构框图

由于对源图像进行的是无损压缩，其像素的重建值等于该像素值，因此利用这一特点对因果模板的构建过程进行优化。由于处理图像边界处像素因果模板构建过程的需要，本文将比特深度为 8、尺寸为 256*256 的灰度图像进行边界填充后，在 ROM（Read-Only Memory）IP（Intellectual Property）核生成过程中以 COE 文件格式写入到位宽为 8 比特、深度为 66306 的 ROM 中。除了图像第一列待编码像素因果模板的构建需要访问 5 次源图像存储器，分别读出待编码像素 Ix 及其 4 个邻近像素 Ra、Rb、Rc 和 Rd，其余待编码像素只需访问 2 次源图像存储器，分别读出待编码像素 Ix 及其 1 个邻近像素 Rd，余下 3 个邻近像素均取自上一个已编码像素的因果模板，具体过程为：将上一个已编码像素因果模板位置 x 上的像素 Ix、位置 b 上的像素 Rb 以及位置 d 上的像素 Rd 分别赋值给当前待编码像素因果模板位置 a 上的像素 Ra、位置 c 上的像素 Rc 以及位置 b 上的像素 Rb，节约了 3 次访问源图像存储器的时间，有效提高了编码器的吞吐率。如图 4-6 所示，为优化后因果模板构建的结构框图，其中 Ix1、Rb1、Rd1 为上一个已编码像素因果模板中的像素值，Ix、Ra、Rb、Rc、Rd 为当前待编码像素因果模板中的像素值。如图 4-7 所示，为因果模板构建时序控制的状态转换图。

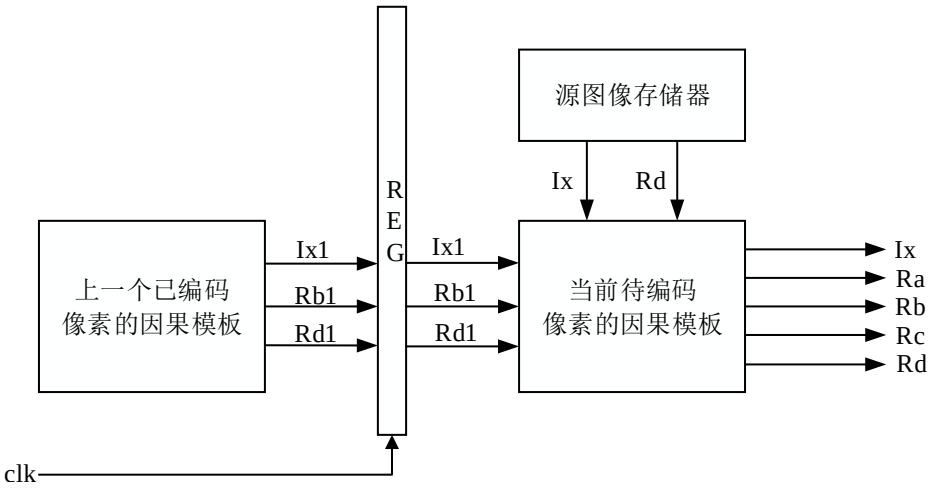


图 4-6 优化后因果模板构建的结构框图

万方数据

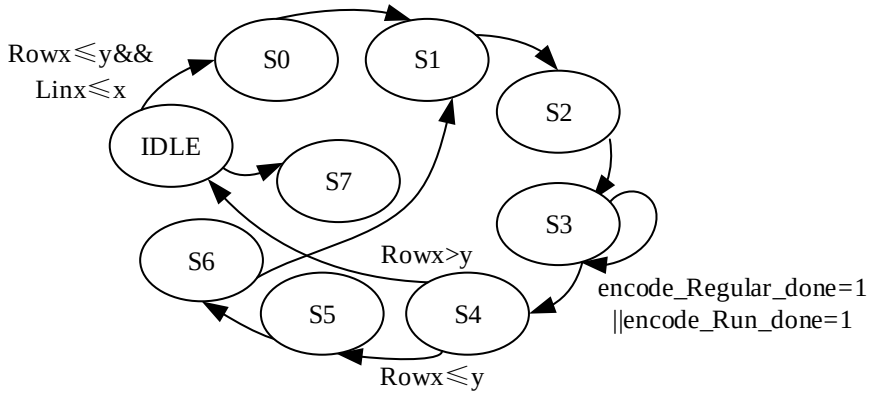


图 4-7 因果模板构建时序控制的状态转换图

状态 IDLE 判断条件(Rowx≤y&&Linx≤x)是否成立，若成立，则跳转到状态 S0，否则跳转到状态 S7。

状态 S0 产生读地址信号 rds_Ix、rds_Ra、rds_Rb、rds_Rc、rds_Rd，读取像素 Ix、Ra、Rb、Rc、Rd，并分别暂存到寄存器 Ix_temp、Ra_temp、Rb_temp、Rc_temp、Rd_temp 中，然后跳转到状态 S1。

状态 S1 将因果模板构建完成标志位赋值为 1，并暂存到寄存器 RI_done，将寄存器 Ix_temp、Ra_temp、Rb_temp、Rc_temp、Rd_temp 中的像素值分别暂存到寄存器 Ix、Ra、Rb、Rc、Rd 中，然后跳转到状态 S2。

状态 S2 将寄存器 RI_done 清零，然后跳转到状态 S3。

状态 S3 检测编码完成标志位，当检测到该标志位为高电平时，对已编码像素所处的列信号进行加 1 运算，并暂存到寄存器 Rowx 中，将游程编码模块编码完成时因果模板中的像素值 Ix、Rb、Rd 分别暂存到寄存器 Ix_temp、Rb_temp、Rd_temp 中，然后跳转到状态 S4，否则一直处于此状态。

状态 S4 判断条件 Rowx>y 是否成立，若成立，则对 Rowx 做减 y 运算后将其暂存到寄存器 Rowx 中，对 Linx 做加 1 运算后将其暂存到寄存器 Linx 中，然后跳转到状态 IDLE，否则跳转到状态 S5。

状态 S5 产生读地址信号 rds_Ix、rds_Rd，读取像素 Ix、Rd，读取完成后跳转到状态 S6。

状态 S6 将寄存器 Ix_temp、Rd_temp、Rb_temp 中的像素值分别暂存到寄存器 Ra_temp、Rb_temp、Rc_temp 中，然后跳转到状态 S1。

状态 S7 将整幅图像编码完成的标志位赋值为 1，并暂存到寄存器 done 中。

4.2.2 编码模式选择

在局部梯度值计算完成后，根据局部梯度值进行编码模式的选择。如图 4-8 所示，为编码模式选择的结构框图。当局部梯度值全部为 0 时，意味着后续待编码像素将极大概率进入图像的平坦区域，将选通信号 start 赋值为 0，选通游程编码模块，对当前待编码像素进行游程编码；当局部梯度值不全为 0 时，意味着后续待编码像素将极大概率进入图像的非平坦区域，将将选通信号 start 赋值为 1，选通正常编码模块，对当前待编码像素进行正常编码。

万方数据

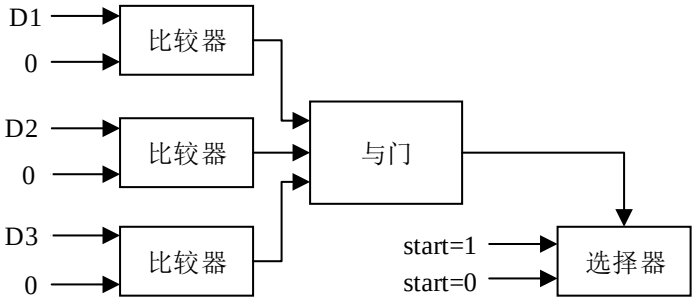


图 4-8 编码模式选择的结构框图

4.2.3 局部梯度值量化与合并

如图 4-9 所示，为局部梯度值量化与合并的结构框图。首先将局部梯度值 D1、D2、D3 分别量化为 Q1、Q2、Q3，然后完成校正符号的生成，然后根据校正符号完成局部梯度值的符号校正，最后，通过将符号校正后的局部梯度值 Q1、Q2、Q3 合并为 Q。

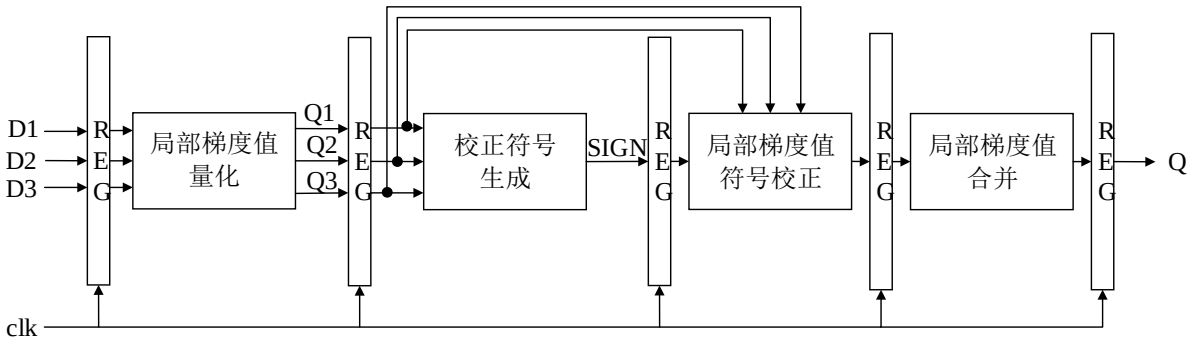


图 4-9 局部梯度值量化与合并的结构框图

其中，由于局部梯度值计算过程引入了负数，因此在设计过程中使用符号位和数值位来表示局部梯度值，因此局部梯度值量化模块先比较符号位，然后再比较数值位，最后完成局部梯度值的量化；校正符号生成模块通过比较器完成校正符号的生成，使最终生成的上下文参数 Q 为非负数；局部梯度值符号校正模块也通比较器完成局部梯度值符号的校正；局部梯度值合并模块通过移位寄存器、加法器等来完成局部梯度值的合并。

本文中所采用的 Q 的合并函数为 Q=(Q1*9+Q2)*9+Q3。因为乘法器不仅占用逻辑资源多，而且运算时间消耗多，极有可能成为关键路径，又因为将矢量（Q1,Q2,Q3）合并为 Q 的合并系数确定，所以可以通过对 Q1、Q2 进行左移操作来实现局部梯度值合并过程中的乘法运算。但是，如果仅仅使用移位运算代替乘法运算，将合并公式优化为 Q=(Q1<<6)+(Q1<<4)+(Q2<<3)+Q1+Q2+Q3，那么该合并过程的组合逻辑路径将有 5 级，这仍将影响到编码器的时钟频率，因此，本文对局部梯度值的合并过程进一步进行了优化，将合并公式优化为 Q=((Q1+Q2)+(Q3+(Q2<<3))+((Q1<<4)+(Q1<<6))，这样就将该合并过程的组合逻辑路径缩短为 3 级。在不对该合并过程采用插入寄存器进行处理的情况下，减小了 Q 所在组合逻辑路径的延迟，有效提高了编码器的时钟频率，同时保证了编码器的吞吐率。如图 4-10 所示，为优化后局部梯度值合并的结构框图。

万方数据

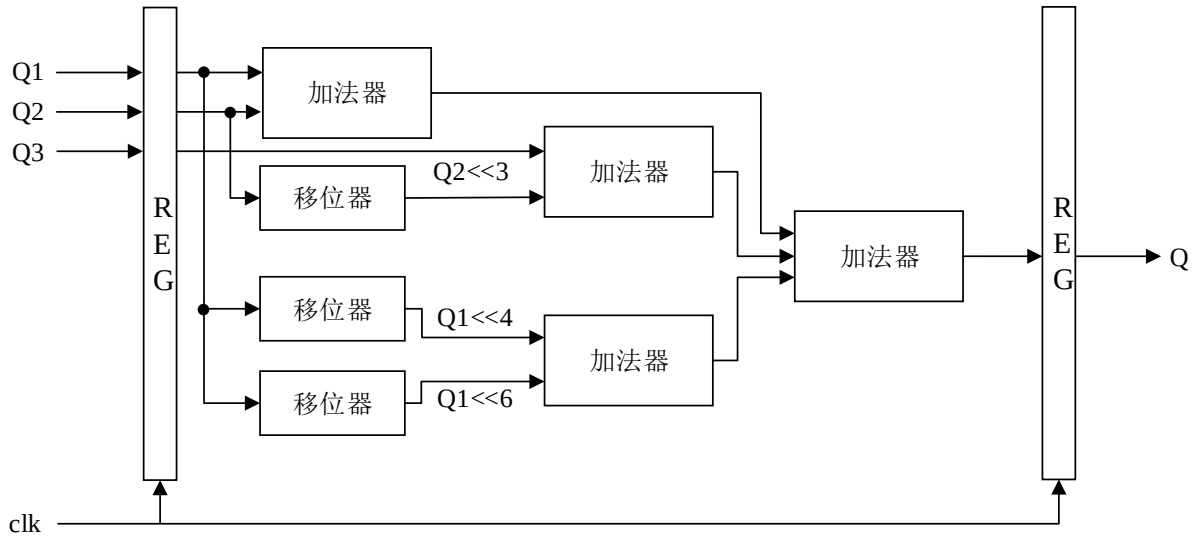


图 4-10 优化后局部梯度值合并的结构框图

4.3 正常编码模块

正常编码模块主要由预测值计算、预测误差计算、预测误差编码以及上下文参数更新等子模块组成。如图 4-11 所示，为正常编码模块的结构框图，其中时序控制模块用来控制整个正常编码过程。如表 4.3 所示，为正常编码模块的基本信号列表。

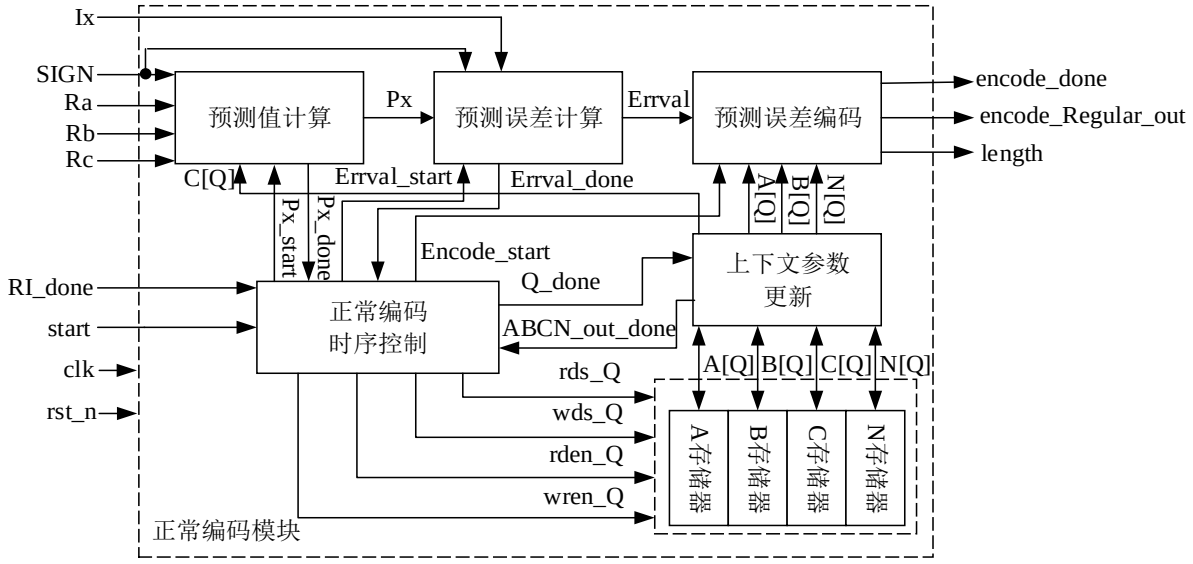


图 4-11 正常编码模块的结构框图

如图 4-12 所示，为本文设计的 JPEG-LS 编码器正常编码模块的时序图。其中，由于预测值计算模块中通过固定预测器得到的预测值在上下文建模过程中已经完成，因此图中所示的预测值计算模块只包括预测值的校正以及映射；由于 Golomb 编码参数 k 的计算在预测值计算完成时也刚好完成，因此图中所示的预测误差编码模块只包括预测误差映射以及 Golomb 编码；上下文参数更新模块包括变量的读取、变量的更新以及变量的写入。

