大连理工大学

硕士学位论文

空间图像CCSDS压缩算法研究与FPGA实现

姓名：雷震霖

申请学位级别：硕士

专业：通信与信息系统

指导教师：殷福亮;陈喆

20071214

摘　要

随着空间技术的发展，遥感图像获得了日益广泛的应用，随之带来的遥感数据海量增长，给存储和传输都带来极大的困难，因而进行有效的数据压缩就显得特别迫切和重要。图像压缩编码技术能降低图像冗余度，从而减小图像的存储容量和传输带宽，它的研究对于遥感图像的应用具有重要的现实意义。CCSDS 图像压缩算法是空间数据系统咨询委员会(CCSDS)提出的图像数据压缩算法。该算法复杂度较低，并行性好，适合于硬件实现，能实现对空间数据的实时处理，从而广泛应用于深空探测和近地观测。本文针对 CCSDS 压缩算法进行了研究，本文主要工作如下：

(1) 对 CCSDS 编码系统的基本理论进行了深入研究，并对其编码性能进行了系统分析。总结了各种常用压缩技术的特点和算法，然后以此为参照，对 CCSDS 在遥感图像压缩中的优势进行了讨论。此外，从应用出发，多角度比较、分析了其编码的有效性。这部分的研究为实际应用奠定了必要的理论基础。

(2) 介绍了 Xilinx 公司 Vertex-II 架构的 FPGA 硬件平台，以及所应用的 Verilog HDL 开发语言。描述了 FPGA 开发的流程，并结合流程介绍了集成开发环境 ISE 和仿真软件 ModelSim。

(3) 在给出总体设计方案后，按算法的功能模块给出了 CCSDS 编码器的 FPGA 实现方法和实现性能，详细介绍了位平面编码的实现和优化重点。

(4) 将源代码下载到硬件上并用不同图像测试，在不同压缩等级设置下均通过测试。测试的验证结果表明：基于 FPGA 的 CCSDS 图像压缩算法占用资源较少，并在较高的频率下运行，设计方案在速度和资源利用率方面达到了较好的平衡，达到了预期的设计目的。

关键词：CCSDS；图像压缩算法；位平面编码；FPGA 设计

Study on CCSDS Space Image Compression Algorithm and FPGA Implementation

Abstract

With the development of remote sensing (RS) technique in recent years, the RS images have growing applications in many different fields. Therefore, it's necessary to compress them efficiently because of the rapid increasing of RS data. Because the RS images have the characteristics of large amount of information, low space redundancy and special applications, some conventional compression methods sometimes could not get better results. Therefore, the Consultative Committee for Space Data System put forward the CCSDS image compression algorithm, which is a special for RS application. The algorithm has lower complexities and good parallel architecture, and it has been widely used in deep-space probes and near-earth observatories. Therefore, this dissertation researches in several aspects as follows.

Firstly, the fundamentals of CCSDS are thoroughly researched, and the coding characteristics are systematically analysed. After the characteristics of the comman image compression methods are summarized, the advantages of CCSDS are analyzed. Furthermore, from application point of view, the coding efficiency is compared and analysed variously. These studied provide necessary theory basis for the application.

Secondly, the Xilinx FPGA of Virtex-II architecture and Verilog Hardware Description Language are introduced. Then the flow of FPGA design is described, and the intergrated developing environment ISE and simulation software ModelSim are described step by step.

Then, after the total design is given, the FPGA implementation methods and performance of CCSDS encoder are presented. The Bit Plane Encode module is described in particular and also the key of optimization.

Finally, the source code is downloaded into FPGA. Then the programme is tested by several test-image in different grades of compression, which is tested successfully. The results show that: CCSDS image compression algorithm base on FPGA cost less hardware resources, and can operate under higher frequency. The design achieved a well utilization of resources and speed of state and the expectant design aim is fulfilled.

Key Words: CCSDS; Image Compression Algorithm; Bit Plane Encode; FPGA Design

独创性说明

作者郑重声明：本硕士学位论文是我个人在导师指导下进行的研究工作及取得研究成果。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的研究成果，也不包含为获得大连理工大学或者其他单位的学位或证书所使用过的材料。与我一同工作的同志对本研究所做的贡献均已在论文中做了明确的说明并表示了谢意。

作者签名：[illegible]　日期：2007年12月26日

大连理工大学学位论文版权使用授权书

本学位论文作者及指导教师完全了解“大连理工大学硕士、博士学位论文版权使用规定”，同意大连理工大学保留并向国家有关部门或机构送交学位论文的复印件和电子版，允许论文被查阅和借阅。本人授权大连理工大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，也可采用影印、缩印或扫描等复制手段保存和汇编学位论文。

作者签名：雷震森

导师签名：

2007年12月26日

第一章 绪论

人类自认识自身赖以生存的地球环境、探索地球起源和生命起源以来，经历了漫长的历史时期。在航空高度观测地球只有百年的历史，在宇宙高度观测地球以至遨游太空只有数十年的历程。随着观测高度的增高和观测技术的进步，人类对自然界的认识也经历着从局部、片断逐渐向整体、全面认识的过程发展。空间技术中，遥感(RS)技术集中了空间、电子、光学、计算机通讯等学科的最新成就，作为当代高新技术的一个重要组成部分，其发展速度是始料不及的。经过短短 30 多年的发展，遥感技术已广泛应用于对地观测、大气探测、军事应用、行星探测等广阔的领域，形成了相当大的应用规模。特别是近 10 年来卫星遥感数据市场平均以每年 20%以上的速度增长，已成为继通信卫星后的第二大应用领域，并逐步朝着产业化的方向发展。目前世界上 100 多个国家数千个机构或企业从事或参与卫星遥感及其应用活动。相应的空间图像压缩算法也成为图像压缩的热门之一。

1.1 图像压缩算法的理论依据和现状

图像的压缩编码方法有很多，可以分为经典压缩方法和现代压缩方法两大类。经典压缩方法是指出现在第一代图像压缩编码阶段(1985 年以前)的压缩方法。1948 年，Oliver 提出了第一个编码理论—脉冲编码调制(Pulse Coding Modulation，简称 PCM)；同年，Shannon 的经典论文—“通信的数学原理”首次提出并建立了信息率失真函数概念；1959 年，Shannon 进一步确立了码率失真理论，以上工作奠定了信息编码的理论基础。目前主要编码方法有预测编码、变换编码和统计编码，也称为三大经典编码方法。下面将简要介绍当前常用的图像压缩标准以及其编码方法。

JPEG[1-2]是由国际标准化组织(ISO)提出面向静止图像的编码标准，包括基于 DCT 的有损压缩算法和基于二维 DPCM 的无损压缩算法两个部分，后者不会产生失真，但压缩比很小；前一种算法进行图像压缩时信息虽有损失但压缩比可以很大，例如压缩 20 倍左右时，人眼基本上看不出失真。因此利用它可以获得较高的压缩比，并保持较好的信噪比，从而大大节省图像存储空间，降低通信带宽。

JPEG2000[3-4]是彩色静态图像的新一代编码标准，以前彩色静态图像编码采用 JPEG，二值图像编码采用 JBIG，无损压缩用 JPEG-LS，多种方式同时存在，而 JPEG2000 则将上述方式统一起来，成为对应各种图像的通用编码方式。JPEG2000 放弃了 JPEG 所采用的以离散余弦变换为主的区块编码方式，而改用以离散小波变换为主的多分辨率编码方式。与 JPEG 相比，JPEG2000 的稳定性高，其码流设计可以有效地抑制比特误

码，JPEG2000 格式的图片压缩比可在 JPEG 的基础上再提高 10%～30%，而压缩后的图像显得细腻光滑，同时 JPEG2000 还考虑了人的视觉特性，增加了视觉权重和掩模，在不损害视觉效果的情况下大大提高了压缩效率，在压缩率相同的情况下，比 JPEG 的信噪比提高 30%左右。另外，JPEG2000 可以在不同的区域采用不同的压缩率，实现交互式压缩，即 ROI(Region of Interest，感兴趣区域)，用户可以指定感兴趣区域，在压缩时指定特定的压缩质量，或恢复时指定特定的解压缩要求。JPEG2000 还拥有 5 种渐进层次的编码形式，可在保证空间清晰度和信噪比的前提下实现各种各样的分辨率。最重要的是 JPEG2000 能实现无损压缩，适用于卫星遥感图像、医学图像等重要图像的压缩。JPEG2000 标准融合了许多新的数字图像压缩编码技术，如小波变换、嵌入式块编码等，在网络应用，彩色传真、打印、扫描，移动通信，数字图书馆和电子商务等诸多领域都表现出良好性能。它的主要特性有：低比特率时的良好表现性能、有损和无损压缩、对感兴趣区的编码、码流的随机访问和处理、开放的架构和更高的安全性。