万方数据

表 4.3 正常编码模块基本信号列表

信号名	位宽	接口状态	备注
SIGN	1	输入	校正符号
Px	8	中间变量	预测值
Errval	9	中间变量	预测误差
Errval_done	1	中间变量	模减完成标志位
rds_Q	9	中间变量	A、B、C、N 存储器读地址信号
wds_Q	9	中间变量	A、B、C、N 存储器写地址信号
rden_Q	1	中间变量	A、B、C、N 存储器读使能信号
wren_Q	1	中间变量	A、B、C、N 存储器写使能信号
A[Q]	14	中间变量	Q 对应预测误差绝对值的和
B[Q]	9	中间变量	Q 对应预测误差的和
C[Q]	9	中间变量	Q 对应预测误差的校正值
N[Q]	9	中间变量	Q 出现的次数

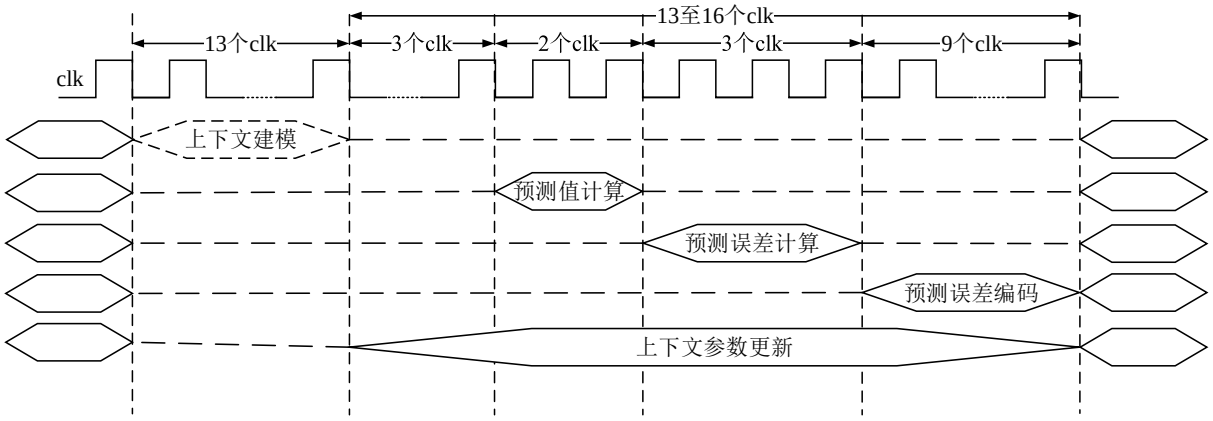


图 4-12 正常编码模块的时序图

4.3.1 预测值计算

预测值计算模块包括预测值计算、预测值校正以及预测值映射等子模块。如图 4-13 所示，为预测值计算模块的结构框图。当正常编码模块检测到因果模板构建完成的标志位 RI_done 为高电平时，通过固定预测器得到预测值；然后检测变量读出的完成标志位 ABCN_out_done，当检测到 ABCN_out_done 为高电平时，通过变量 C[Q]以及 SIGN 完成预测值 Px 的校正；最后将预测值映射到区间[0,MAXVAL]内。

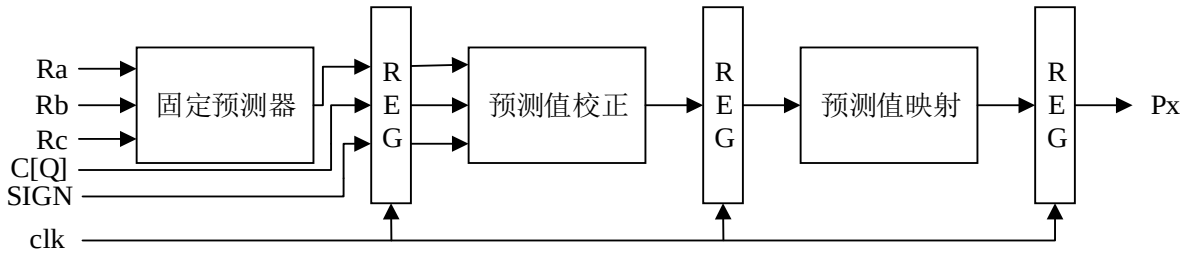


图 4-13 预测值计算模块的结构框图

如图 4-14 所示，为固定预测器的结构框图。通过固定预测器得到预测值的具体过程为：若通过比较器得到Rc≥max(Ra,Rb)，则选择器将min(Ra,Rb)赋值给 Px；若通过比较器得到Rc≤min(Ra,Rb)，则选择器将max(Ra,Rb)赋值给 Px；若通过比较器发现以上两种情况均不满足，则选择器将Ra+Rb-Rc

万方数据

赋值给 Px。若固定预测器计算出当前待编码像素处于水平边界，则预测值与紧邻其左侧的像素值相等，将 Ra 赋值给 Px；若固定预测器计算出当前待编码像素处于垂直边界，则预测值与紧邻其正上方的像素值相等，将 Rb 赋值给 Px；若固定预测器计算出当前待编码像素既不处于水平边界，又不处于垂直边界，则预测值既不与紧邻其左侧的像素值相等，又不与紧邻其正上方的像素值相等，将 Ra+Rb-Rc赋值给 Px。

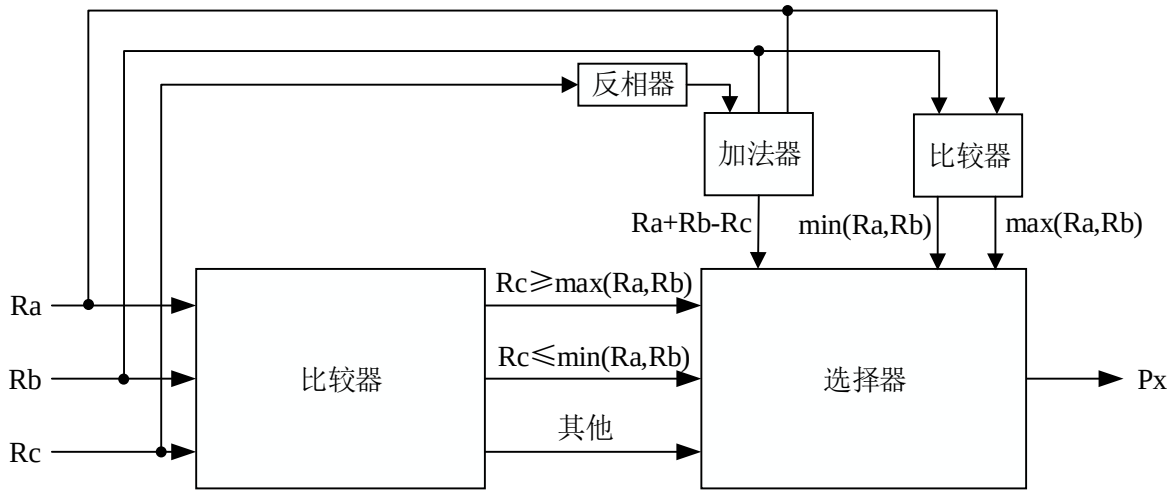


图 4-14 固定预测器的结构框图

如图 4-15 所示，为预测值校正与映射的结构框图。预测值校正与映射的具体过程为：当上下文更新模块检测到上下文参数 Q 的完成标志位 Q_done 为高电平时，首先通过控制读使能信号 rden，以 Q 为读地址信号访问 C 双口 RAM，读出 C[Q]，在 C[Q]读出后，将完成标志位 ABCN_out_done 赋值为 1；然后当预测值计算模块检测到 ABCN_out_done 为高电平时，将预测值 Px 和 C[Q]送入加法器中，得到校正后的预测值 Px；最后，将校正后的预测值送入比较器中，若Px>MAXVAL，则将 MAXVAL 赋值给 Px；若Px<0，则将 0 赋值给 Px；若0≤Px≤MAXVAL，则 Px 保持不变，将校正后的预测值映射到区间[0,MAXVAL]内。

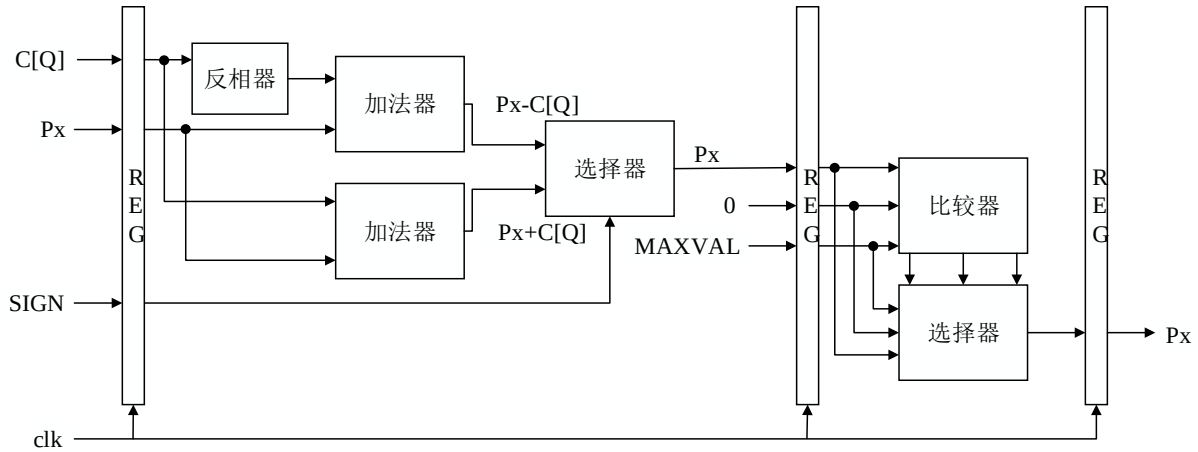


图 4-15 预测值校正与映射的结构框图

万方数据

4.3.2 预测误差计算

预测误差计算模块包括预测误差计算、预测误差校正以及预测误差模减等子模块。如图 4-16 所示，为预测误差计算模块的结构框图。在预测值计算模块完成当前待编码像素预测值的映射后，将预测值 Px 送入预测误差计算模块。预测误差计算模块首先通过如图 4-17 所示的预测误差计算模块完成当前待编码像素 Ix 与其预测值 Px 的差值计算，得到预测误差 Errval；然后根据校正符号 SIGN 完成预测误差 Errval 的校正，若SIGN=-1，则将 Errval 取反后赋值给 Errval，否则 Errval 保持不变；最后完成校正后预测误差 Errval 的模减，将 Errval 规整到区间[-(RANGE-1)/2,RANGE/2)]内。

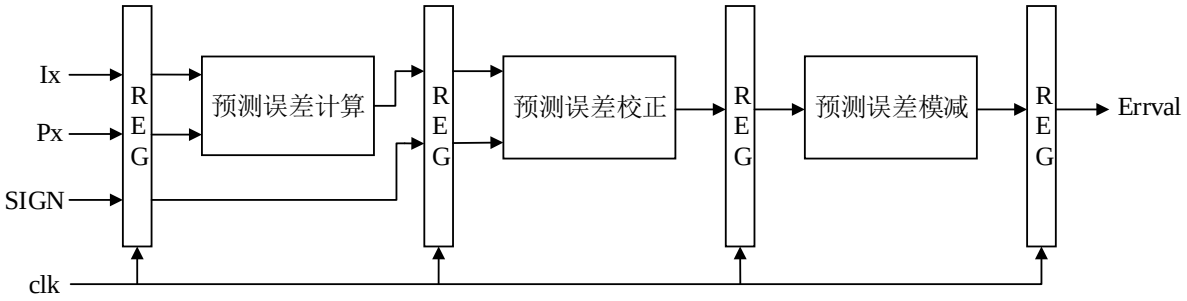


图 4-16 预测误差计算模块的结构框图

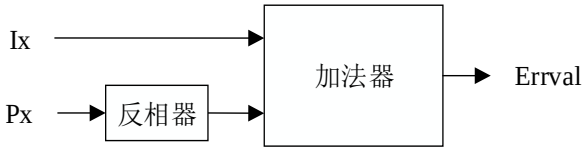


图 4-17 预测误差计算框图

4.3.3 预测误差编码

预测误差编码模块包括 Golomb 编码参数 k 计算、预测误差映射以及 Golomb 编码等子模块。由于预测误差的映射所涉及到的是乘 2 运算以及加减运算，硬件实现简单，故不对预测误差映射模块做进一步分析。

如图 4-18 所示，为本文设计 JPEG-LS 编码器预测误差编码模块的时序图。

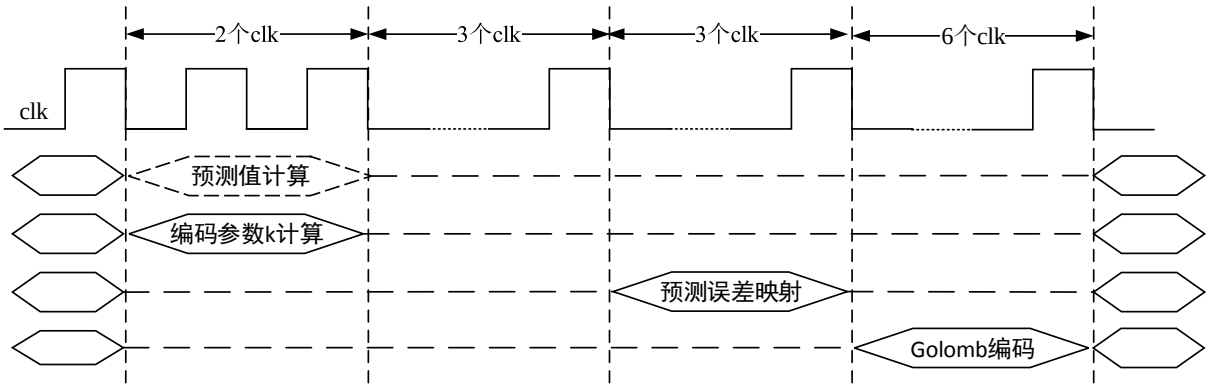


图 4-18 预测误差编码模块的时序图

（1）Golomb 编码参数 k 计算

通过比较器以及移位寄存器等来完成 Golomb 编码参数 k 的计算。对于一幅比特深度为 8 的灰

万方数据

度图像来说，Golomb 编码参数 k 的最大值为 7，最小值为 0。当变量读出的标志位 ABCN_out_done 为高电平时，开始 Golomb 编码参数 k 的计算。若采用一般的顺序比较方法，即先比较N[Q]<<0和A[Q]的大小，再比较N[Q]<<1和 A[Q]的大小，依次类推，则最多需要比较 7 次 A[Q]和 N[Q]的大小才能确定 k 的值。这就意味着该组合逻辑路径的延迟包括 7 级移位延迟和 7 级比较延迟。若不对此组合逻辑路径加以处理，则该路径将极有可能成为关键路径，进而影响编码器的时钟频率；若简单地采用插入寄存器进行处理的方法，则将影响编码器的吞吐率。如图 4-19 所示，为 A[Q]和 N[Q]移位比较的结构框图。

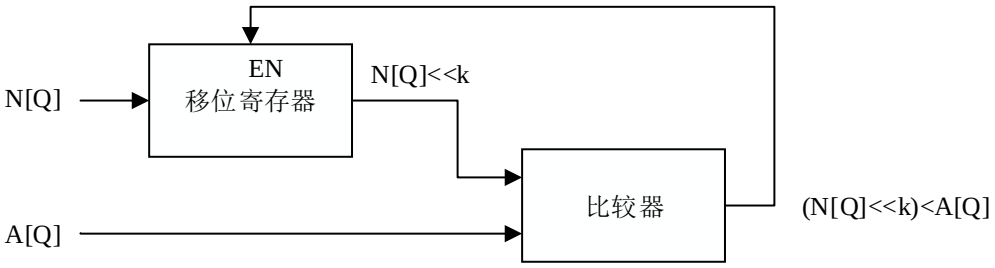


图 4-19 A[Q]和 N[Q]移位比较的结构框图

本文对 Golomb 编码参数 k 的计算路径进行了优化，采用二分查找法来比较 N[Q]和 A[Q]的大小，以确定 k 的值。如图 4-20 所示，为优化后 Golomb 编码参数 k 计算的结构框图，图中 A 和 N 分别为 A[Q]和 N[Q]。