自 Shapiro 1993 年提出了嵌入式零树量化编码 EZW(Embedded Zerotree Wavelet)算法后[1]，许多学者开始利用小波变换系数特有的组织结构特点，提出了一系列基于小波变换的编码方案。Said 和 Pearlman 于 1996 年提出了基于分层树集合分割排序的 SPIHT(Set Partitioning Hierarchical Trees)算法[5]，SPIHT 算法是 EZW 算法的改进，该算法发展了零树的基本思想，采用集合分裂和显著性排序原则，使其在编码效率方面比 EZW 有了很大的提高，同时该算法保留了 EZW 实现简单、码流具有嵌入特性等优点，因此成为零树编码方法中最有代表性的一种方法。

遥感图像的编码同其它数字图像的编码方法相比较有其共同性，即它们都是通过一定的方法来去除数据冗余。但是，遥感图像的编码又有其特殊性。这是因为，大多数遥感图像具有信息量大、空间冗余度低的特点，所以通常的编码方法如 JPEG、DPCM 在应用于遥感图象压缩时都存在一定的局限性，压缩效果不理想。因此针对遥感图像的压缩应用，国际空间数据系统咨询委员会(CCSDS)在 2005 年发布了其推荐标准[6]，其压缩算法在压缩效率和算法复杂度上都有很大的改进。CCSDS 图像压缩算法采用离散小波变换(DWT)对图像数据进行去相关处理，并结合位平面编码技术和熵编码技术对小波变换系数进行压缩编码。它支持有损和无损两种压缩方式以及基于帧扫描和基于带扫描两种扫描方式，能够实现码率可调和图像质量可控，最大支持 16 比特像素深度的图像压缩，能提升科学数据的收集能力，减少系统内存量以及节约遥感勘测的传输带宽。相对于 SPIHT 和 JPEG2000 算法，它的结构易于硬件实现，这也是制定该算法时重点考虑的性能之一[7]。

1.2 FPGA 概述

当今社会是数字化的社会，是数字集成电路广泛应用的社会。数字集成电路本身在不断地进行更新换代，它由早期的电子管、晶体管、小中规模集成电路，发展到超大规模集成电路(VLSIC)以及许多具有特定功能的专用集成电路(ASIC，Application Specific Integrated Circuit)。但是，随着微电子技术的发展，设计与制造集成电路的任务已不完全由半导体厂商来独立承担，系统设计师们更愿意自己设计专用集成电路(ASIC)芯片，而且希望 ASIC 的设计周期尽可能短，最好是在实验室里就能设计出合适的 ASIC 芯片，并且立即投入实际应用之中，因而出现了现场可编程逻辑器件(FPLD，Field Programmable Logic Device)，其中应用最广泛的当属现场可编程门阵列(FPGA, Filed Programmable Gate Array)和复杂可编程逻辑器件(CPLD，Complex Programmable Logic Device)。

自 1985 年 XILINX 公司推出第一片现场可编程逻辑器件(FPGA)至今，FPGA 已经历了二十多年的发展历史。在这二十多年的发展过程中，以 FPGA 为代表的数字系统现场集成技术取得了惊人的发展：现场可编程逻辑器件从最初的 1200 个可利用门，发展到 90 年代的 25 万个可利用门，目前，国际上现场可编程逻辑器件的著名厂商 XILINX 公司、ALTERA 公司又陆续推出了数百万门乃至上千万门的 FPGA 芯片，将现场可编程器件的集成度提高到了一个新的水平[8]。随着工艺技术的发展和市场需求的扩大，超大规模、高速、低功耗的新型 FPGA 不断推陈出新。新一代的 FPGA 甚至集成了中央处理器(CPU)或数字处理器(DSP)内核[9]，在一片 FPGA 上进行软硬件协同设计，为实现片上可编程系统(SOPC，System On Programmable Chip)提供了强大的硬件支持。

目前，FPGA 的主要发展动向是：随着大规模现场可编程逻辑器件的发展，系统设计进入“片上可编程系统(SOPC)”的新纪元；芯片朝着高密度、低电压、低功耗方向挺进；国际各大公司都在积极扩充其 IP 库，以优化的资源更好地满足用户的需求，扩大市场；特别是引人注目的所谓 FPGA 动态可重构技术的开拓，将推动数字系统设计观念的巨大转变。近年来，随着集成芯片制造技术的发展，现场可编程门阵列在速度和集成度两方面得到了飞速提高。由于它具有功耗低、体积小、集成度高、速度快、开发周期短、费用低、用户可定义功能及可重复编程和重复擦写等许多优点，应用领域不断扩大，越来越多的电子系统开始采用可编程逻辑器件来实现。

目前 FPGA 的品种很多，主要有 XILINX 公司的 Spartan、Vertex 系列、ALTERA 公司的 FIEX 系列、Actel 公司的 ProASIC 系列以及 TI 公司的 TPC 系列等。根据 FPGA 基本结构的不同，可以将其分为基于乘积项(Product-Term)技术的 FPGA 和基于查找表(Look-Up-Table)技术的 FPGA 两种[10]。

基于乘积项技术的 FPGA 主要由 3 个模块组成——逻辑单元阵列(Logic Cell Array)、可编程连线(PIA)和 I/O 控制块。逻辑单元阵列是 FPGA 的基本结构，由它来实现基本的逻辑功能；可编程连线负责信号传递，连接所有的宏单元；I/O 控制块负责输入输出的电气特性控制。

基于查找表技术的 FPGA 是目前的主流产品。查找表(LUT)本质上就是一个 RAM，目前 FPGA 中多使用四输入的 LUT，所以每一个 LUT 可以看成一个有 4 位地址线的 16×1 的 RAM。当用户通过原理图或 HDL 语言描述了一个逻辑电路以后，FPGA 开发软件会自动计算逻辑电路的所有可能的结果，并把结果事先写入 RAM，这样每输入一个信号进行逻辑运算就等于输入一个地址进行查找。

FPGA 的使用非常灵活。目前，大部分的 FPGA 在使用时都需要外接一个 EPROM 来保存其程序，加电时，FPGA 芯片将 EPROM 中的数据读入到片内编程 RAM，配置完成后，FPGA 进入工作状态；掉电后，FPGA 恢复成白片，内部逻辑关系消失。FPGA 的编程无需专用的 FPGA 编程器，只需用通用的 EPROM、PROM 编程器即可。当需要修改 FPGA 的功能时，只需更换一块 EPROM 即可，这样，同一片 FPGA，配置不同的数据就可以实现不同的电路功能。

对于图像压缩算法，特别是应用于深空探测的遥感图像压缩算法，大部分采用单片 FPGA 作为实现方案[11-13]。当然也有采用软硬件相结合的方式来实现，如计算量较密集的部分采用 FPGA 来实现，编码部分采用 DSP 芯片[14]。由于受限于处理速度过慢和功率消耗太大，纯软件方法实现图像压缩算法和加密算法基本被放弃用于太空探测和近地观测。而传统的 ASIC 实现方法，由于实现成本太高，并且缺乏必要的灵活性，不能在系统启动后进行再配置，严重地限制了它的应用。而 FPGA 实现恰好能避免纯软件和 ASIC 实现所存在的弊端：在速度上 FPGA 明显快于纯软件实现；而在灵活性和成本方面，FPGA 要优于 ASIC 实现。本项目采用的是 Macro Blaze 开发版，该开发版是专门为多媒体应用设计的，其采用 Xilinx 公司推出的 Vertex-II 系列 FPGA。由于考虑到代码的平台可移植性，本文并没有像文献[14]中那样采用 DSP 芯片来协助完成图像压缩算法，而是直接在单块 FPGA 芯片上实现了 CCSDS 图像压缩算法。

1.3 本文的主要内容及其安排

本论文详细介绍了 CCSDS 图像压缩算法，给出了 CCSDS 图像压缩算法的编码器的 FPGA 设计方法、系统验证方法以及软硬件仿真结果。主要内容安排如下：

第一章对深空图像压缩算法以及 FPGA 设计进行了简单介绍。

第二章深入介绍了 CCSDS 图像压缩算法的原理，重点是离散小波变换和位平面编码算法的介绍，并给出了 CCSDS 算法、SPIHT 算法和 JPEG2000 算法的性能比较。

第三章对基于 EDA 工具的 FPGA 设计流程、设计方法、设计工具等进行了介绍。

第四章介绍了基于 FPGA 的 CCSDS 图像压缩算法编码器的实现方法，包括系统的总体设计以及离散小波变换、直流编码和交流编码等功能模块的设计。

第五章给出了软件和硬件仿真验证平台的设计方法及其验证结果。

第六章对本文的工作进行了总结，并对下一步的研究进行了展望。

第二章 CCSDS 图像压缩算法

2.1 CCSDS 图像压缩算法概述

CCSDS 图像压缩算法是空间数据系统咨询委员会(Consultative Committee for Space Data System)正在制定的一个针对深空探测图像的压缩标准。该标准采用一系列高效的算法，降低了运算复杂度，以方便其在硬件上高速实现。该算法不需要存储大量的中间数据，从而也减少了内存开销。该压缩算法既适合以帧为单位的图片(例如 CCD 阵列)也适合推扫输入(例如每次获得一行数据)。

该标准针对最大 16 比特深度的二维灰度图像，可以进行无损压缩和有损压缩。无损压缩时，图像信息可以完全的恢复；有损压缩时，则可以按照用户指定的比率进行压缩。压缩算法采用了离散小波变换(DWT, Discrete Wavelet Transform)，分整数小波变换和浮点小波变换两种方式。整数小波变换时，不会丢失信息且算法简单；浮点小波变换虽然可以提高压缩率，但需要进行浮点计算且不能实现无损压缩。

为了减少在信道传输时比特丢失的影响，对小波变换后的数据进行了分块，每一个块记为一个段。每个段是单独进行压缩编码，并且由于小波变换后，每个段的信息与整幅图像的原始信息相关性差了很多，因此即使一个段丢失了的信息，对整幅图像的影响也不是十分大。段的大小可以根据传输质量的要求进行调整，段小的话抗信息丢失能力就强，但是整幅图像的压缩率就会降低。

在图像小波变换后的段中，先得到的数据对恢复图像的贡献要大于后得到的数据，所形成的嵌入式的数据格式使得用户可以按照所需的压缩率对码流进行截断。协议对图像的压缩可以从图像质量和压缩倍数两个方面进行控制。如果对图像质量有要求，可以在某个位平面截止码流；如果要控制压缩倍数，则可以限制一个段的最大比特数，在达到压缩倍率时截止。

CCSDS 图像压缩算法编码器由两个功能模块组成，如图 2.1 所示。离散小波变换(DWT)模块负责对图像数据进行去相关处理，位平面编码(BPE)模块负责对去相关后的数据进行压缩编码。下面几节将详细介绍两个功能模块。

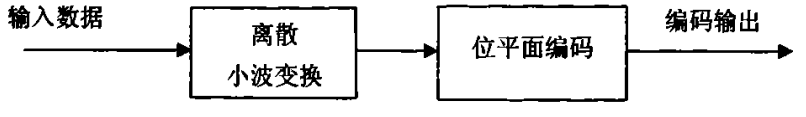


图 2.1 CCSDS 编码器的功能图

Fig. 2.1 Function Schematic of CCSDS coder

2.2 离散小波变换(DWT)

小波变换以傅立叶(Fourier)变换为基础，是变换分析领域的新发展。经过近 10 年的研究，小波分析建立了较为完善的数学体系，小波分析是泛函分析、傅立叶分析、样条分析、数值分析的完美结晶。在信号和信息处理领域，小波分析广泛应用在信号分析、语音合成、图像识别、计算机视觉、地震勘探、大气与海洋波分析等方面并已取得有价值的应用成果。与傅立叶变换对信号进行时/频域整体刻画相对，小波变换对信号进行局部变换，通过伸缩和平移进行多分辨率分析，因而能够有效地提取信号信息[15]。小波图像编码多分辨率的本质非常适合人眼视觉对频率感知的对数特性，因此十分适合图像信号处理。另外，小波变换还具有多层质量控制和嵌入式码流等功能，而且与传统的图像变换方法离散余弦变换(DCT)相比，在大压缩比时它的图像重构质量也明显提高。

2.2.1 小波变换实现图像压缩原理

小波变换用于图像压缩的基本思想是：把图像多分辨率成不同空间、不同频率的子图像，然后再对子图像系数进行编码。系数编码是小波变换用于图像压缩的核心，压缩的实质是对系数的量化压缩[16]。图像经过小波变换后生成的小波图像的数据总量与原图像的数据总量相等，即小波变换本身并不具有压缩功能。之所以将它用于图像压缩，是因为生成的小波图像具有与原图像不同的特性，表现在图像的能量主要集中在低频部分，而水平、垂直和对角线部分的能量则较少；水平、垂直和对角线部分表征了原图像在水平、垂直和对角线部分的边缘信息，具有明显的方向特性。低频部分称为亮度图像，水平、垂直和对角线部分则称为细节图像[17]。

2.2.2 CCSDS 的小波变换

在 CCSDS 图像压缩标准中，采用了离散小波变换对图像数据去相关。它提供了两种形式的小波变换：浮点小波变换和整数小波变换，其滤波器系数如表 2.1 所示。

此低通滤波器的阶数为 9，高通滤波器的阶数为 7，所以也称之为 9/7 离散小波变换。令 C_j 表示低通小波系数，代表原始信号的平滑信息；D_j 表示高通小波系数，代表原始信号的细节信息，则对于 $2N$ 个输入样点 $\{x_0, x_1, \cdots x_{2N-1}\}$，其一维单级 9/7 离散小波变换公式为：

$$\begin{cases} C_j = \sum_{n=-4}^{4} h_n x_{2j-n} \\ D_j = \sum_{n=-3}^{3} g_n x_{2j+1-n} \end{cases}, \qquad j = 1,2,\cdots,N-1 \tag{2.1}$$

这里对输入样点的左边界和右边界需要进行镜像对称延拓处理。

$$\begin{cases} x_m = x_{-m}, & m < 0 \\ x_{2N-1+m} = x_{2N-1-m}, & m > 0 \end{cases} \tag{2.2}$$

浮点小波变换要求进行浮点运算，在低压缩率下能得到更好的性能；而整数小波变换可以实现无损压缩，并且不需要浮点运算，更适合硬件实现，CCSDS 标准推荐使用此方法。

表 2.1 整数和浮点 9/7 分析滤波器系数

Tab. 2.1 Integer and float point analysis filter coefficients for the 9/7 filter

	整数分析滤波器系数		浮点分析滤波器系数	
i	低通滤波器系数 h_i	高通滤波器系数 g_i	低通滤波器系数 h_i	高通滤波器系数 g_i
0	23/32	1	0.852698679009	-0.788485616406
±1	1/4	-9/16	0.377402855613	0.418092273222
±2	-1/8	0	-0.110624404418	0.040689417609
±3	0	1/16	-0.023849465020	-0.064538882629
±4	1/64		0.037828455507	

在实际中，图像经过数字图像处理后是以整数表示的。目前图像处理应用的小波变换常通过 Mallat 算法实现，而其小波滤波器通常都是小数形式。这时，滤波器的输出结果就不再是整数，此时的变换就不能实现无失真的重构[19,23]。为了实现无损压缩，Sweldens 等提出了一种新的方案，该算法采用提升的方法，构造了能够将整数映射到整数的紧支集双正交小波，从而实现图像的无损压缩。该小波变换也称为第二代小波 [20,21]。

提升结构具有诸多优点，原因体现在其自身的结构优势上。如图 2.2 所示[22]，基于提升的滤波由简单的滤波器组成，其基本思想是将小波变换通过分裂、预测和更新三个步骤实现[23]。下面结合框图简单说明提升变换的变换原理。

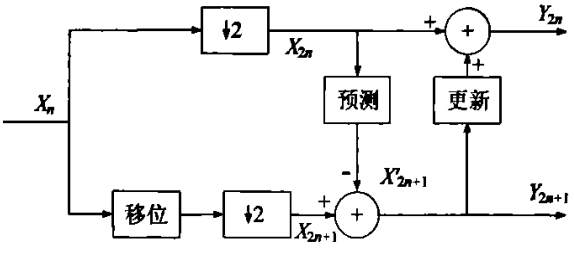


图 2.2 提升方法实现小波分解

Fig. 2.2 Lifting method for wavelet decomposition

图 2.2 中的输入信号 X_n 经过直接下采样和移位下采样，得到偶数序列 X_{2n} 和奇数序列 X_{2n+1}，这个过程称为 Lazy 小波分解；然后利用偶序列 X_{2n} 来预测奇数序列，预测结果记为 X'_{2n+1}；X'_{2n+1} 与奇数序列 X_{2n+1} 之差作为信号的高频分量 Y_{2n+1}，即 $Y_{2n+1}=X_{2n+1}-P(X_{2n})$；最后将误差 Y_{2n+1} 经过更新过程对偶序列更新，获得低频分量 Y_{2n}，即 $Y_{2n}=X_{2n}+U(Y_{2n+1})$[7]。