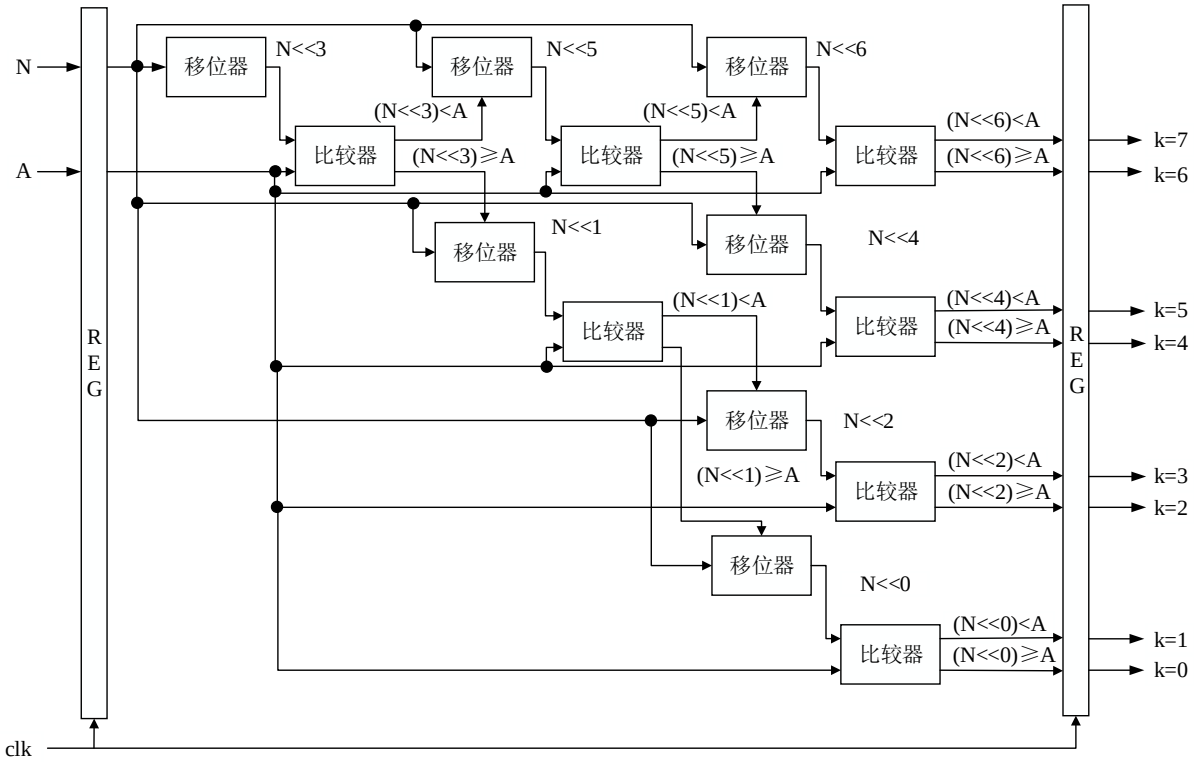


图 4-20 Golomb 编码参数 k 计算的结构框图

当检测到 ABCN_out_done 为高电平时，先将 N[Q]送入移位寄存器左移 3 位，再将移位后的 N[Q]和 A[Q]送入比较器，若满足(N[Q]<<3)<A[Q]，则将移位前的 N[Q]送入移位寄存器左移 5 位，再将移位后的 N[Q]和 A[Q]送入比较器，否则将移位前的 N[Q]送入移位寄存器左移 1 位，再将移位后的

万方数据

N[Q]和 A[Q]送入比较器，若满足(N[Q]<<1)<A[Q]，则将移位前的 N[Q]送入移位寄存器左移 2 位，再将移位后的 N[Q]和 A[Q]送入比较器，否则将移位前的 N[Q]送入移位寄存器左移 0 位，再将移位后的 N[Q]和 A[Q]送入比较器，若满足(N[Q]<<0)<A[Q]，则将 1 赋值给 k，否则将 0 赋值给 k，若满足(N[Q]<<2)<A[Q]，则将 3 赋值给 k，否则将 2 赋值给 k，若满足(N[Q]<<5)<A[Q]，则将移位前的 N[Q]送入移位寄存器左移 6 位，再将移位后的 N[Q]和 A[Q]送入比较器，否则将将移位前的 N[Q]送入移位寄存器左移 4 位，再将移位后的 N[Q]和 A[Q]送入比较器，若满足(N[Q]<<6)<A[Q]，则将 7 赋值给 k，否则将 6 赋值给 k，若满足(N[Q]<<4)<A[Q]，则将 5 赋值给 k，否则将 4 赋值给 k。然后将 N[Q]送入移位寄存器左移 1 位，再将移位后的 N[Q]和 A[Q]送入比较器，若满足(N[Q]<<1)<A[Q]，则将 2 赋值给 k，否则 k 保持不变，一直进行此操作直到移位后的 N[Q]不满足(N[Q]<<k)<A[Q]，并将 k 值计算完成标志位赋值为 1，并暂存到寄存器 k_done 中。通过采用二分查找法对 Golomb 编码参数 k 计算的组合逻辑路径优化后，该路径最多只包含 3 级移位延迟以及 3 级比较延迟，有效提高了编码器的时钟频率以及吞吐率。

（2）Golomb 编码

当预测误差映射完成标志位 MErrval_done 为高电平时，开始对映射后的预测误差进行 Golomb 编码。Golomb 编码模块主要通过比较器、移位器以及加法器等来实现。如图 4-21 所示，为 Golomb 编码时序控制的状态转换图。

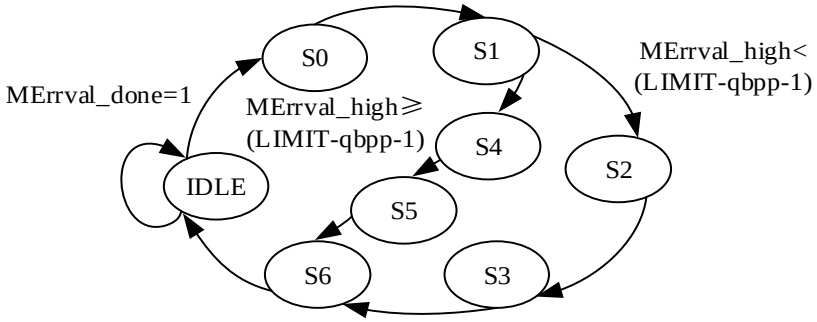


图 4-21 Golomb 编码时序控制的状态转换图

状态 IDLE 检测预测误差映射完成的标志位，若该标志位为高电平，则将映射后的预测误差暂存到寄存器 MErrval_high 以及寄存器 MErrval_reg 中，然后跳转到状态 S0，否则一直处于此状态。

状态 S0 对暂存在寄存器 MErrval_high 中映射后的预测误差进行右移 k 位操作，并暂存到寄存器 MErrval_high 中，然后跳转到状态 S1。

状态 S1 判断条件MErrval_high<LIMIT-qbpp-1是否成立，若成立，则将编码码流的长度更新为 MErrval_high+1+k，并暂存到寄存器 length 中，然后跳转到状态 S2，否则跳转到状态 S4。

状态 S2 截取 MErrval_reg 的低 k 位，并暂存到寄存器 MErrval_lowk 中，将暂存在寄存器 encode_out 的编码码流进行左移 MErrval_high+1 位操作，并将 1 比特 1 暂存到寄存器 encode_out 的最低位，然后跳转到状态 S3。

状态 S3 先对暂存在寄存器 encode_out 中的编码码流进行左移 k 位操作，然后将暂存的 MErrval_lowk 以二进制的形式暂存到寄存器 encode_out 的低 k 位上，将编码完成标志位赋值为 1，并暂存到寄存器 encode_Regular_done 中，然后跳转到状态 S6。

万方数据

状态 S4 截取 MErrval_reg-1 的低 k 位，并暂存到寄存器 MErrval_lowk 中，对暂存位寄存器 encode_out 中的编码码流进行左移(LIMIT-qbpp-1)+1位操作，然后将 1 比特 1 暂存到寄存器 encode_out 的最低位，然后跳转到状态 S5。

状态 S5 先将暂存在寄存器 encode_out 中的编码码流进行左移 qbpp 位操作，然后将暂存的 MErrval_lowk 暂存到寄存器 encode_out 的低 qbpp 位上，将编码完成标志位赋值为 1，并暂存到寄存器 encode_Regular_done 中，然后跳转到状态 S6。

状态 S6 将寄存器 encode_Regular_done 清零，然后跳转到状态 IDLE。

4.3.4 上下文参数更新

上下文更新模块完成变量 A[Q]、B[Q]、C[Q]、N[Q]的读取、更新以及写入。本文中采用双口 RAM（Random Access Memory）IP 核来存储 A[Q]、B[Q]、C[Q]、N[Q]。当上下文参数更新模块检测到 Q_done 为高电平时，开始读取变量 A[Q]、B[Q]、C[Q]、N[Q]，当上下文参数更新模块检测到预测误差计算模块的完成标志位 Errval_done 为高电平时，进行变量 A[Q]、B[Q]、C[Q]、N[Q]的更新，更新后将变量 A[Q]、B[Q]、C[Q]、N[Q]写入相应的双口 RAM。如图 4-22 所示，为上下文参数更新的结构框图，其中 A[Q]、B[Q]、C[Q]、N[Q]为更新前变量，A1[Q]、B1[Q]、C1[Q]、N1[Q]为更新后变量。

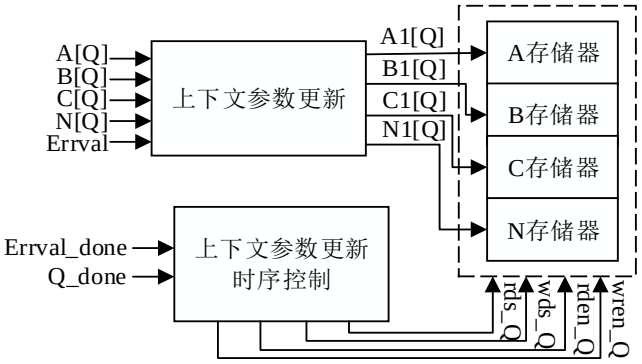


图 4-22 上下文参数更新的结构框图

JPEG-LS 软件压缩过程中，变量 A[Q]、B[Q]、C[Q]、N[Q]的更新在每次编码完成之后进行；常见的硬件设计过程中，下一个待编码像素因果模板的构建同样也在当前待编码像素编码完成之后进行。而由于 FPGA 硬件设计的并行性，因此本文对上下文参数更新模块进行优化。当上下文参数更新模块检测到预测误差计算模块的完成标志 Errval_done 为高电平时，就开始对变量 A[Q]、B[Q]、C[Q]、N[Q]进行更新，而不必等到编码结束后才开始对 A[Q]、B[Q]、C[Q]、N[Q]进行更新，这就节省了上下文参数更新模块所需要的 5 到 8 个 clk 时钟周期；同时，由于上下文参数更新模块所需要的时钟周期足够用来对源图像存储器进行 2 次读操作，因此当上下文参数更新模块检测到上下文建模的完成标志位 Q_done 为高电平时，就开始对源图像存储器进行读操作，完成下一个待编码像素 Ix 及其因果模板中位置 d 上像素 Rd 的读取，而不必等到当前待编码像素编码结束后才开始下一个待编码像素因果模板的构建，这就节省了 2 次读源图像存储器的时间。通过对编码器结构的优化，包括上下文参数更新模块开始信号的优化以及对下一个待编码像素因果模板构建过程的优化，节省

万方数据

了 11 到 14 个 clk 时钟周期，有效提高了编码器的吞吐率。

如图 4-23 所示，为上下文参数更新模块时序控制的状态转换图。

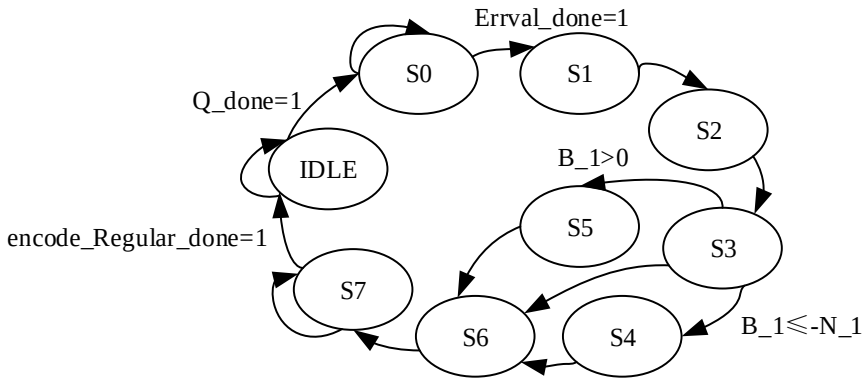


图 4-23 上下文参数更新时序控制的状态转换图

状态 IDLE 检测上下文建模完成的标志位，当检测到该标志位为高电平时，开始从存储器中读取变量 A[Q]、B[Q]、C[Q]、N[Q]，变量读出后将变量读取标志位赋值为 1，并暂存到寄存器 ABCN_out_dne 中，同时将变量分别暂存到寄存器 A_1、B_1、C_1、N_1，从源图像存储器中读出下一个待编码像素及其因果模板中位置 d 上的像素值，并分别暂存到寄存器 Ix_next 以及 Rd_next 中，然后跳转到状态 S0，否则一直处于此状态。

状态 S0 检测预测误差模减完成的标志位，当检测到该标志位时，对变量 A_1 做加模减后预测误差绝对值的运算，对变量 B_1 做加模减后预测误差的运算，并分别暂存到寄存器 A_1、B_1 中；对寄存器 ABCN_out_dne 清零；然后跳转到状态 S1，否则一直处于此状态。

状态 S1 判断条件N_1=RESET是否成立，若成立，则分别对变量 A_1、B_1、N_1 进行右移一位操作，并分别暂存到寄存器 A_1、B_1、N_1 中，然后跳转到状态 S2 中。

状态 S2 对变量 N_1 做加 1 运算，并暂存到寄存器 N_1 中，然后跳转到状态 S3。

状态 S3 判断B_1≤-N_1是否成立，若成立，则对变量 B_1 做加 N_1 的运算，并暂存到寄存器 N_1 中，然后跳转到状态 S4；否则若B_1>0成立，则对变量 B_1 做减 N_1 的运算，并暂存到寄存器 N_1 中，然后跳转到状态 S5；否则跳转到状态 S6。

状态 S4 判断C_1>MIN_C是否成立，若成立，则对变量 C_1 做减 1 运算，并暂存到寄存器 C_1 中；判断B_1<N_1是否成立，若成立，则对 1 做减 N_1 的运算，并暂存到寄存器 B_1 中；然后跳转到状态 S6。其中，MIN_C=-128为 C[Q]的最小值。

状态 S5 判断C_1<MAX_C是否成立，若成立，则对变量 C_1 做加 1 运算，并暂存到寄存器 C_1 中；判断B_1>0是否成立，若成立，则将变量 B_1 赋值为 0，并暂存到寄存器 B_1 中；然后跳转到状态 S6。其中，MAX_C=127为 C[Q]的最大值。

状态 S6 对存储器做写操作，当将变量 A_1、B_1、C_1、N_1 分别写入相应的存储器后，跳转到状态 S7。

状态 S7 检测正常编码结束的标志位，当该标志位为高电平时，跳转到状态 IDLE，否则一直处于此状态。

万方数据

4.4 游程编码模块

游程编码模块主要由游程长度扫描、游程长度编码以及游程中断编码等子模块组成。当游程编码模块检测到开始信号 start 为低电平时，将当前待编码像素送入游程编码模块进行编码。如图 4-24 所示，为游程编码模块结构框图，其中时序控制模块用来控制整个游程编码过程。如表 4.4 所示，为游程编码模块的基本信号列表。

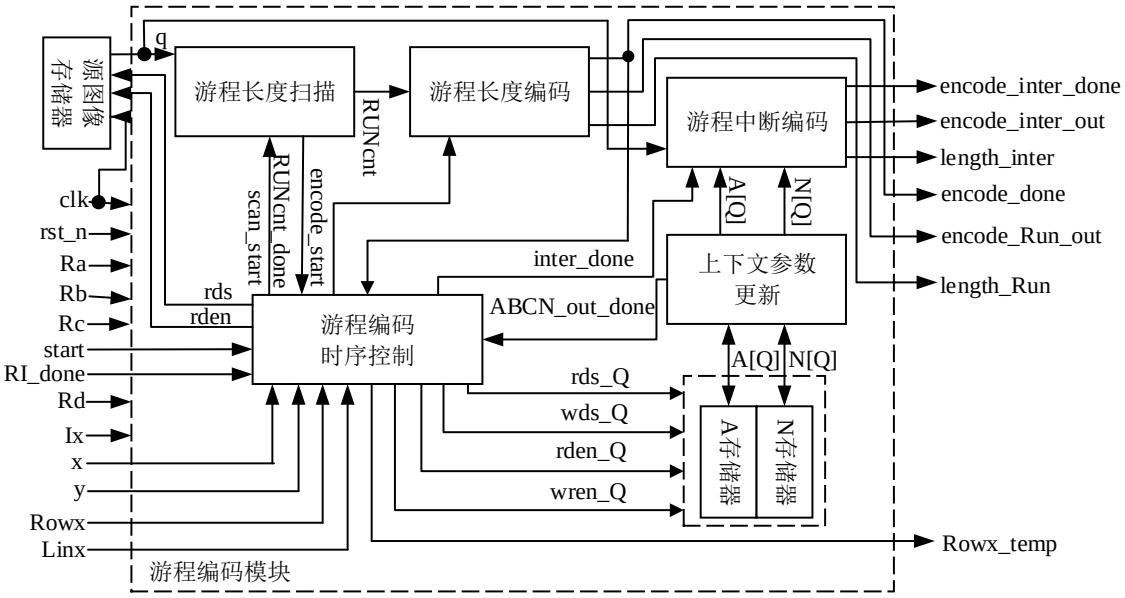


图 4-24 游程编码模块的结构框图

表 4.4 游程编码模块基本信号列表

信号名	位宽	接口状态	备注
Rowx	9	输入	当前待编码像素所处的列
Linx	9	输入	当前待编码像素所处的行
RUNcnt	9	中间变量	游程长度
RUNcnt_done	1	中间变量	游程长度扫描模块完成标志位
inter_done	1	中间变量	游程中断模块选通信号
Rowx_temp	9	输出	游程中断编码完成后像素所处的列

如图 4-25 所示，为本文设计的 JPEG-LS 编码器游程编码模块的时序图。其中，游程长度扫描模块与游程长度编码模块由于无法确定待编码像素与基准像素是否相等，导致游程长度不确定，进而导致游程长度扫描所需的时钟个数以及游程长度编码所需的时钟个数均为不确定值，图中所示“至少 3 个 clk”为 RUNcnt=0 的情况。

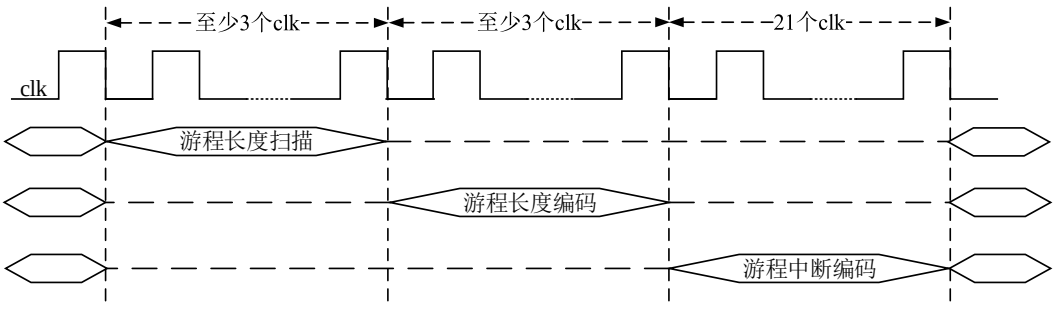


图 4-25 游程编码模块的时序图

万方数据

4.4.1 游程长度扫描

游程长度扫描模块通过比较器与加法器完成游程长度 RUNcnt 的扫描，并通过比较器完成行末标志位 EOLine 的产生。如图 4-26 所示，为游程长度 RUNcnt 扫描的结构框图。

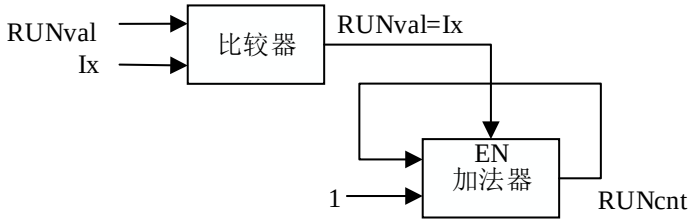


图 4-26 游程长度 RUNcnt 扫描的结构框图

游程长度 RUNcnt 扫描的具体过程为：将进入游程编码模块时当前待编码像素因果模板中的像素 Ra 作为基准像素，并暂存到寄存器 RUNval 中，然后通过比较器判断后续待编码像素 Ix 是否与基准像素相等，一旦相等，就通过加法器对游程长度 RUNcnt 进行加 1 运算，并通过比较器判断当前待编码像素所处的列信号 Rowx 与源图像的宽 y 是否相等，若相等，则将 1 赋值给行末标志位 EOLine，否则行末标志位 EOLine 保持 0 不变，然后继续比较下一个待编码像素，直到出现与基准像素不相等的待编码像素或者扫描到行末像素。

4.4.2 游程长度编码

当游程长度编码模块检测到游程长度扫描模块的完成标志位 RUNcnt_done 为高电平时，开始对游程长度 RUNcnt 进行游程长度编码。如图 4-27 所示，为游程长度编码的结构框图。

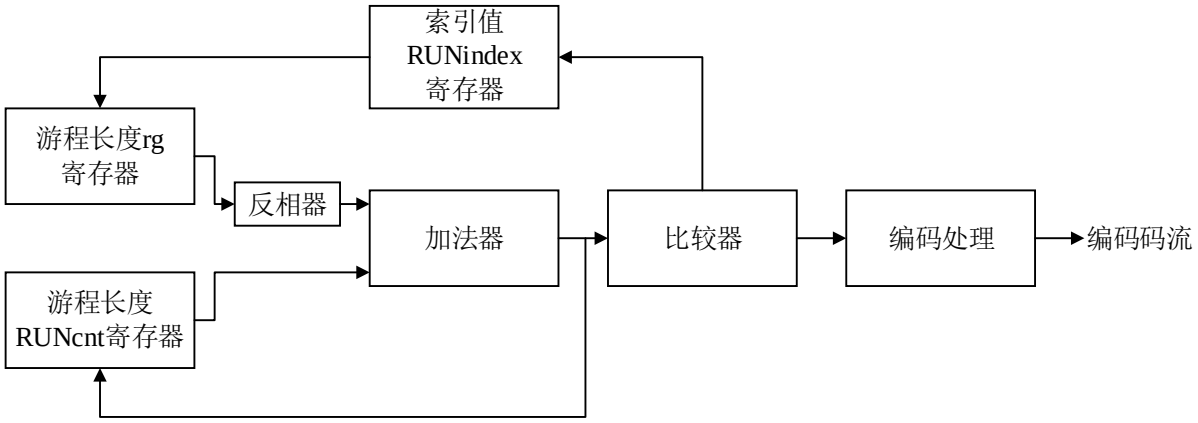


图 4-27 游程长度编码的结构框图

首先对入口向量表 J 所指向的游程长度 rg 和游程长度 RUNcnt 做减法运算，然后将运算结果送入比较器和 0 进行比较，若运算结果大于等于 0，则进行编码处理，同时对 RUNindex 做加 1 运算，游程长度 rg 寄存器、游程长度 RUNcnt 寄存器也随之进行更新，开始新一轮的比较，直到输入比较器中的运算结果小于 0 为止。

如图 4-28 所示，为游程长度编码时序控制的状态转换图。

万方数据

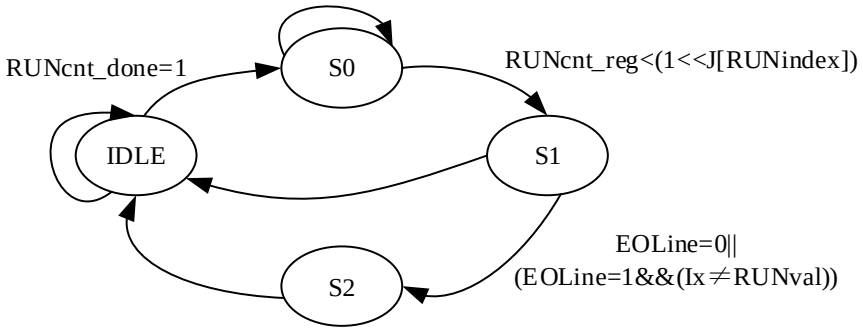


图 4-28 游程长度编码时序控制的状态转换图

状态 IDLE 检测游程长度扫描完成的标志位，若检测到该标志位为高电平，则将游程长度 RUNcnt 暂存到寄存器 RUNcnt_reg 中，将编码码流长度寄存器 length_Run 清零，然后跳转到状态 S0，否则一直处于此状态。

状态 S0 判断条件RUNcnt_reg≥(1<<J[RUNindex])是否成立，若成立，则先对暂存在寄存器 encode_out 中的编码码流进行左移 1 位操作，然后将 1 比特 1 暂存到寄存器 encode_out 的最低位，同时将编码码流长度进行加 1 运算，并暂存到寄存器 length 中；对游程长度进行减(1<<J[RUNindex])运算，并暂存到寄存器 RUNcnt_reg 中；判断条件RUNindex<31是否成立，若成立，则一直处于此状态，否则跳转到状态 S1。

状态 S1 判断游程扫描终止的原因，若是由于扫描到差异像素而终止，则对暂存在寄存器 encode_out 中的编码码流进行左移一位操作，并暂存到寄存器 encode_out 中，同时对暂存在寄存器 length 中的编码码流长度进行加 1 运算，并暂存到寄存器 length 中，然后跳转到状态 S2；若是由于扫描到行末像素而终止，则判断 RUNcnt_reg>0 是否成立，若成立，则先对暂存在寄存器 encode_out 中的编码码流进行左移 1 位操作，并将 1 比特 1 暂存到寄存器 encode_out 的最低位，同时对暂存在寄存器 length 中的编码码流长度进行加 1 运算，并暂存到寄存器 length 中，将编码完成标志位赋值为 1，并暂存到寄存器 encode_done 中，然后跳转到状态 IDLE，否则将编码完成标志位赋值为 1，并暂存到寄存器 encode_done 中，然后跳转到状态 IDLE。

状态 S2 先对暂存在寄存器 encode_out 中的编码码流进行左移 J[RUNindex]位操作，然后将暂存在寄存器 RUNcnt_reg 中的游程长度暂存到编码码流的低 J[RUNindex]位，并暂存到寄存器 encode_out 中；对暂存在寄存器 length 中编码码流长度进行加 J[RUNindex]运算，并暂存到寄存器 length 中；判断RUNindex>0是否成立，若成立，则对暂存在寄存器 RUNindex 中的索引值进行减 1 运算；将编码完成标志位赋值为 1，并暂存到寄存器 encode_done 中，将扫描中断标志位赋值为 1，并暂存到寄存器 inter_done 中；最后跳转到状态 IDLE。

4.4.3 游程中断编码

游程中断编码模块主要包括预测值计算、预测误差计算、预测误差编码以及上下文参数更新等子模块。当游程中断编码模块检测到游程中断标志位 inter_done 为高电平时，对差异像素进行游程中断编码编码。

游程中断编码的具体过程为：游程中断编码模块首先通过比较器比较差异像素 Ix 的邻近像素

万方数据

Ra 和 Rb 的大小来获得游程中断编码的索引值 RItype，若Ra=Rb，则将 1 赋值给 RItype，否则将 0 赋值给 RItype；然后通过比较器判断 RItype 是否为 1，若RItype=1，则将 Ra 赋值给 Px，否则将 Rb 赋值给 Px，完成当前待编码像素预测值 Px 的计算；然后进行预测误差的计算、校正以及模减，预测误差模减完成后进行上下文参数更新；然后将模减后的预测误差 Errval 送入预测误差编码模块，完成 Golomb 编码参数 k 的计算、预测误差的映射以及对映射后的预测误差 MErrval 进行 Golomb 编码；最后将 1 赋值给游程中断编码完成标志位 encode_inter_done。与正常编码模块中上下文更新模块类似，游程中断编码模块在检测到预测误差模减完成的标志位 Errval_done 为高电平时就开始对变量 A[Q]和 N[Q]以及变量 Nn[365]和 Nn[366]进行更新，而不必等到编码完成后再进行上下文参数的更新。如图 4-29 所示，为差异像素预测误差计算的结构框图。

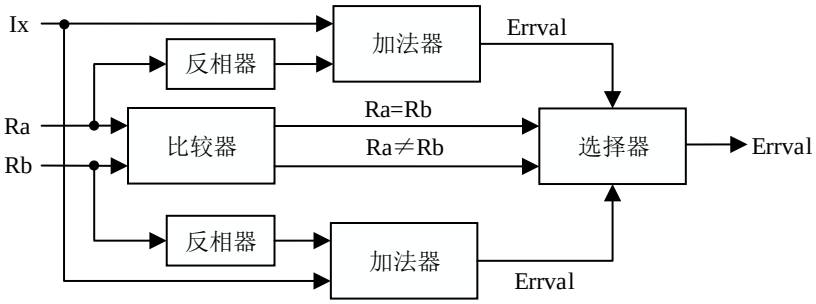


图 4-29 差异像素预测误差计算的结构框图

如图 4-30 所示，为本文设计的 JPEG-LS 编码器游程中断编码模块的时序图。其中，预测误差计算模块包括 1 个 clk 的预测值计算，“2 至 3 个 clk”是因为无法确定是否对 A[Q]和 N[Q]进行取半操作，“3 至 4 个 clk”是因为 Golomb 编码所需要的时钟个数是固定的。

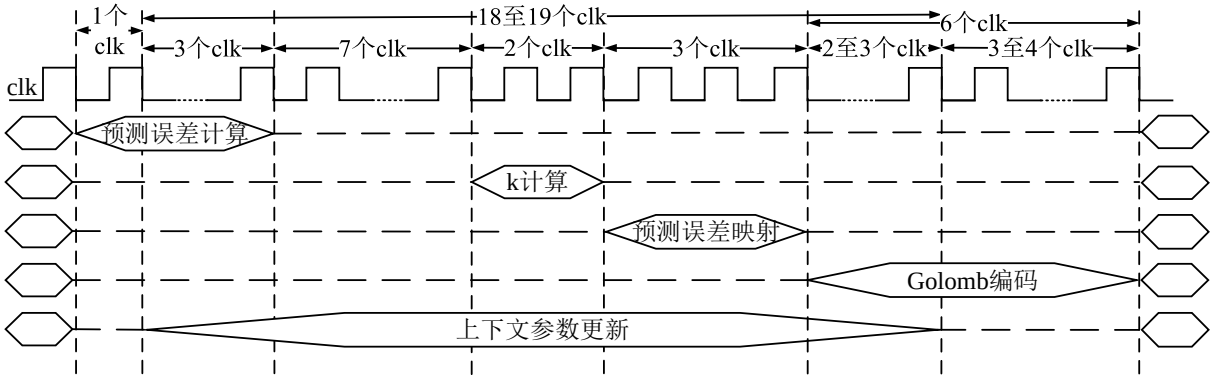


图 4-30 游程中断编码的时序图

4.5 输出模块

输出模块完成上次拼接剩余码流和本次编码码流的拼接，并将拼接后的 32 位编码码流以 8 比特为单位写入到编码码流双口 RAM 中。如图 3-31 所示，为输出模块的结构框图。其中，码流拼接时序控制模块控制上次拼接剩余码流和本次编码码流的拼接过程；输出时序控制模块控制编码码流写入存储器的过程，每当写入 8 比特编码码流后，就判断 32 位编码码流是否全部写入双口 RAM，若未全部写入，就控制移位寄存器进行左移 8 位操作以及加法器对写地址信号 wds 进行加 1 运算，开始下一个 8 比特编码码流的写操作，否则就继续等待下一个拼接完成标志位 Writecode_done 高电平