构造预测阶段 P 时需要考虑原始信号本身的特点，选择合适的预测模型及相应的预测系数等。如果预测方法选择得好，则有利于数据的去相关，预测结果 X'_{2n+1} 会更接近 X_{2n+1}，这样得到的细节部分(高频分量)会更小，从而变换后的系数能量就越集中于低频分量，更有利于后面的熵编码，得到更高的压缩比。更新阶段 U 的目的是寻找一个更好的子集，使其保持原始数据集的特性，即所得低频分量的均值和原始信号均值相同，这是一个能量集中的过程。无论是预测还是更新，都可称为是提升格式的一个提升环节。为了提高算法性能，通常需要进行多次预测和更新，即对图 2.2 的输出继续重复 P 和 U 的过程，最终获得低频和高频分量。

由此可见，提升小波变换运算复杂度低，不需要额外的存储空间，易于逆变换(结构反转)，且与正变换具有同样的复杂性，因而有利于硬件实现[24]。通过分析可知，参与变换的信号可以是任意长度，而不必是 2 的整数次幂。它的计算量要比用一般卷积运算方法小很多，而得到的小波系数与使用传统小波变换得到的结果相同。另外，提升方法很容易扩展到整数变换，并能提供精确重构，同时还易于构造非线性小波变换，通过合理地选择预测因子，使恢复的图像品质更好。

前面提到，提升小波变换可以实现整数到整数的变换。但究其本质，任何小波变换都可以用提升方法实现，计算得出的小波系数是整数还是小数取决于预测和更新步骤的系数。对于待变换原始数据为整数的情况，如果 P 和 U 的系数为小数，则结果是小数(特殊情况除外)。为了达到无损压缩，可以对每个提升操作运算的结果取整，小波变换就实现了整数到整数的映射。取整操作相当于对原来的小波滤波器系数作了很小的改动，但是仍然保留小波分解的特性。

设一维待变换的数据 x 长度为 $2N$，即 $x_i(i=0,\ldots,2N-1)$，并令 $D_i(i=0,\ldots,N-1)$表示高频分量，$C_i(i=0,\ldots,N-1)$表示低频分量，则有[6]

$$D_0 = x_1 - \left\lfloor \frac{9}{16}(x_0+x_2) - \frac{1}{16}(x_2+x_4) + \frac{1}{2} \right\rfloor \tag{2.3}$$

$$D_j = x_{2j+1} - \left\lfloor \frac{9}{16}(x_{2j}+x_{2j+2}) - \frac{1}{16}(x_{2j-2}+x_{2j+4}) + \frac{1}{2} \right\rfloor, \qquad j=1,\ldots,N-3 \tag{2.4}$$

$$D_{N-2} = x_{2N-3} - \left\lfloor \frac{9}{16}(x_{2N-4}+x_{2N-2}) - \frac{1}{16}(x_{2N-6}+x_{2N-2}) + \frac{1}{2} \right\rfloor \tag{2.5}$$

$$D_{N-1} = x_{2N-1} - \left\lfloor \frac{9}{8}x_{2N-2} - \frac{1}{8}x_{2N-4} + \frac{1}{2} \right\rfloor \tag{2.6}$$

$$C_0 = x_0 - \left\lfloor -\frac{D_0}{2} + \frac{1}{2} \right\rfloor \tag{2.7}$$

$$C_j = x_{2j} - \left\lfloor -\frac{D_{j-1} + D_j}{4} + \frac{1}{2} \right\rfloor, \qquad j = 1,..., N\text{-}1 \tag{2.8}$$

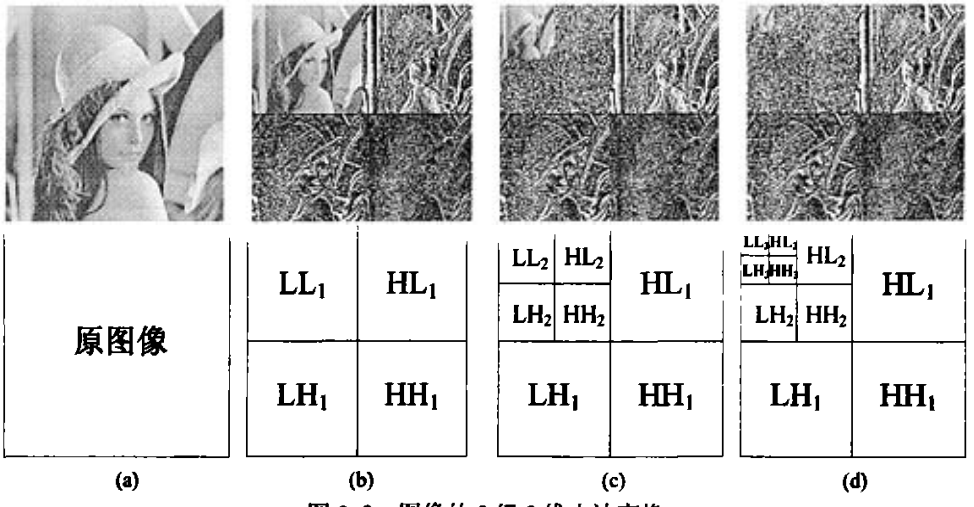


图 2.3 图像的 3 级 2 维小波变换

Fig. 2.3 3-Level 2-D DWT decomposition of an image

其中$\lfloor x \rfloor$表示对 x 作向下取整操作。其重构公式，即整数提升小波变换的反变换为[6]

$$x_0 = C_0 + \left\lfloor -\frac{D_0}{2} + \frac{1}{2} \right\rfloor \tag{2.9}$$

$$x_{2j} = C_j + \left\lfloor -\frac{D_{j-1} + D_j}{4} + \frac{1}{2} \right\rfloor, \quad j = 1, \cdots, N-1 \tag{2.10}$$

$$x_1 = D_0 + \left\lfloor \frac{9}{16}(x_0 + x_2) - \frac{1}{16}(x_2 + x_4) + \frac{1}{2} \right\rfloor \tag{2.11}$$

$$x_{2j+1} = D_j + \left\lfloor \frac{9}{16}(x_{2j} + x_{2j+2}) - \frac{1}{16}(x_{2j-2} + x_{2j+4}) + \frac{1}{2} \right\rfloor, \quad j = 1, \cdots, N-3 \tag{2.12}$$

$$x_{2N-3} = D_{N-2} + \left\lfloor \frac{9}{16}(x_{2N-4} + x_{2N-2}) - \frac{1}{16}(x_{2N-6} + x_{2N-2}) + \frac{1}{2} \right\rfloor \tag{2.13}$$

$$x_{2N-1} = D_{N-1} + \left\lfloor \frac{9}{8}x_{2N-2} - \frac{1}{8}x_{2N-4} + \frac{1}{2} \right\rfloor \tag{2.14}$$

CCSDS 建议进行 3 级 2 维的 9/7 整数变换，其变换过程的相应结果如图 2.3。其中“LL”表示经过低通行变换和低通列变换所得的子带，“LH”表示经过低通行变换和高通列变换所得的子带，“HL”和 “HH”的含义同理可知，其中下标数字表示变换的级数。经过 3 级 2 维小波变换，子带 LL_3 为直流系数，其余子带均为交流系数。为了更加充分的利用整数提升小波变换的特性，需对变换后的各个子带作进一步的加权处理，其各子带加权系数如表 2.2 所示。从表 2.2 中可以看出，各个子带加权系数都是 2 的整数次幂，所以可以通过移位操作处理，提高程序的执行效率。

表 2.2 CCSDS 标准 9/7 整数提升小波变换各子带加权系数

Tab. 2.2 CCSDS standards subband weights for the 9/7 integer DWT

子带	HH_1	HL_1,LH_1	HH_2	HL_2,LH_2	HH_3	HL_3,LH_3	LL_3
加权系数	2^0	2^1	2^1	2^2	2^2	2^3	2^3

2.3 位平面编码(BPE)