万方数据

的到来。

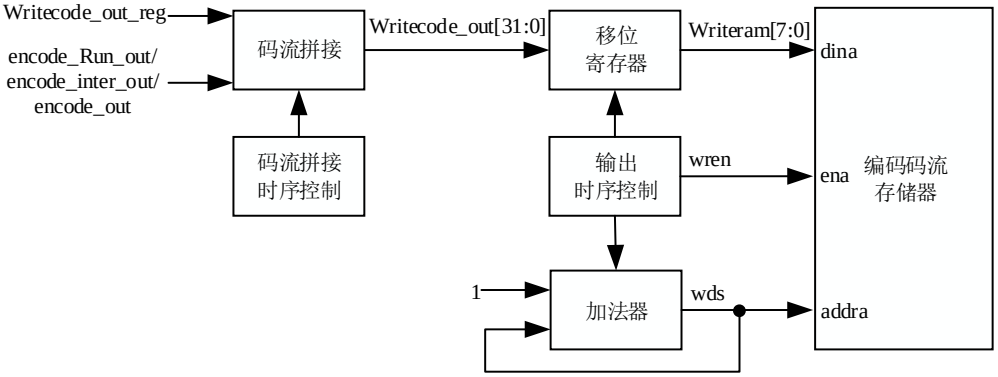


图 4-31 输出模块的结构框图

编码码流的拼接是输出模块的最主要功能。如图 4-32 所示，为码流拼接时序控制的状态转换图。

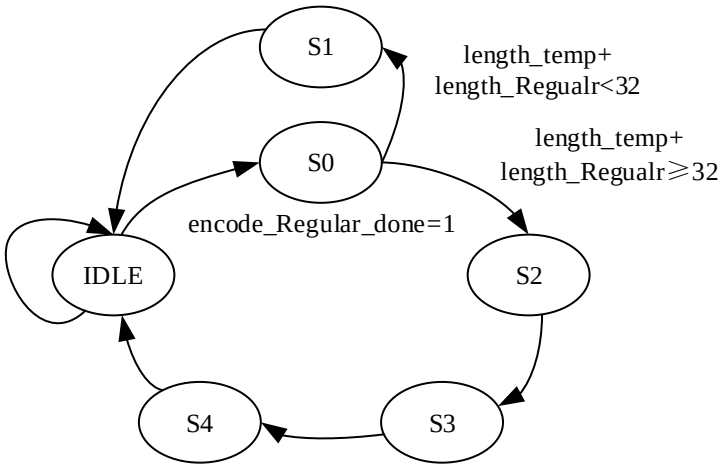


图 4-32 码流拼接时序控制的状态转换图

状态 IDLE 检测编码完成的标志位，若检测到正常编码完成的标志位为高电平，则将正常编码码流暂存到寄存器 encode_out_Regular_reg 中，然后跳转到状态 S0，否则一直处于此状态。由于检测到游程长度编码以及游程中断编码完成的标志位为高电平时的码流拼接情况与检测到正常编码完成的标志位为高电平时相似，故不对检测到游程长度编码以及游程中断编码作阐述。

状态 S0 判断条件length_temp+length_Regular<32是否成立，若成立，则跳转到状态 S1，否则跳转到状态 S2。

状态 S1 先对暂存在寄存器 Writecode_out_reg 中的上次拼接剩余码流进行左移 length_Regular 位操作，然后将暂存在寄存器 encode_out_Regular_reg 中的编码码流暂存到寄存器 Writecode_out_reg 的低 length_Regular 位，对 length_temp 做加 length_Regular 的运算，并暂存到寄存器 length_temp 中，然后跳转到状态 IDLE。

状态 S2 先对暂存在寄存器 encode_out_Regular_reg 中的编码码流做右移 length_temp+length_Regular-32位操作，并暂存到寄存器 encode_out_Regular_reg 中，然后跳转到状态 S3。

状态 S3 先对暂存在寄存器 Writecode_out_reg 中的上次拼接剩余码流进行左移32-length_temp位

万方数据

操作，然后将暂存在寄存器 encode_out_Regular_reg 中的编码码流暂存到寄存器 Writecode_out_reg 的低 32-length_temp 位，然后将码流拼接完成标志位赋值为 1，并暂存到寄存器 Writecode_done 中，然后跳转到状态 S4。

状态 S4 先截取暂存在寄存器 encode_out_Regular_reg 中编码码流的低 length_temp+length_Regular-32 位，并暂存到寄存器 Writecode_out_reg 中；将 length_temp+length_Regular-32暂存到寄存器 length_temp 中；将寄存器 Writecode_done 清零；然后跳转到状态 IDLE。

4.6 本章小结

本章完成了 JPEG-LS 编码器的整体架构及其各个模块进行硬件电路的设计，包括各个模块的结构框图、模块时序控制的有限状态机、状态转换图，并通过 RTL 级设计完成 JPEG-LS 编码器中上下文建模模块、正常编码模块、游程编码模块以及输出模块等的 FPGA 实现。实现了二分查找法对 Golomb 编码参数 k 计算过程进行优化，减小其组合逻辑路径延迟，有效提高了编码器的时钟频率；实现了对上下文因果模板的构建过程进行优化，一方面利用上一个已编码像素因果模板对本次待编码像素因果模板的构建过程进行优化，另一方面在上下文参数更新过程中完成下一个待编码像素因果模板的构建，有效提高了编码器的吞吐率；实现了对上下文建模模块、上下文参数更新模块等的并行化处理，有效提高了编码器的吞吐率。

万方数据

第五章 仿真与验证

本章使用 Mentor 公司的仿真工具 Modelsim 对所设计的 JPEG-LS 编码器及其各个子模块进行功能仿真，并对其进行结果分析；使用搭载 Xilinx Artix-7 系列 XC7A200T 型号的 FPGA 开发板对所设计的 JPEG-LS 编码器进行验证，并对其进行结果分析。

5.1 功能仿真

在 Windows 操作系统下，采用常用的仿真工具 Modelsim 对本文所设计的 JPEG-LS 编码器及其子模块进行功能仿真，以保证 JPEG-LS 编码器及其各个子模块所实现功能的正确性，主要包括上下文建模模块、正常编码模块、游程编码模块以及输出模块等。

5.1.1 上下文建模模块

上下文建模模块主要包括局部梯度值计算、编码模式选择以及局部梯度值量化与合并等子模块。

（1）局部梯度值计算模块

理论上，输入数据为Ix=8’h6e，Ra=8’h0，Rb=8’h0，Rc=8’h0，Rd=8’h0，当因果模板构建完成标志位 RI_done 为高电平时，完成局部梯度值的计算，输出数据为D1=0，D2=0，D3=0；输入数据为Ix=8’h69，Ra=8’h6e，Rb=8’h0，Rc=8’h0，Rd=8’h0，当因果模板构建完成标志位 RI_done 为高电平时，完成局部梯度值的计算，输出数据为D1=0，D2=0，D3=-110。如图 5-1 所示，为局部梯度值计算模块功能仿真图，仿真结果表明局部梯度值计算模块功能正确。

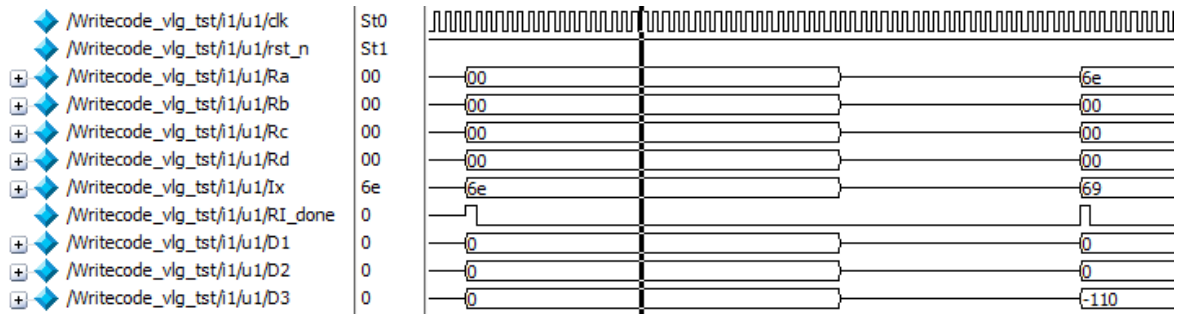


图 5-1 局部梯度值计算模块功能仿真图

（2）编码模式选择模块

理论上，输入数据为D1=D2=D3=0，输出数据为start=0；输入数据为D1=D2=0，D3=-110，输出数据为start=1。如图 5-2 所示，为编码模式选择模块功能仿真图，仿真结果表明编码模式选择模块功能正确。

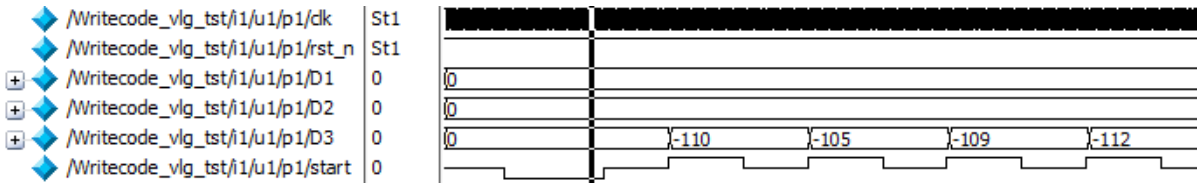


图 5-2 编码模式选择模块功能仿真图

万方数据

（3）局部梯度值量化与合并模块

理论上，输入数据为D1=0，D2=0，D3=-110，输出数据为Q=4，合并完成后标志位 Q_done 为高电平。如图 5-3 所示，为局部梯度值量化与合并模块功能仿真图，仿真结果表明局部梯度值量化与合并模块功能正确。

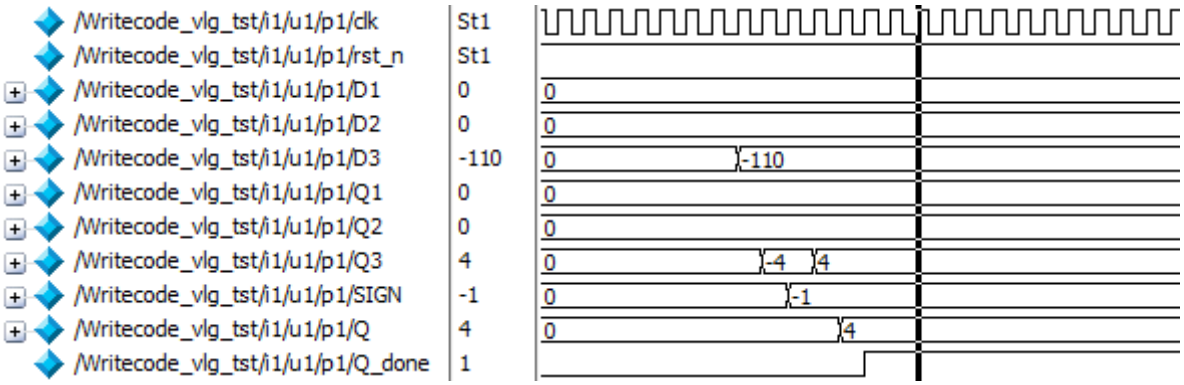


图 5-3 局部梯度值量化与合并模块功能仿真图

5.1.2 正常编码模块

正常编码模块主要包括预测值计算、预测误差计算、预测误差编码以及上下文参数更新等子模块。

（1）预测值计算模块

理论上，输入数据为Ix=8’h69，Ra=8’h6e，Rb=8’h0，Rc=8’h0，Rd=8’h0，SIGN=-1，C=0，当ABCN_out_done 为高电平时，对预测误差进行校正，校正后预测误差 Px_1=110，输出数据为Px_2=110。如图 5-4 所示，为预测值计算模块功能仿真图，仿真结果表明预测值计算模块功能正确。

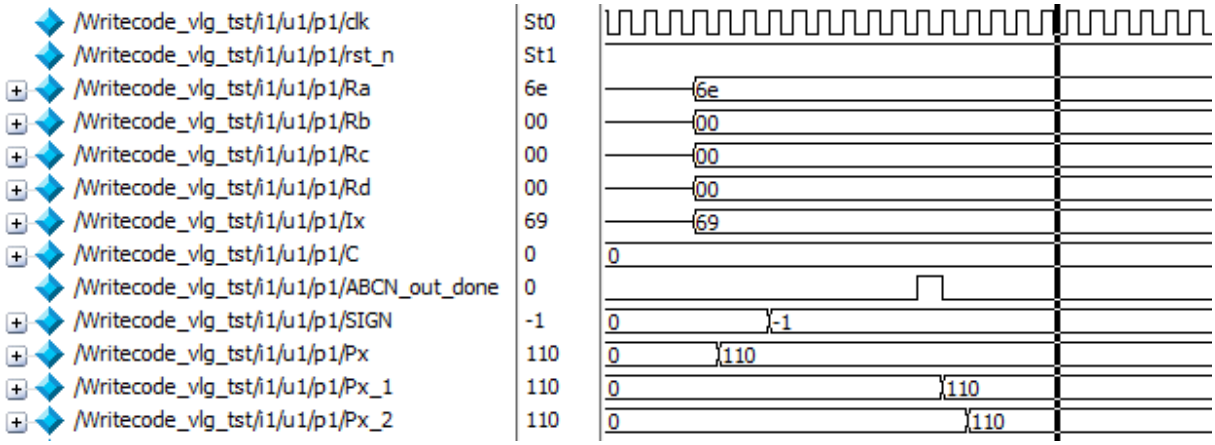


图 5-4 预测值计算模块功能仿真图

（2）预测误差计算模块

理论上，当ABCN_out_done为高电平时，开始预测误差的校正。输入数据为Ix=8’h69，Px_2=110，SIGN=-1，输出数据为Errval_2=5，预测误差模减完成后标志位 Errval_done 为高电平。如图 5-5 所示，为预测误差计算模块功能仿真图，仿真结果表明预测误差计算模块功能正确。

万方数据

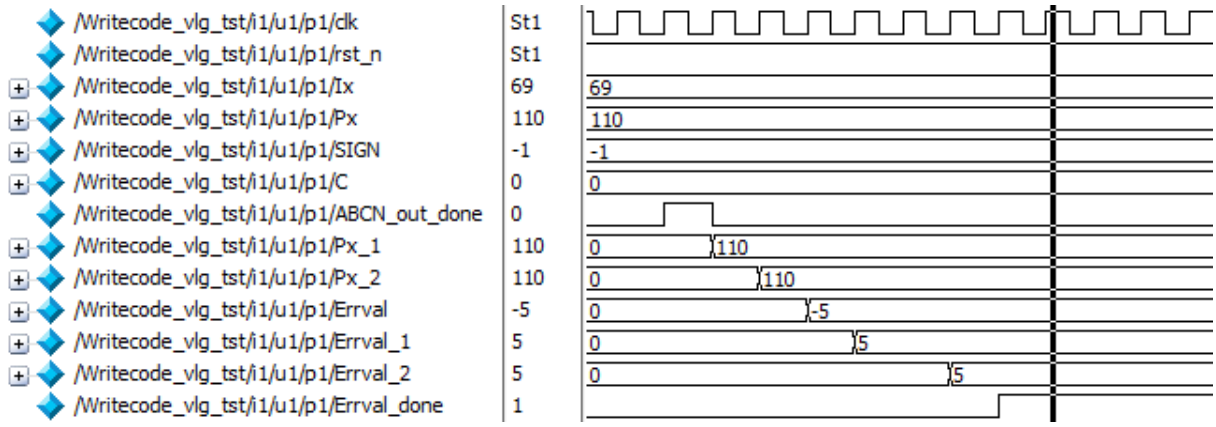


图 5-5 预测误差计算模块功能仿真图

（3）预测误差编码模块

理论上，当 ABCN_out_done 为高电平时，开始预测误差编码。输入数据为N1=1，A1=4，输出数据为k=2，Golomb 编码参数 k 计算完成后标志位 k_done 为高电平。如图 5-6 所示，为 Golomb 编码参数 k 计算模块功能仿真图，仿真结果表明 Golomb 编码参数 k 计算模块功能正确。

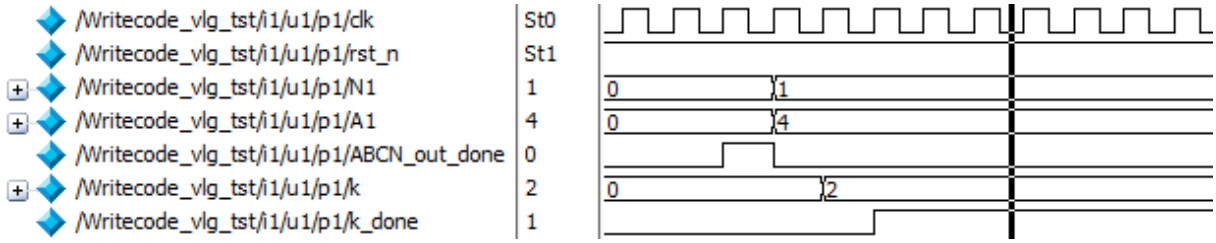


图 5-6 Golomb 编码参数 k 计算模块功能仿真图

理论上，当 k_done 为高电平时，开始预测误差的映射。输入数据为Errval_2=5，B2=0，N2=1，k=2，输出数据为MErrval=10，预测误差映射完成后标志位 MErrval_done 为高电平。如图 5-7 所示，为预测误差映射模块功能仿真图，仿真结果表明预测误差映射模块功能正确。

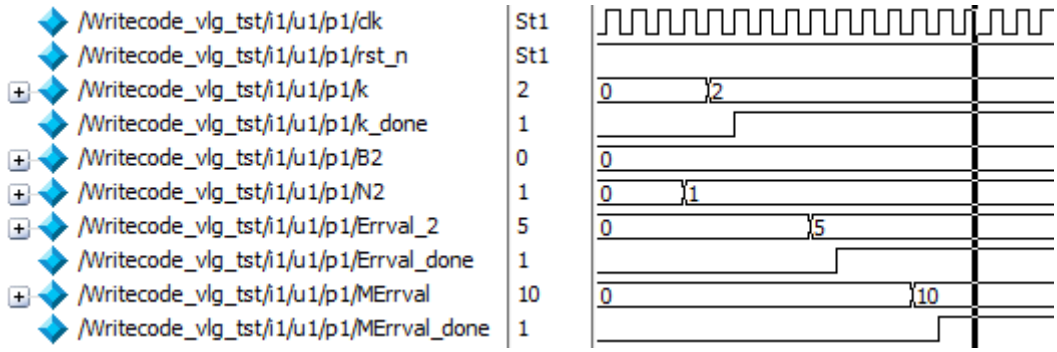


图 5-7 预测误差映射模块功能仿真图

理论上，当 MErrval_done 为高电平时，开始编码。输入数据为k=2，MErrval=10，输出数据为encode_out=32’b0000_0000_0000_0000_0000_0000_0000_0110，length=5，编码完成后标志位 encode_done 为高电平。如图 5-8 所示，为 Golomb 编码模块功能仿真图，仿真结果表明 Golomb 编码模块功能正确。

万方数据

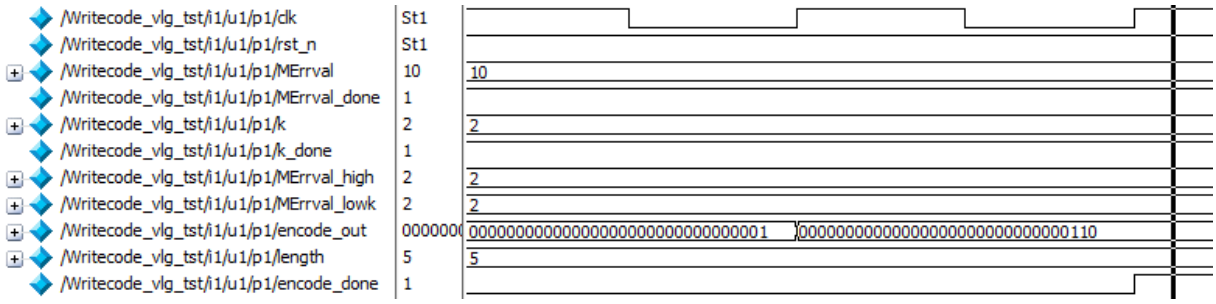


图 5-8 Golomb 编码模块功能仿真图

（4）上下文参数更新模块

理论上，当 Errval_done 为高电平时，开始变量的更新。输入数据为A_1=4，B_1=0，C_1=0，N_1=1，Errval_2=5，Q=4，输出数据为A_1=9，B_1=0，C_1=1，N_1=2，wds=4，wren=1。如图 5-9 所示，为上下文参数更新模块功能仿真图，仿真结果表明上下文参数更新模块功能正确。

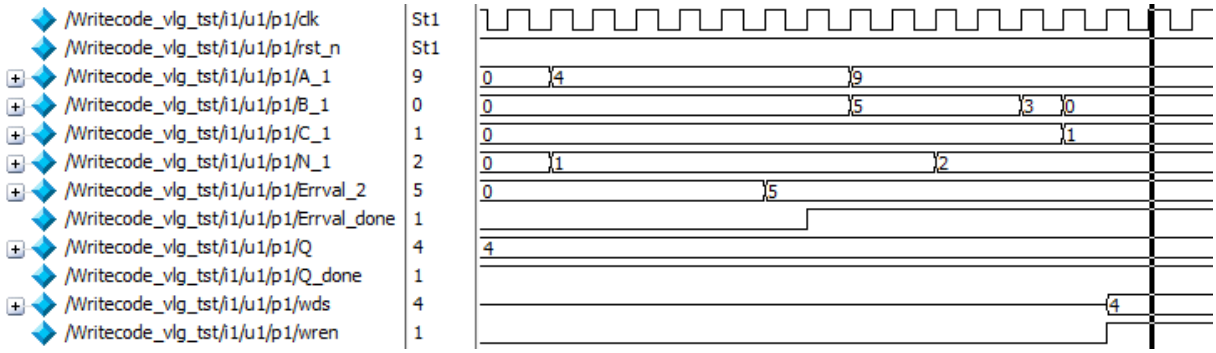


图 5-9 上下文参数更新模块功能仿真图

5.1.3 游程编码模块

游程编码模块主要包括游程长度扫描、游程长度编码以及游程中断编码等子模块。

（1）游程长度扫描模块

理论上，当start=0时，游程长度扫描模块开始游程长度的统计。输入数据为 Ix=8’h6e，RUNval=8’h0，输出数据RUNcnt=0，游程长度扫描完成后标志位 RUNcnt_done 为高电平。如图 5-10 所示，为游程编码模块功能仿真图，仿真结果表明游程编码模块功能正确。

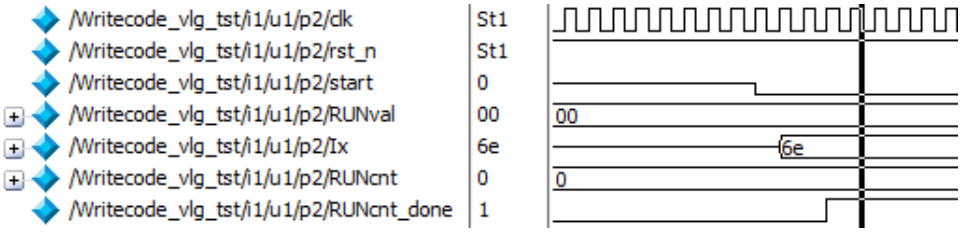


图 5-10 游程编码模块功能仿真图

（2）游程长度编码模块

理论上，当 RUNcnt_done 为高电平时，对游程长度 RUNcnt 进行编码。输入数据为RUNcnt=0，输出数据为encode_out=32’d0，length=1，EOLine=0，游程长度编码完成后标志位 encode_done 产生一个高电平脉冲，游程中断编码开始标志位 inter_done 产生高电平。如图 5-11 所示，为游程长度编码模块功能仿真图，仿真结果表明游程长度编码模块功能正确。

万方数据

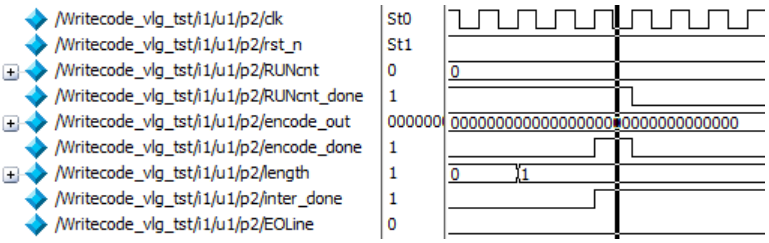


图 5-11 游程长度编码模块功能仿真图

（3）游程中断编码模块

理论上，当 inter_done 为高电平时，游长中断编码模块开始对差异像素进行编码。输入数据为 Ix=8’h6e，Ra=0，Rb=0，输出数据为 encode_inter_out=32’b0000_0000_0000_0000_0000_0001_1101_1010，length1=31，编码完成标志位 encode_inter_done 产生一个高电平脉冲。如图 5-12 所示，为游程中断编码模块功能仿真图，仿真结果表明游程中断编码模块功能正确。

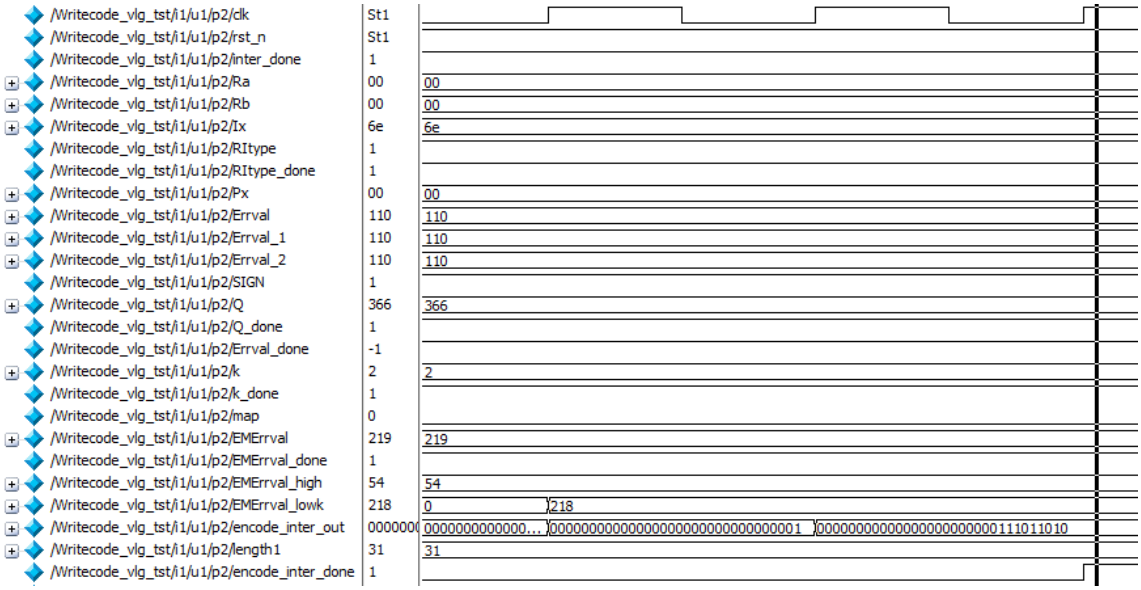


图 5-12 游程中断编码模块功能仿真图

理论上，当 Errval_done 为高电平时，对变量 A 和 N 进行更新。输入数据为A_reg1=4，N1_reg=0，EMErrval=219，Q=366，输出数据为A_reg2=113，N1_reg=2，wds_A=366，wds_N=366，wren_A=1，wren_N=1。如图 5-13 所示，为上下文参数更新模块功能仿真图，仿真结果表明上下文参数更新模块功能正确。

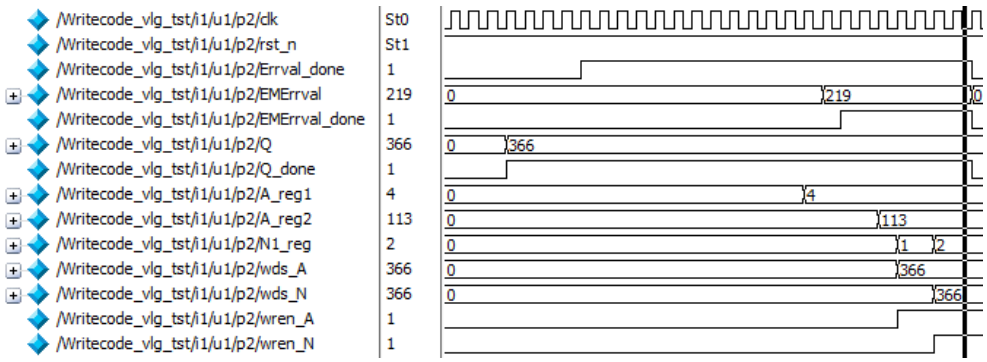


图 5-13 上下文参数更新模块功能仿真图

万方数据

5.1.4 输出模块

理论上，当检测到编码完成标志位 encode_Run_done 或者 encode_Regular_done 或者 encode_inter_done 为高电平时，进行编码码流的拼接。输入数据为encode_out_Run=32'd0，length_Run=1，encode_inter_out=32'b0000_0000_0000_0000_0000_0001_1101_1010，length_inter=31，encode_out_Regular=32'b0000_0000_0000_0000_0000_0000_0000_0110，length_Regular=5，Writecode_out_reg=32'd0，length_temp=0，输出数据为 Writecode_out=32'h00001da，Writecode_out_reg=32'b0000_0000_0000_0000_0000_0000_0000_0110，length_temp=5，码流拼接完成标志位 Writecode_done 产生一个高电平脉冲。如图 5-14 所示，为输出模块码流拼接功能仿真图，仿真结果表明输出模块码流拼接功能正确。

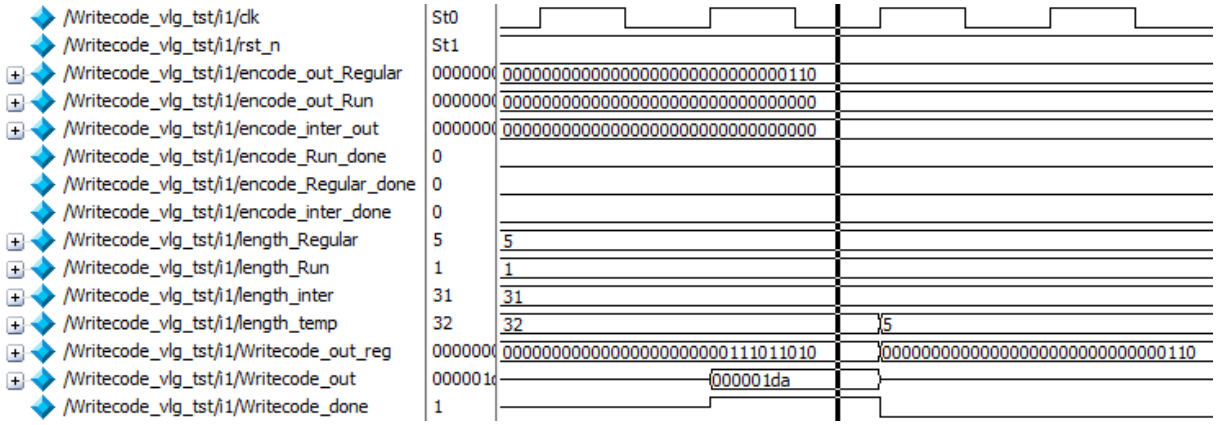


图 5-14 输出模块码流拼接功能仿真图

理论上，当 Writecode_done 为高电平时，对码流进行 RAM 写入。输入数据为 Writecode_out=32'h00001da，中间数据为 8'h00，8'h00，8'h01、8'had，当整幅图像写入 RAM 后，标志位 Writeram_done 为高电平，否则为低电平。如图 5-15 所示，为输出模块编码码流写双口 RAM 功能仿真图，仿真结果表明输出模块写双口 RAM 功能正确。

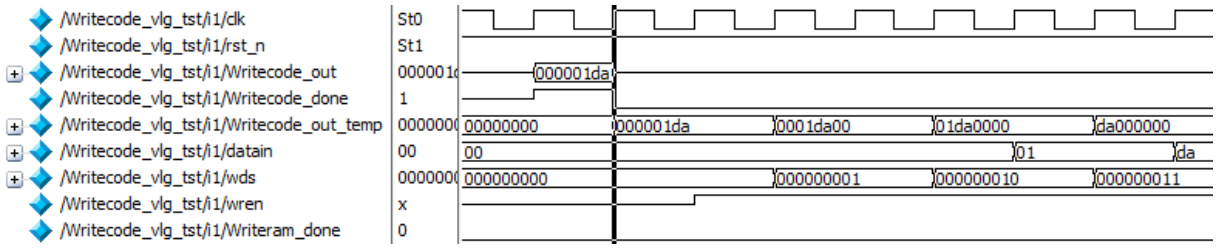


图 5-15 输出模块编码码流写双口 RAM 功能仿真图

5.1.5 整个 JPEG-LS 编码器

在对 JPEG-LS 编码器各个子模块进行功能仿真后，对整个 JPEG-LS 编码器进行功能仿真。为了便于判断 JPEG-LS 编码器编码功能的正确性，将 JPEG-LS 编码器的编码码流写入 result_founction.txt 文本文件中。然后通过文件内容比较工具 Hex Comprasion 来比较 result_founction.txt 与 JPEG-LS 软件压缩结果 result_soft.txt 内容。理论上，result_founction.txt 与 result_soft.txt 内容相同。如图 5-16 所示，为 Hex Comprasion 比较结果图，其中上半区为 result_founction.txt 文件内容，下半区为