2.3.1 概述

CCSDS 图像编码标准中，位平面编码的对象是小波变换系数。为了能对小波系数进行更好地组织，CCSDS 图像编码标准定义了小波变换系数相互间的具体关系，如图 2.4 所示。如此组织的原因是基于如下考虑：在图像编码时，高位平面数据的重要程度要大于低位平面，低频部分的重要程度要大于高频部分。因此，这样划分后可以优先编码重要的部分，以提高压缩率和便于截断。位平面编码的主要过程 stage1~stage3，即是从重要的低频部分向相对次要的高频部分编码的过程。详细的流程将在以下的段落进行介绍。采用这种技术不但可以消除或减少编码冗余，也能消除或减少图像中的像素间冗余[25]。

CCSDS 编码处理的单位是“段(segment)”，其大小定义在 32×32 和 8192×8192 之间。对于同一幅图像来说，段划分的越大，则重建图像的质量越好；段划分得小，则运算时需要较少的存储空间。每个段由固定大小的 8×8 块组成，每个块是位平面编码处理的基本单元，它是含有 64 个小波系数的组合，如上图的深色部分。CCSDS 协议规定，一个段中可以包含 16 至 2^{20} 个“块”[6]。

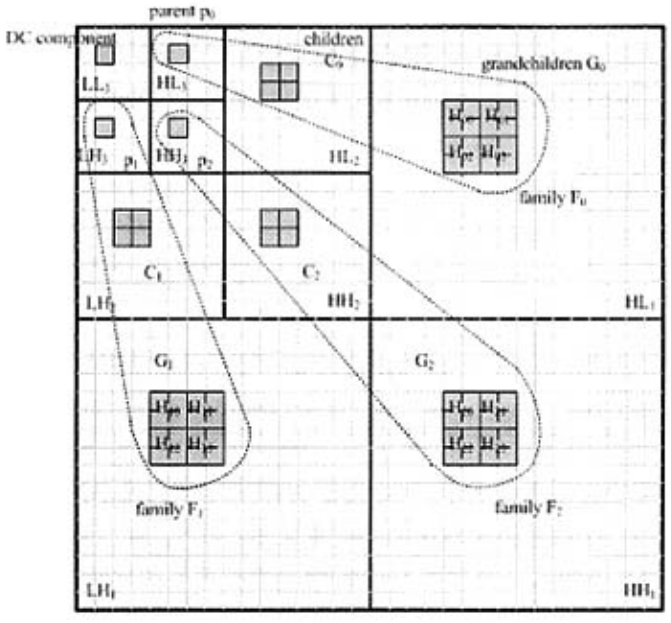


图 2.4　离散小波变换系数图解

Fig. 2.4　Schematic of discrete wavelet transform coefficients

一个块包括 1 个直流系数和 63 个交流系数。直流系数对应 LL_3，它是横向与纵向小波变换后低频的部分。HL_3、HH_3 和 LH_3 子带系数称为双亲(Parents)系数；接下来频率较高的 HL_2、HH_2 和 LH_2 子带系数称为孩子(Children)系数；最后在最外层频率最高的 HL_1、HH_1 和 LH_1 子带系数称为孙子(Grandchildren)系数。为了将相关的系数联系起来，把同一方向的 1 个双亲系数、4 个孩子系数和 16 个孙子系数，这 21 个交流系数共同组成一个“家庭(Family)”。如“家庭”F_0 是由 P_0、C_0 和 G_0 共同组成的。由于位平面编码是每 4 比特编成一个码字，因此对孙子系数又进行了分组，记为 H_{ij}，其中 i=0,1,2，代表家庭，j=0,1,2,3，代表分组的位置。

每个“段”中的 16 个块组成一个“群(gaggle)”，CCSDS 图像编码算法以“群”为单位完成一个“段”的编码。最后一个“群”可能不足 16 个块，而是“群”长度对 16 取模后的值。

根据图 2.3 所示的小波变换系数关系可知，只要知道某个块的直流系数位置，即可以推断各个“家庭”的所有成员在块中的坐标值。为此，为每个子带都建立了一个坐标系，原点是各子带的左上角，并以 (r,c) 表示在某个子带中第 r 行、第 c 列的坐标。表 2.3 给出了一个“家庭”中各成员在各自子带中的坐标值。

表 2.3 一个“家庭”中各系数在各自子带中的坐标

Tab. 2.3 Within-subband for coefficients in a single family

家庭 i 中的系数组	系数坐标
双亲 P_i	(r,c)
孩子 C_i	$(2r,2c),(2r,2c+1),(2r+1,2c),(2r+1,2c+1)$
孙子 H_{i0}	$(4r,4c),(4r,4c+1),(4r+1,4c),(4r+1,4c+1)$
孙子 H_{i1}	$(4r,4c+2),(4r,4c+3),(4r+1,4c+2),(4r+1,4c+3)$
孙子 H_{i2}	$(4r+2,4c),(4r+2,4c+1),(4r+3,4c),(4r+3,4c+1)$
孙子 H_{i3}	$(4r+2,4c+2),(4r+2,4c+3),(4r+3,4c+2),(4r+3,4c+3)$

对于直流系数和交流系数的表示，CCSDS 标准规定，直流系数直接用二进制补码表示，而交流系数用一个符号位和对应的幅度来表示。设 c_m 为一个“段”中的第 m 个直流系数，即此“段”中的第 m 个块的直流系数，则 c_m 需要的比特位数可以用式(2.15)计算，即

$$\begin{cases} 1+\lceil \log_2 |c_m| \rceil, & 若 c_m < 0 \\ 1+\lceil \log_2 (1+c_m) \rceil, & 若 c_m \geq 0 \end{cases} \tag{2.15}$$

首先需要用式(2.15)分别求出“段”中各个直流系数所需的比特位数，并找出最大值用 *BitDepthDC* 表示，称为直流系数比特深度。求出最大值后，该“段”的所有直流系数都可以用 *BitDepthDC* 个比特表示。

交流系数与直流系数的计算方法类似，但需要计算每个块中所有的交流系数。首先求出一个“段”中每个块的交流系数幅度的最大值，再求出这些幅度最大值对应的所需比特位数，用 $BitDepthAC\_Block_m$ 来表示。

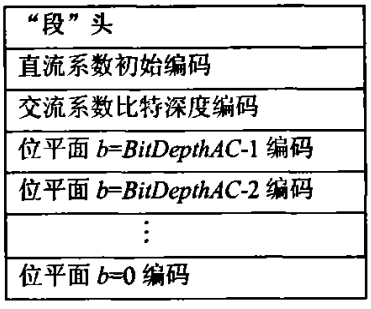
$$BitDepthAC\_Block_m = \lceil \log_2 (1+\max_x |x|) \rceil \tag{2.16}$$

最后求得“段”中所有($BitDepthAC\_Block_m$，m=1,...S)的最大值，用 *BitDepthAC* 表示，即整个“段”中所有交流系数最大幅度所需的比特数，称为交流系数比特深度 *BitDepthAC*。

$$BitDepthAC = \max_{m=0,1,\cdots,S-1} BitDepthAC\_Block_m \tag{2.17}$$

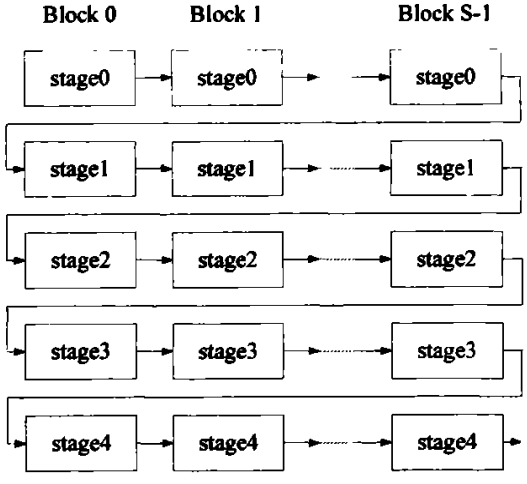
位平面编码是指对交流系数的幅度进行编码，并在第一次出现比特“1”时，插入符号信息。位平面 b 是指小波变换系数的各个比特，其中 b=0 表示最低比特位。位平面编码从 $b = BitDepthAC - 1$ 开始，到 $b = 0$ 结束，每增加一个位平面编码，小波变换系数的分辨率就提高 1 倍。

在一个位平面的编码过程中，对于基本单元“段”来说需要经历图 2.5 所示的几个阶段：首先对“段”头编码，然后对直流系数编码，交流系数比特深度编码，最后是按位平面从大到小进行位平面编码。在一个“段”中，对所有位平面是按图 2.5(a)所示。在一个位平面上则是按图 2.5(b)所示流程进行编码。具体的细节将在接下来的几小节进行讨论。这样编码的好处是能形成嵌入式的码流，能有效的控制图像质量和压缩倍率。



(a) 对所有位平面

(a) All bit planes



(b) 一个位平面内

(b) A bit plane

图 2.5 段编码结构

Fig. 2.5 The structure of a coded segment

2.3.2 “段”头编码

一个“段”的头共包括 Part1、Part2、Part3 和 Part4 四部分，总长为 20 个字节，其中 Part1 又被分为 Part1A 和 Part1B 两部分，如图 2.6 所示。由于篇幅所限，对于各部分的具体含义请参考文献[6]。这 4 个部分中，Part1A 对于每个“段”都是必须的；而 Part1B 只在图像的最后一个“段”中是必须的。Part2、Part3 和 Part4 都是可选的。

Part 1A	Part 1B	Part 2	Part 3	Part4
(3 bytes)	(1 byte)	(5 bytes)	(3 bytes)	(8 bytes)

图 2.6 段头的完全结构

Fig. 2.6 The entire structure of a segment

Part1 的结构如图 2.7 所示。

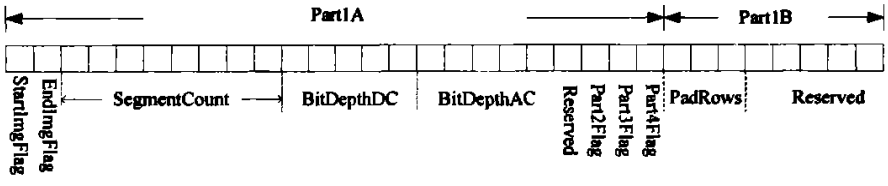


图 2.7 Part1 的完全结构

Fig. 2.7 The entire structure of Part1

Part2 的结构如图 2.8 所示。

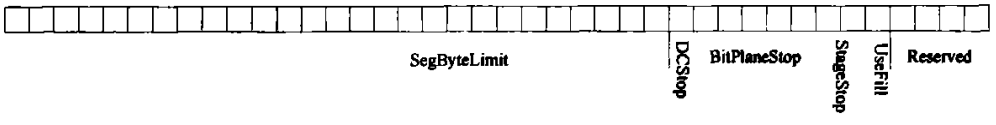


图 2.8 Part2 的结构

Fig. 2.8 The structure of Part2

Part3 的结构如图 2.9 所示。

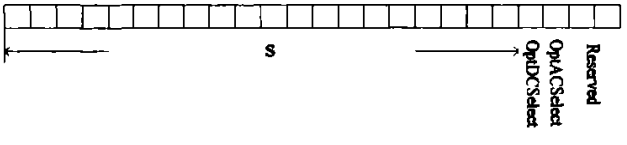


图 2.9 Part3 的结构

Fig. 2.9 The structure of Part3

Part4 的结构如图 2.10 所示。

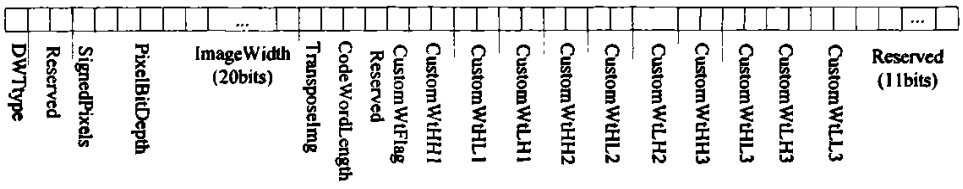


图 2.10 Part4 的结构

Fig. 2.10 The structure of Part4

2.3.3 直流系数量化与编码

直流系数编码主要由两部分组成：首先采用 Rice 算法[26]进行初始编码，未编码的比特在 stage0 中进行编码。在 BitShift(LL_3)以下的位平面不需要编码，对于整数提升小波变换而言，BitShift(LL_3) = 3。直流系数的初始编码由两步完成：第一步进行量化直流系数编码，第二步进行附加位平面编码，如图 2.11 所示。注意，这里要求量化因子 q 大于交流系数比特深度 *BitDepthAC*，否则不存在附加位平面编码。

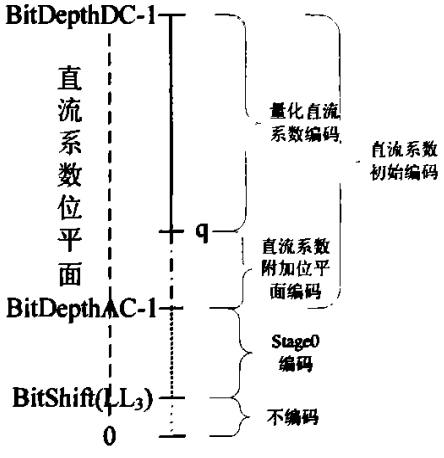


图 2.11 直流系数编码的结构

Fig. 2.11 The structure of DC coefficients coding

(1) 直流系数的量化

首先要对直流系数进行量化。由直流系数(DC)和交流系数(AC)的动态范围来确定量化参数 q'，见表 2.4。

表 2.4 直流系数的量化参数

Tab. 2.4 Quantizaion parameter of DC coefficients

AC 和 DC 的动态范围	参数 k 的选取
$BitDepthDC \le 3$	q=0
$BitDepthDC-(1+\lfloor BitDepAC/2 \rfloor) \le 1$ 并且 $BitDepthDC > 3$	q'=BitDepthDC-3
$BitDepthDC-(1+\lfloor BitDepAC/2 \rfloor) > 10$ 并且 $BitDepthDC > 3$	q'=BitDepthDC-10
其他	$q'=1+\lfloor BitDepAC/2 \rfloor$

由于对直流系数存在加权处理，所以量化参数 q' 并不能作为量化因子。真正的量化因子 q 由式(2.18)决定

$$q = \max(q', BitShift(LL_3)) \tag{2.18}$$

量化因子 q 应该是性能与效率的折中：当量化因子 q 足够小时，可以使直流系数之间的相关性尽量多的保留，更有利于预测编码；当量化因子 q 足够大时，可以减少 stage0 中直流系数量化的比特，更有利于质量渐进式的实现。因此在选择 q 时需要从性能与效率两方面考虑以达到最优。

获得 q 后就可以对所有的直流系数进行量化。设长度为 S 的直流系数序列为 $\{c_m, m=0,1,\ldots,S-1\}$，则量化后的直流系数 c_m' 为

$$c_m' = \lfloor c_m / 2^q \rfloor \tag{2.19}$$

量化后的直流系数 c_m' 可以用 N 比特来表示：

$$N = \max\{BitDepthDC - q,\ 1\} \tag{2.20}$$

由表 2.6 可知，N 的最大值为 10。

当 N=1 时，也就是量化后的直流系数 c_m' 可以由 1 个比特表示，则直接把这个直流系数写入编码码流中。当 N>1 时，第一个量化后的直流系数 c_0' 称为参考样点，直接编码，剩余的 S-1 个直流系数将计算其自身与前一个系数的差分序列 δ_m'，即

$$\delta_m' = c_m' - c_{m-1}' \tag{2.21}$$

然后，将这些差分序列根据式(2.22)映射成非负整数δ_m

$$\begin{cases}\delta_m = 2(\delta_m'), & 0 \le \delta_m' \le \theta_m \\ \delta_m = 2|\delta_m'|-1, & -\theta_m \le \delta_m' < 0 \\ \delta_m = \theta_m + |\delta_m'|, & \text{其它}\end{cases} \tag{2.22}$$

其中$\theta_m = \min(c_m' - x_{\min}, x_{\max} - c_{m-1}')$，$x_{\min} = -2^{N-1}$，$x_{\max} = 2^{N-1} - 1$。

直流系数编码的对象是δ_m序列中的16个δ_m构成的“群”。第一个“群”由于存在一个参考样点，所以只包含15个δ_m。而当直流系数长度S不是16的整数倍时，则最后一个“群”只包含J个δ_m，其中$J=S \bmod 16$。其余“群”都包含16个δ_m。

为了对这些差分参数进行编码，每个群需要包含一个指示编码方式的参数k。它是一个变长的码字，码字长度是由自身的取值以及N值的范围共同确定，具体关系如表2.6所示。如果k为uncoded，该群(gaggle)进行定长编码，即按照原有的N比特形式编码，具体结构如图2.12所示；否则进行变长编码，此时δ_m被分为两部分。第一部分由z个“0”和末尾一个“1”组成，其中$z = \lfloor \delta_m / 2^k \rfloor$；第二部分由$\delta_m$余下的$k$个比特位组成。具体结构如图2.13所示。