万方数据

result_soft.txt 文件内容，比较结果表明本文所设计的 JPEG-LS 编码器的编码功能正确。满足本文预设的功能指标。

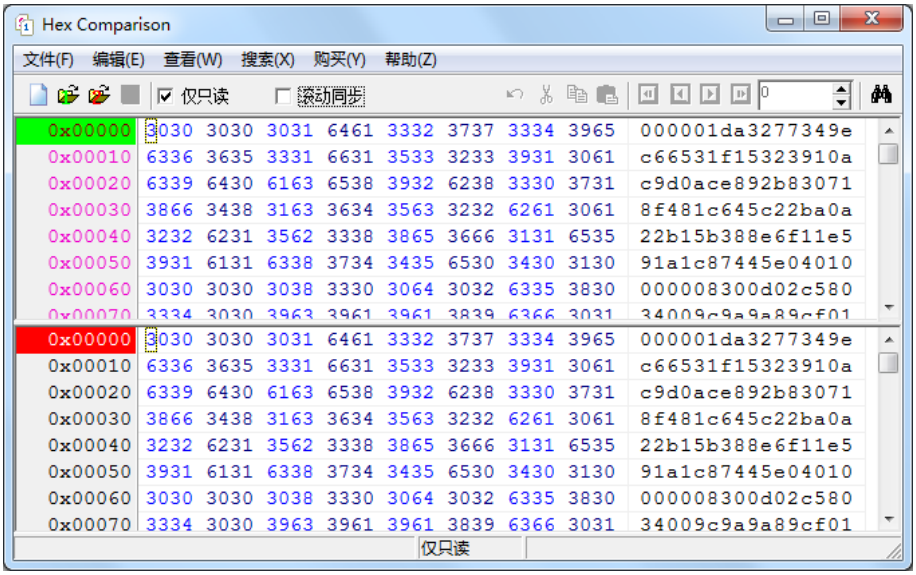


图 5-16 Hex Comprasion 比较结果图

5.2 FPGA 验证

为了确保本文所设计的 JPEG-LS 编码器不出现时序违例，需对其进行静态时序分析，周期约束中时钟频率设为 100MHz；在 JPEG-LS 编码器的 FPGA 实现过程中，其时钟频率设为 100MHz。

5.2.1 综合报告与结果分析

在 ISE 14.7 环境下，基于 Xilinx 的 xc7a200t-2lfbg484 器件完成了验证系统的逻辑综合与布局布线，验证系统包括 JPEG-LS 编码器、UART 接收模块以及 UART 发送模块。如表 5.1 所示，为验证系统的资源利用报告。如图 5-17 所示，为验证系统的时序报告。

表 5.1 验证系统资源利用报告

资源类型	使用量	总量
Slice 中寄存器	1521	269200
Slice 中 LUT	3373	134600
块 RAM	61	730

```
Timing summary:
---------------

Timing errors: 0  Score: 0  (Setup/Max: 0, Hold: 0)

Constraints cover 482665 paths, 0 nets, and 18344 connections

Design statistics:
   Minimum period:   8.686ns{1}   (Maximum frequency: 115.128MHz)
```

图 5-17 验证系统时序报告

由表 5.1 可知，该系统所占用的资源少，Slice 中寄存器、Slice 中 LUT 以及块 RAM 分别仅占总量的 1%、2%以及 8%。由图 5-17 可知，其时钟频率最高能够达到 115.1MHz。结合硬件设计与功能

万方数据

仿真，对一个比特深度为 8 的像素编码所需要的平均时钟个数约为 25，能够计算出本文所设计的 JPEG-LS 编码器的吞吐率约为 36.8Mbps。

为了与以往的研究成果相比较，本文所设计的 JPEG-LS 编码器基于 Virtex-2 系列 XC2V2000 芯片、Virtex-2 系列 XC2V3000 芯片以及 Virtex-4 系列 XC4VLX80 芯片进行了逻辑综合与布局布线。如表 5.2 所示，为本文所设计的编码器与以往研究成果的资源利用对照表。如表 5.3 所示，为本文所设计的编码器与以往研究成果的性能对照表。

表 5.2 本文所设计的 JPEG-LS 编码器与以往研究成果资源利用对照表

芯片型号	比较对象	Slice 使用量	Flip Flop 使用量	LUT 使用量
XC2V2000	文献[48]	2671	-	-
	文献[27]	4759	3360	8417
	本文	3461	1571	6248
XC2V3000	文献[49]	3400	-	-
	本文	3461	1571	6246
XC4VLX80	文献[33]	1967	1503	3417
	本文	3472	1579	6255

表 5.3 本文所设计的 JPEG-LS 编码器与以往研究成果性能对照表

芯片型号	比较对象	时钟频率/MHz	吞吐率/Mbps
XC2V2000	文献[48]	60	22.9
	文献[27]	77.1	-
	本文	83.3	26.7
XC2V3000	文献[49]	40	-
	本文	83.3	26.7
XC4VLX80	文献[33]	41	-
	本文	89.6	28.7

因为文献[27]以及文献[48]中均为未列出芯片的速度等级，所以表格中仅列出了该芯片最低速度等级下编码器的时钟频率，文献[48]中吞吐率是以其处理一个像素所需要的最少时钟个数计算所得。由表 5.2 和表 5.3 可知，与参考文献[48]相比，本文虽然消耗了更多的 Slice 资源，但是还实现了游程长度编码，而且时钟频率以及吞吐率较之有较大的提高；与文献[27]相比，本文不仅消耗了更少的 Slice、Flip Flop 以及 LUT 资源，而且时钟频率较之有一定的提高；与文献[49]相比，本文消耗了与之相当的 Slice 资源，但是时钟频率较之有较大的提高；与文献[33]相比，本文虽然消耗了较多的 Slice、Flip Flop 以及 LUT 资源，但是还实现了游程长度编码，而且时钟频率较之也有一定的提高。在最高的时钟频率上，本文较上述文献有明显的优势，在最高的吞吐率上，本文较文献[48]也有一定的优势，而且本文还实现了文献[48]以及文献[33]中未能实现的游程长度编码。通过对本文所设计编码器所占用的资源、最高时钟频率、最高吞吐率以及 JPEG-LS 标准中无损压缩算法结构完整性等方面的分析，本文所设计的编码器在实际应用中有一定的应用价值，可以更好地实现对灰度图像的无损压缩。

5.2.2 FPGA 验证与结果分析

本文基于 FPGA 对 JPEG-LS 编码器进行了硬件设计，并结合 UART 接口与串口调试助手对所设计的编码器进行了验证。本文 FPGA 实现平台选用搭载 Xilinx Artix-7 系列 XC7A200T 芯片的开发板，

万方数据

开发工具选用 Xilinx ISE 14.7。该款芯片资源丰富，包括 134600 个 Slice LUTs，269200 个 Slice Registers 以及 13140kb Block RAM，足够用来设计整个 JPEG-LS 编码器。如图 5-18 所示，为本文采用的 FPGA 实现平台。

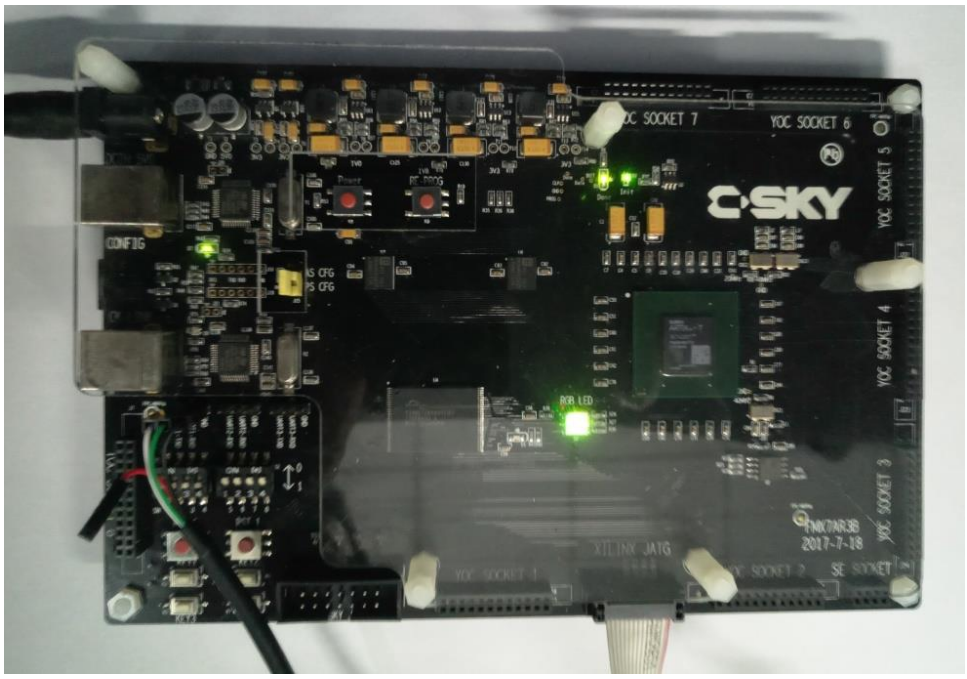


图 5-18 FPGA 实现平台

为了便于对 JPEG-LS 编码器的编码结果进行验证，采用了 UART 接口，并通过 UART 串口调试助手完成 PC 机与 FPGA 开发板间的数据传输。如图 5-19 所示，为本文所设计的 JPEG-LS 编码器 FPGA 验证的结构框图。当 UART 接收模块接收到 PC 机发来的 1 组 32 比特数据时，JPEG-LS 编码器得到源图像的宽和高，开始对源图像进行无损压缩，当编码结束后，UART 发送模块自动按照从低地址位到高地址位的顺序依次将存储在双口 RAM 中的编码码流回传到 PC 机，发送结束即表示一次 FPGA 验证的完成。PC 机接收开发板所发送出的编码码流并在串口调试助手中显示出来，能够直观显示出编码结果。

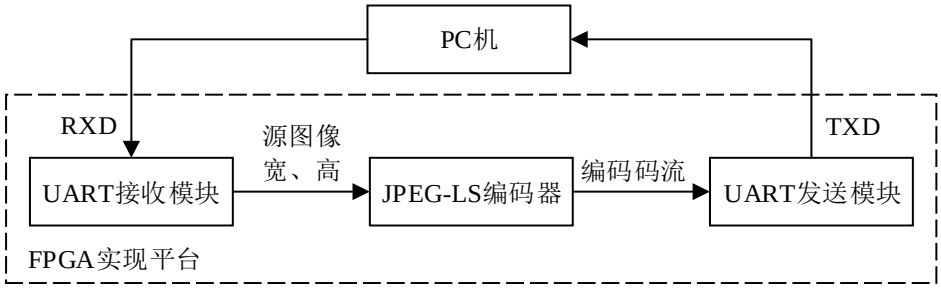


图 5-19 JPEG-LS 编码器 FPGA 验证的结构框图

其中，UART 接收模块接收 PC 机发送的源图像的宽和高信息，UART 发送模块向 PC 机发送 JPEG-LS 编码器的编码码流。

如图 5-20 所示，为本文所设计的 JPEG-LS 编码器的 FPGA 板级验证结果。其中，发送区显示的十六进制数据 00 01 00 01 指的是源图像的宽和高分别为 256 和 256；接收区显示的 00 00 01 DA 等十六进制数据指的是编码码流。

万方数据

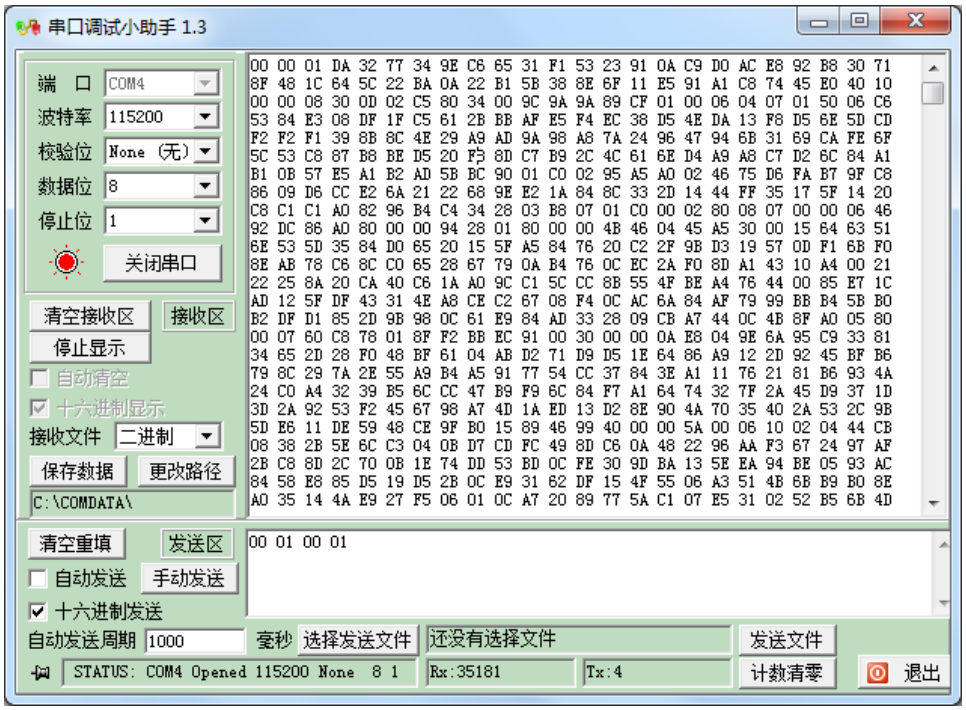
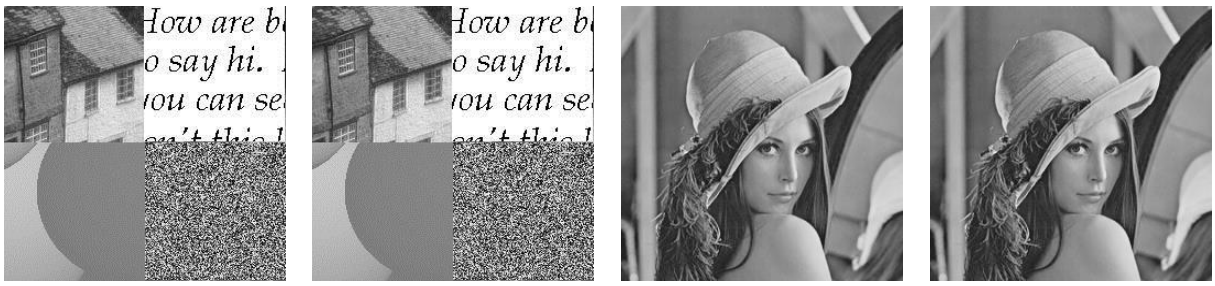


图 5-20 FPGA 板级验证结果

在编码码流发送结束后，将回传到 PC 机的编码码流保存为二进制文本文件 result_FPGA.txt，然后使用 Hex Comparison 比较 result_FPGA.txt 和 result_soft.txt 内容，比较结果表明本文所设计的 JPEG-LS 编码器能够完成对灰度图像的无损压缩；然后通过 JPEG-LS 解压软件对格式转换后的编码码流文件进行解压，然后同样使用 Hex Comparison 对解压后的图像数据和源图像数据进行比较，比较结果表明本文所设计的 JPEG-LS 编码器编码后的图像数据能够还原为源图像数据。满足本文预设的功能指标。

本文对 JPEG-LS 标准中所规定的测试图像集合中的 3 幅比特深度为 8、尺寸为 256*256 的灰度图像 TEST8R.PGM、TEST8G.PGM、TEST8B.PGM 以及 10 幅常用的比特深度为 8、尺寸为 256*256 的灰度图像 lena.bmp、hploco.bmp、bank.bmp、fingerprint.bmp、finger.bmp、couple.bmp、beth.bmp、cameraman.bmp、barbara.bmp、pepper.bmp 进行了 JPEG-LS 编码器的软硬件测试。其中，软件测试平台为 intel Pentium(R) Dual-Core CPU，主频为 2.6GHz。图 5-21（a）以及 5-21（c）分别为 TEST8R.PGM 以及 lena.bmp 的源图像，图 5-21（b）以及 5-21（d）分别为 FPGA 压缩后经 JPEG-LS 软件解压的图像。表 5.4 为 JPEG-LS 软硬件压缩的比较结果。



(a) (b) (c) (d)