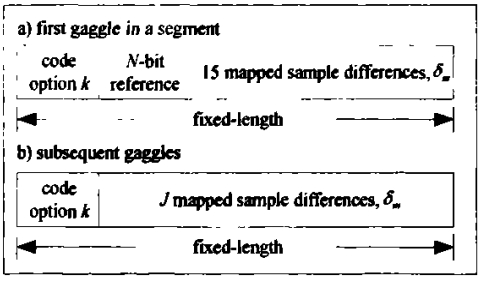


图2.12 固定长度群的码流结构

Fig. 2.12 Coded data format for a fixed-length gaggle

参数k有两种计算方法，最优化方法和自适应方法。其中，最优化方法就是遍历k的所有可能取值，之后选择使群码流长度最短的k值，但这种方法计算量较大；另一种自适应方法是依据群内δ_m参数的相关信息得到k值，较为简单直接，本文的实现过程使用这种方法。令$\Delta = \sum_m \delta_m$为群内所有δ_m参数的和，J为群内所有δ_m参数的个数，则k值

由表 2.5 确定。选择好 k 值后，其对应的码字如表 2.6 所示。

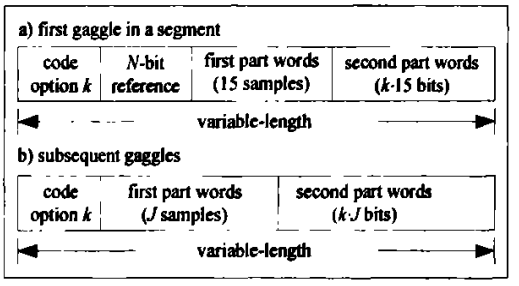


图 2.13　变长群的码流结构

Fig. 2.13　Coded data format for a variable-length gaggle

表 2.5　编码选项 k 的选取规则

Tab. 2.5　Code option selection rules of k

条件	参数 k 的选取
$64\cdot\Delta \geq 23\cdot J\cdot 2^N$	uncoded
$207\cdot J > 128\cdot\Delta$	$k=0$
$J\cdot 2^{N-5} \leq 128\cdot\Delta+49\cdot J$	$k=N$-2
其他	k 是小于等于 N-2 非负的且满足 $J\cdot 2^{k+7} \leq (128\cdot\Delta+49\cdot J)$ 的最大的整数

表 2.6　编码参数 k 的取值与对应码字

Tab. 2.6　Values and codewords for code option identifier k

编码参数 k	N=2	2<N≤4	4<N≤8	8<N≤10
k=0	0	00	000	0000
k=1	—	01	001	0001
k=2	—	10	010	0010
k=3	—	—	011	0011
k=4	—	—	100	0100
k=5	—	—	101	0101
k=6	—	—	110	0110
k=7	—	—	—	0111
k=8	—	—	—	1000
uncoded	1	11	111	1111

'—'表示不可能的取值

以上详细论述了量化后的直流系数编码，即处理的是比特平面大于 q 的部分。由于后面的位平面编码过程针对的是小于 *BitDepthAC* 的比特平面(图 2.5)，所以对于 $q>$ *BitDepthAC* 情况下第[q-1, *BitDepthAC*]个位平面部分的 DC 系数需要另外处理。具体做法就是将它们按照位平面上从高到低的顺序，位平面内光栅扫描顺序直接串行输出，这个过程将在 stage0 中处理。

2.3.4 交流系数的比特深度编码

交流系数比特深度编码与直流系数类似。根据不同的 *BitDepthAC* 的值，交流系数比特深度 $BitDepthAC\_Block_m$ 有如下几种编码方式：当 *BitDepthAC* 等于 0 时，说明各个块的 $BitDepthAC\_Block_m$ 都为 0，那么不需要对其进行编码，交流系数也不需要编码。当 *BitDepthAC* 等于 1 时，说明 $BitDepthAC\_Block_m$ 等于 0 或者 1，可以只用 1 比特来表示，也就是直接将 $BitDepthAC\_Block_m$ 的值写入编码码流中。当 *BitDepthAC* 不等于 0 或 1 时，将采用直流系数初始编码方法对 $BitDepthAC\_Block_m$ 编码，但需要修正以下两点：

① N 的计算公式改为

$$N=\lceil \log_2(1+BitDepthAC) \rceil \tag{2.23}$$

对于 16 比特深度的图像，$N\le 5$。

② 由于 $\mathrm{BitDepthAC\_Block_m}$ 是非负整数，所以 $x_{\min}$ 与 $x_{\max}$ 的表达式改为

$$x_{\min}=0,\ x_{\max}=2^N-1 \tag{2.24}$$

2.3.5 位平面编码

位平面编码一共分 5 个阶段(stage)，每个阶段的编码对象是块。位平面编码是从最高的位平面(*BitDepthAC*-1)开始，到较低的位平面结束。在同一个位平面上则由阶段 0 至阶段 4 构成，每个阶段都要对所有的块进行编码。由于编码过程是分阶段的，先后不受影响，故称为这样的码流为嵌入式码流。嵌入式的码流可以在图像质量和压缩倍数之间得到较好的折中，得到用户需要的效果。对于图像质量，可以控制在某个位平面截止；对于码流长度，可以限定“段”的大小，当一个“段”编码一定比特数时停止编码。这些控制字则可以在“段”头的 Part2 部分的前四个参数中进行设置。

(1) 编码阶段 0 (stage 0)

在直流系数量化编码中，已经将高于 q 位的直流系数编入码流，阶段 0 则是针对直流系数的低 q 位进行编码。当位平面 b 大于等于 q 时，或者当位平面 b 小于直流系数的加权值 LS(LL_3)时，阶段 0 不进行编码，否则进行阶段 0 编码。

(2) 编码阶段 1～3 (stage 1~3)

stage1 到 stage3 编码阶段的作用是对交流系数幅值冗余位进行编码，由于交流系数的最高有效位的信息要远远大于其它位。在 stage1 将编码双亲系数，在 stage2 编码孩子系数，stage3 编码孙子系数。对于一个位平面 b，为了确定交流系数在哪一个阶段编码，需要计算每个交流系数 x 在此位平面上的类型 $t_b(x)$，即

$$\begin{cases} t_b(x)=0, & |x|<2^b \\ t_b(x)=1, & 2^b \le |x|<2^{b+1} \\ t_b(x)=2, & 2^{b+1} \le |x| \\ t_b(x)=-1, & b<BitShift(\Gamma) \end{cases} \tag{2.25}$$

其中 Γ 表示交流系数 x 的子带权值。如果 $t_{b+1}(x)$ =1，则说明这个交流系数 x 正在编码，也就是说此交流系数 x 第一次出现了“1”位平面；如果 $t_{b+1}(x)=0$，则称这个交流系数 x 还未被编码；如果 $t_{b+1}(x)=2$，则称这个交流系数 x 在 b 之前的位平面已经被编码；如果 $t_{b+1}(x)$ = -1，则认为这个交流系数 x 在 b 位平面上一定是 0。对于一个交流系数，一般都会经历从类型 0，到类型 1，然后到类型 2，最后到类型-1 的编码过程。

一个 AC 系数 x 在哪个阶段(stage)编码取决于位平面 b 时 x 的类型。当 AC 系数 x 是类型 0 或类型 1(此时 $t_{b+1}(x)=0$)时，则 x 的 b^{th} 位在 stage1 到 stage3 中编码。此时可以判别 x 系数在一个块中的位置，如果是双亲系数则在 stage1 中编码，如果是孩子系数则在 stage2 中编码，如果是孙子则系数在 stage3 中编码。为了进一步减少比特流长度，还需要用到一些先验的知识，首先把预编码的比特信息定义为一系列的可变长二进制码字；然后，对这些码字中的某些部分做进一步的熵编码。当 x 的类型为 2 时，则该比特的信息不进行编码，而在 stage4 中直接写入码流。若类型为-1 时则不进行处理。

根据图 2.4，我们可以定义一些字符来表示小波变换系数类型与系数位置之间的具体关系。其中 C_i 表示第 i 个家庭的孩子系数；G_i 表示第 i 个家庭的孙子系数；H_{ij} 表示第 i 个家庭中第 j 组孙子系数；$P=\{p_0,p_1,p_2\}$ 表示一个块中双亲的集合；$D_i=\{C_i,G_i\}$ 表示第 i 个家庭的子孙(后代)系数；$B=\{D_0,D_1,D_2\}$ 为一个块中所有子孙(后代)的集合；此外，还定义了字符 Ψ、$types_b(\Psi)$、$signs_b(\Psi)$ 和 $tword[\Lambda]$：Ψ 表示交流系数集合；$types_b(\Psi)$ 为集合 Ψ 中交流系数 x 的类型为 0 或 1 时，第 b 比特位的集合；$signs_b(\Psi)$ 为集合 Ψ 中满足 $t_b(x)=1$ 的系数的符号位的集合，其中正数的符号位表示为二进制的 0，负数的符号位表示为二进制的 1；$tword[\Lambda]$ 表示类型序列 $\Lambda=\{\lambda_0,\lambda_1,\lambda_2,\cdots,\lambda_7\}$ 中，元素 λ_i 的值等于 0 或 1 的集合。