图 5-21 TEST8R.PGM 和 lena.bmp 源图像与解压后图像

万方数据

表 5.4 软硬件压缩结果比较结果

图像名称	源图像大小/KB	软件压缩大小/KB	本文压缩大小/KB	软件压缩时间/s	硬件压缩时间/s
TEST8R.PGM	64	33	33	0.031	0.013
TEST8G.PGM	64	34	34	0.031	0.013
TEST8B.PGM	64	34	34	0.032	0.014
lena.bmp	65	36	36	0.034	0.016
hploco.bmp	65	35	35	0.033	0.014
bank.bmp	65	36	36	0.034	0.015
fingerprint.bmp	65	33	33	0.032	0.013
finger.bmp	65	36	36	0.035	0.017
couple.bmp	65	36	36	0.034	0.016
beth.bmp	65	35	35	0.033	0.014
cameraman.bmp	65	34	34	0.032	0.013
barbara.bmp	65	35	35	0.033	0.015
pepper.bmp	65	34	34	0.032	0.014

由表 5.4 可知，本文所设计的 JPEG-LS 编码器对 13 幅测试图像的平均压缩比约为 1.9:1，与软件压缩比一致，满足本文预设的性能指标；硬件压缩速度约为软件的 2 倍以上。由于图像 TEST8R.PGM 的平滑性较好，落入平坦区域的像素点较多，即进入游程编码模式的像素较多，因此图像的压缩比较高，压缩时间较短；与之相比，lena.bmp 的平滑性较差，落入平坦区域的像素点较少，即进入游程编码模式的像素较少，因此图像的压缩比较低，压缩时间较长。其他图像同样如此，因为对落入平坦区域的像素进行的是游程长度编码，所以图像越平坦，其压缩比越高，压缩时间越短。

5.2.3 设计指标总结

通过对本文所设计的 JPEG-LS 编码器进行 Modelsim 功能仿真以及 Xilinx FPGA 验证，现将本文所设计的 JPEG-LS 编码器实际测试结果与设计指标进行比较，其功能完成情况如表 5.5 所示，性能完成情况如表 5.6 所示。

表 5.5 实际功能结果与设计指标对照表

设计指标		实际结果
功能指标	上下文建模模块、正常编码模块、游程编码模块以及输出模块等的 FPGA 实现	功能正确
	实现对比特深度为 8 的灰度图像的无损压缩，并能通过 JPEG-LS 软件解压	功能正确

表 5.6 实际性能结果与设计指标对照表

设计指标		实际结果
性能指标	压缩比约为 2:1，与软件压缩比一致 时钟频率不低于 80MHz，吞吐率不低于 25Mbps	压缩比约为 1.9:1，与软件压缩比一致 最高时钟频率达到 115.1MHz，吞吐率为 36.8Mbps

5.3 本章小结

本章对所设计的 JPEG-LS 编码器进行了仿真与验证，首先对编码器的整体及其各个子模块进行了功能仿真，以保证所有模块功能的正确性，结果表明编码器的所有模块均通过了 Modelsim 的功能

万方数据

仿真；然后在 ISE 14.7 开发环境下对编码器进行了逻辑综合与布局布线，并采用搭载 Xilinx Airtex-7 系列芯片的 FPGA 开发板进行了验证，结果表明编码器能够实现对灰度图像的无损压缩，最高时钟频率为 115.1MHz，吞吐率为 36.8Mbps，压缩比约为 1.9:1，满足预期的设计指标；最后在 ISE 10.1 开发环境下基于 Xilinx 的 Virtex-2 系列以及 Virtex-4 系列芯片对编码器进行了逻辑综合与布局布线，最高时钟频率分别为 83.3MHz 以及 89.6MHz，吞吐率分别为 26.7Mbps 以及 28.7Mbps，结果表明本文所设计的编码器在时钟频率以及吞吐率上较以往的研究成果有不同程度的提高。

万方数据

第六章 总结与展望

6.1 总结

随着图像传感器分辨率的不断提高，需要对大量图像数据进行实时处理，这将给整个系统的存储容量以及传输带宽来极大的挑战。由于数字图像通常包含大量的冗余信息，因此通常会对其进行压缩以消除冗余，减小系统的存储容量和传输带宽。在众多图像压缩的国际标准中，JPEG-LS 由于其优越的压缩性能被广泛应用。JPEG-LS 具有无损压缩和近无损压缩两种功能，其中无损压缩在医学影像、生物特征、卫星遥感等对图像质量要求较高的领域具有重要的应用价值。本文完成了 JPEG-LS 无损压缩算法的硬件设计与 FPGA 实现，主要工作如下。

（1）在对 JPEG-LS 无损压缩标准研究基础上，对 JPEG-LS 算法进行了优化，使其适合硬件电路的并行化处理，并给出了硬件设计方案。

（2）完成上下文建模模块、正常编码模块、游程编码模块以及输出模块等的 RTL 级设计，并将其集成为完整的 JPEG-LS 编码器。为提高所设计 JPEG-LS 编码器的时钟频率以及吞吐率，在硬件设计与实现过程中采用了多种方法对其进行优化。对上下文因果模板的构建过程进行优化，一方面利用上一个已编码像素因果模板对本次待编码像素因果模板的构建过程进行优化，另一方面在上下文参数更新过程中完成下一个待编码像素因果模板的构建；利用二分查找法对 Golomb 编码参数 k 的计算过程进行优化。

（3)通过 Modelsim 对本文所设计的 JPEG-LS 编码器及其各个子模块进行功能仿真，在 ISE 14.7 开发环境下对设计进行了综合，给出资源利用报告和时序报告，并对其进行结果分析；本文所设计的 JPEG-LS 编码器在 FPGA 平台上实现，并对其进行结果分析。

6.2 展望

本文完成的基于 FPGA 的 JPEG-LS 无损压缩算法的硬件设计具有一定的应用价值，所设计编码器的最高时钟频率达到 115.1MHz，吞吐率为 36.8Mbps，压缩比为 1.9:1，但是仍然有不少方面值得改进。

（1）本文所设计的 JPEG-LS 编码器只实现了 JPEG-LS 标准中的无损压缩功能，但是标准中的近无损压缩功能也有广泛的应用。为了进一步提高图像的压缩效率，对于一些对图像质量要求不高的应用场合，可以采用近无损压缩。

（2）对于非平坦区域较少的图像来说，游程长度很小，游程长度编码的意义不大。因此，在允许图像的压缩比以及压缩时间有一定牺牲的情况下，在 JPEG-LS 编码器的硬件设计过程中可以舍弃游程编码模块，能够减少较多的硬件资源消耗。

万方数据

万方数据

参考文献

[1] Deng L, Huang Z. The FPGA design of JPEG-LS image lossless decompression IP core[C]//2015 Chinese Automation Congress (CAC). IEEE, 2015: 2199-2203.

[2] Kabir M A, Mondal M R H. Edge-based transformation and entropy coding for lossless image compression[C]//2017 International Conference on Electrical, Computer and Communication Engineering (ECCE). IEEE, 2017: 717-722.

[3] Amashi R, Baligar V P, Kalwad P. Experimental study on JPEG-LS algorithm[C]//2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI). IEEE, 2017: 1521-1527.

[4] Haddad S, Coatrieux G, Cozic M, et al. Joint watermarking and lossless JPEG-LS compression for medical image security[J]. Irbm, 2017, 38(4): 198-206.

[5] Thärnå J, Nilsson K, Bigun J. Orientation scanning to improve lossless compression of fingerprint images[C]//International Conference on Audio-and Video-Based Biometric Person Authentication. Springer, Berlin, Heidelberg, 2003: 343-350.

[6] 吴美建, 林行刚. 一种改进的遥感图象准无损压缩 JPEG-LS 算法[J]. 中国图象图形学报, 2003, 8(5): 596-600.

[7] Weinberger M J, Seroussi G, Sapiro G. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS[J]. IEEE Transactions on Image processing, 2000, 9(8): 1309-1324.

[8] Ryan P, Connell J. Real-time lossless image compression in a hardware environment[J]. 2005.

[9] Shalev O, Schveiger L, Sutskover I, et al. A modified JPEG-LS image coder with increased channel error robustness[C]//21st IEEE Convention of the Electrical and Electronic Engineers in Israel. Proceedings (Cat. No. 00EX377). IEEE, 2000: 245-248.

[10] ZHANG T, Zou S, ZENG Y. Fpga-based implementation of image lossless compression[J]. Systems Engineering and Electronics, 2004, 10.

[11] Niu H, Shang Y, Yang X, et al. Design and research on the JPEG-LS image compression algorithm[C]//2010 Second International Conference on Communication Systems, Networks and Applications. IEEE, 2010, 1: 232-234.

[12] Pavlidis G, Tsompanopoulos A, Papamarkos N, et al. A multi-segment image coding and transmission scheme[J]. Signal processing, 2005, 85(9): 1827-1844.

[13] Rane S D, Sapiro G. Evaluation of JPEG-LS, the new lossless and controlled-lossy still image compression standard, for compression of high-resolution elevation data[J]. IEEE Transactions on Geoscience and Remote sensing, 2001, 39(10): 2298-2306.

[14] Klimesh M, Stanton V, Watola D. Hardware implementation of a lossless image compression algorithm using a field programmable gate array[J]. Mars (Pathfinder), 2001, 4(4.69): 5.72.

[15] Merlino P, Abramo A. A fully pipelined architecture for the LOCO-I compression algorithm[J]. IEEE Transactions on very large scale integration (VLSI) Systems, 2009, 17(7): 967-971.

[16] Daryanavard H, Abbasi O, Talebi R. FPGA implementation of JPEG-LS compression algorithm for real time applications[C]//2011 19th Iranian Conference on Electrical Engineering. IEEE, 2011: 1-4.

[17] Kim B S, Baek S, Kim D S, et al. A high performance fully pipeline JPEG-LS encoder for lossless compression[J]. IEICE Electronics Express, 2013: 10.20130348.

[18] Mert Y M. FPGA-based JPEG-LS encoder for onboard real-time lossless image compression[C]//Satellite Data Compression, Communications, and Processing XI. International Society for Optics and Photonics, 2015, 9501: 950106.

[19] Nazar F, Murugan S. Implementation of JPEG-LS compression algorithm for real time applications[C]//2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). IEEE, 2016: 2772-2774.

[20] Bhimani D, Chaurasiya P N, Sedani B. Optimized lossless image compression algorithm LOCO-I for small images[C]//2016 Conference on Advances in Signal Processing (CASP). IEEE, 2016: 223-225.

[21] Chen S L, Liu T Y, Shen C W, et al. VLSI implementation of a cost-efficient near-lossless CFA image compressor for wireless capsule endoscopy[J]. IEEE Access, 2016, 4: 10235-10245.

[22] Jallouli S, Zouari S, Masmoudi N, et al. An Adaptive Block-Based Histogram Packing for Improving the Compression Performance of JPEG-LS for Images with Sparse and Locally Sparse Histograms[C]//International Conference on Image and Signal Processing. Springer, Cham, 2018: 63-71.

[23] Haddad S, Coatrieux G, Cozic M, et al. Joint watermarking and lossless JPEG-LS compression for medical image security[J]. Irbm, 2017, 38(4): 198-206.

万方数据

[24] 刘波, 姜宏旭, 练有品, 逯进元. 机载图像无损/近无损压缩方案及其 FPGA 实现[J]. 北京航空航天大学学报, 2006(12): 1443-1446+1467.
[25] 韩俊萍, 程永强, 戴鑫. JPEG-LS 图像无损压缩的 IP 固核设计[J]. 太原理工大学学报, 2010, 41(6): 713-716.
[26] 陈军, 王怀超, 顾晓东, 陈晓敏. 基于 LOCO-I 算法的星载图像无损压缩的 FPGA 实现[J]. 微电子学与计算机, 2011, 28(11): 169-173.
[27] 王舒瑶. 基于 JPEG-LS 标准的图像编码器硬件系统设计[D]. 西安电子科技大学, 2013.
[28] 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.
[29] 聂永康, 雷杰, 李云松, 宋长贺, 吴宪云. 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, 2016, 43(04): 75-80.
[30] 范文晶, 王召利, 王惠娟, 费聚锋, 李萧萧. 基于 FPGA 的无损图像压缩算法实现[J]. 电子科技, 2016, 29(11): 126-128+132.
[31] 蒙红英, 柴昱洲, 韩宇. 一种基于稀疏表示的JPEG-LS改进算法[J]. 第四届高分辨率对地观测学术年会论文集, 2017.
[32] 陈聪, 张学全, 周盛雨. JPEG-LS 码率控制算法改进及硬件实现[J]. 空间科学学报, 2018, 38(06): 112-120.
[33] 李长兴. 基于 FPGA 实现 JPEG-LS 无损压缩算法的研究[D]. 沈阳航空航天大学, 2018.
[34] 曹健. 一种无损 JPEG 压缩电路的设计[D]. 东南大学, 2016.
[35] Papadonikolakis M, Pantazis V, Kakarountas A P. Efficient high-performance ASIC implementation of JPEG-LS encoder[C]//2007 Design, Automation & Test in Europe Conference & Exhibition. IEEE, 2007: 1-6.
[36] Wang Z, Klaiber M, Gera Y, et al. Fast lossless image compression with 2D Golomb parameter adaptation based on JPEG-LS[C]//2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO). IEEE, 2012: 1920-1924.
[37] 张雪松, 倪国强, 周立伟, 金伟其. 图像编码技术发展综述[J]. 光学技术, 1997, 3: 39-41.
[38] Huang Y, Rao R P N. Predictive coding[J]. Wiley Interdisciplinary Reviews: Cognitive Science, 2011, 2(5): 580-593.
[39] 冯希. 几种图像无损压缩与编码方法的比较研究[D]. 中国科学院研究生院（西安光学精密机械研究所）, 2008.
[40] Nazar F, Murugan S. Implementation of JPEG-LS compression algorithm for real time applications[C]//2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). IEEE, 2016: 2772-2774.
[41] Hudson G, Léger A, Niss B, et al. Jpeg at 25: Still going strong[J]. IEEE MultiMedia, 2017, 24(2): 96-103.
[42] ISO I S, JTC I E C. Information technology-JPEG 2000 image coding system-Part 1: Core coding system[J]. ISO/IEC 15444-1, 2001.
[43] 臧景峰, 崔春姬. 静态图像压缩—从 JPEG 到 JPEG—2000[J]. 长春理工大学学报(自然科学版), 2002, 25(3): 34-36.
[44] Weinberger M J, Seroussi G, Sapiro G. LOCO-I: A low complexity, context-based, lossless image compression algorithm[C]//Proceedings of Data Compression Conference-DCC'96. IEEE, 1996: 140-149.
[45] Wang Z, Simoncelli E P, Bovik A C. Multiscale structural similarity for image quality assessment[C]//The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003. Ieee, 2003, 2: 1398-1402.
[46] 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.
[47] Merhav N, Seroussi G, Weinberger M J. Optimal prefix codes for sources with two-sided geometric distributions[J]. IEEE Transactions on Information Theory, 2000, 46(1): 121-135.
[48] 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.
[49] 郝勇峥. 基于 JPEG-LS 算法的星载图像压缩系统设计[D]. 西安电子科技大学, 2011.

万方数据

致谢

时光流转，三年的研究生生涯也即将结束。回首往昔，昨天的点点滴滴又涌上心头。在这里我向所有支持、鼓励、帮助过我的人致以衷心的感谢。

首先，感谢我的导师徐平平教授和李冰教授。徐平平教授博学多才、严谨务实、温文尔雅、诲人不倦，是我学习的好榜样。李冰教授学识渊博、治学严谨、平易近人，从生活到学习都给予了我无微不至的关怀、谆谆善诱的指导。两位老师一丝不苟的工作态度、严谨认真的科研态度、积极向上的生活态度都深深地影响着我，为我以后的工作、学习打下了良好的基础。在这儿向两位教授致以最崇高的敬意以及最真挚的感谢。

其次，感谢实验室的博士们以及师兄师姐们，在专业学习、项目研究、工作寻找等方面都给了我莫大的帮助；感谢我亲爱的同学们，我们一起学习，一起努力，互相帮助，互相鼓励，一起走过了一千余个难忘的日日夜夜。正是因为有了他们，三年的学习生活才显得这般充实与踏实。

最后，感谢我的家人对我的默默支持与理解，正是因为有了他们，我才能全身心投入到专业学习与科学研究中。

万方数据

万方数据

攻读硕士学位期间发表的论文

[1] 尤传亮，李冰. JPEG-LS 图像无损压缩算法的 FPGA 实现. 东南大学校庆研究生学术报告会，2018.

万方数据

万方数据