在高位平面下，在大量块中有很多连续 0 的情况。为了减少对连零的编码，需要定义一些“过渡(transition)”字，如下所示：

$tran_B = tword[\{t_{\max}(B)\}]$;

$tran_D = tword[t_{\max}(D_0), t_{\max}(D_1), t_{\max}(D_2)]$;

$tran_G = tword[t_{\max}(G_0), t_{\max}(G_1), t_{\max}(G_2)]$, 并且在当前或之前的位平面上满足 $t_{\max}(D_i) > 0, i = 0,1,2$;

$tran_{H_i} = tword[\{t_{\max}(H_{i0}), t_{\max}(H_{i1}), t_{\max}(H_{i2}), t_{\max}(H_{i3})\}], i = 0,1,2$，其中 $t_{\max}(\Psi)$ 表示集合 Ψ 内系数类型的最大值。

接下来利用上面的定义，对阶段 1、阶段 2 和阶段 3 的按照如下方法进行编码：

① Stage1（双亲系数）

$types_b[P]$，$signs_b[P]$.

② Stage2（孩子系数）

a) $tran_B$.

b) $tran_D$，若 $tran_B \neq 0$ 且 $t_{max}(B) \neq -1$.

c) $types_b[C_i]$ 和 $signs_b[C_i]$，且对于每一个 i 都满足 $t_{max}(D_i) \neq 0,-1$.

③ Stage3（孙子系数）

如果 $tran_B$=0 或者 $t_{max}(B)$=-1,则不需要进行 stage3 编码；否则由以下三步构成：

a) $tran_G$.

b) $tran_{Hi}$，且对于每一个 i 都满足 $t_{max}(G_i) \neq 0,-1$.

c) $types_b[H_{ij}]$ 和 $signs_b[H_{ij}]$，且对于每一个 i 都满足 $t_{max}(G_i) \neq 0,-1$；对于每一个 i 和 j 都满足 $t_{max}(H_{ij}) \neq 0,-1$.

为论述方便，将上述方法中出现的所有需要编码的元素统称为编码字。以上各个编码阶段(stages)所产生的所有码字均为可变长，其中包括空码字(NULL word)。

上面得到的 $types_b[P]$, $types_b[C_i]$, $types_b[H_{ij}]$, $tran_D$, $tran_G$, $tran_{Hi}$ 码字，除了只有 1 比特长度和符号比特的码字直接打入码流以外，均需要进行熵编码，下面进行具体介绍。

对于某些码字来讲，并不是所有的可能值都会出现。如 $tran_D$ 不可能为“000”，因为这种情况可以由 $tran_B = 0$ 代表。表 2.7 给出了欲进行熵编码码字的最大可能长度和非可能值。

熵编码分两步进行：首先，将编码字查表映射为整数值，这些整数值称为符号(symbol)；然后，将符号转化成变长码字。

表 2.8，2.9 和 2.10 分别给出了将 2 比特、3 比特和 4 比特长度的编码字转化为符号的映射表。

表 2.7 最大长度和不可能的码字值的总结

Tab. 2.7 Summary of maximum word lengths and impossible word values

编码字	最大长度(bits)	不可能的值
$types_b[P]$	3	—
$types_b[Ci]$	4	—
$types_b[Hij]$	4	0000
$tran_D$	3	000
$tran_G$	3	—
$tran_{Hi}$	4	0000

表 2.8 2 比特编码字整数映射表

Tab. 2.8 Integer mapping for two-bit words

编码字 (Word)	符号 (Symbol)
00	0
01	2
10	1
11	3

表 2.9 3 比特编码字整数映射表

Tab. 2.9 Integer mapping for three-bit words

编码字 (Word)	符号(symbol) ($types_b[P]$,$types_b[Ci]$, $types_b[Hij]$,$tran_G$,$tran_{Hi}$)	符号(symbol) ($tran_D$)
000	1	—
001	4	3
010	0	0
011	5	4
100	2	1
101	6	5
110	3	2
111	7	6

表 2.10　4 比特编码字整数映射表

Tab. 2.10　Integer mapping for four-bit words

编码字 (Word)	符号(symbol) ($types_b[C_i]$)	符号(symbol) ($types_b[Hij]$,$tran_{Hi}$)
0000	10	—
0001	1	1
0010	3	3
0011	6	6
0100	2	2
0101	5	5
0110	9	9
0111	12	11
1000	0	0
1001	8	8
1010	7	7
1011	13	12
1100	4	4
1101	14	13
1110	11	10
1111	15	14

表 2.11，2.12 和 2.13 分别给出了将 2 比特、3 比特和 4 比特长度编码字对应的符号转化为变长码的映射表。当每个映射表有多个编码选项时，需要选择一个选项使编码长度最短。这个选择的过程有点类似与直流系数编码中确定选项标识符 k 的方法。它是以一个“群”为单位，选择一个选项使得变长码编码长度最短。对于一个群来说这个选项在 stage1~stgage3 中是同一个值。

表 2.11　2 比特编码字的可变长码选项

Tab. 2.11　Variable length code options for two-bit words

输入符号 (symbol)	编码选项 0	编码选项 1 (uncode)
0	1	00
1	01	01
2	001	10
3	000	11

表 2.12　3 比特编码字的可变长码选项

Tab. 2.12　Variable length code options for three-bit words

输入符号 (symbol)	编码选项 0	编码选项 1	编码选项 3 (uncode)
0	1	10	000
1	01	11	001
2	001	010	010
3	00000	011	011
4	00001	0010	100
5	00010	0011	101
6	000110	0000	110
7	000111	0001	111

表 2.13　4 比特编码字的可变长码选项

Tab. 2.13　Variable length code options for four-bit words

输入符号 (symbol)	编码选项 0	编码选项 1	编码选项 2	编码选项 3 (uncode)
0	1	10	100	0000
1	01	11	101	0001
2	001	010	110	0010
3	0001	011	111	0011
4	0000000	0010	0100	0100
5	0000001	0011	0101	0101
6	0000010	000000	0110	0110
7	0000011	000001	0111	0111
8	00001000	000010	00100	1000
9	00001001	000011	00101	1001
10	00001010	000100	00110	1010
11	00001011	000101	00111	1011
12	00001100	0001100	00000	1100
13	00001101	0001101	00001	1101
14	00001110	0001110	00010	1110
15	00001111	0001111	00011	1111

在一个“群”中，编码选项最多为 5 比特，表 2.14 给出了不同编码字对应的编码选项长度和码字。当群(gaggle)的某个长度的码字出现时，其编码选项的码字会在该码字之前打入码流。

表 2.14 编码选项的码字

Tab. 2.14 The codeword of the variable length code options

编码字的长度(bits)	编码选项的长度(bits)	编码选项的码字
2	1	0：编码选项 0 1：uncode
3	2	00：编码选项 0 01：编码选项 1 11：uncode
4	2	00：编码选项 0 01：编码选项 1 10：编码选项 2 11：uncode

(4) 阶段 4 编码 (stage 4)

当交流系数 x 的类型 $t_b(x)$等于 2 时，进行阶段 4 的编码。编码方法是将 b 位平面上的比特不进行任何压缩，将系数信息直接输出。对于一个块来讲，其各个系数打入码流的顺序为：$p_i(i=0,1,2)$，C_i 的成员$(i=0,1,2)$和 H_{ij} 的成员$(i=0,1,2,\ j=0,1,2,3)$。

2.4 CCSDS 图像压缩算法性能分析及比较

CCSDS 图像压缩算法的主要特点是以离散小波变换为基础，采用位平面嵌入式编码方式进行编码，当前与其类似的典型图像压缩算法还有 JPEG2000 和 SPIHT[11,13]两种。其中 JPEG2000 算法采用优化截断嵌入式块编码(EBCOT, Embedded Block Coding with Optimized Truncation)方法对小波系数进行编码和排序。JPEG2000 为了达到最佳的码率和图像效果，需要对多个块进行率失真计算，从而增加了缓冲区的大小和硬件的复杂度。SPIHT(集分割层次树，Set Partitioning in Hierarchical Trees)算法主要基于嵌入式零树编码算法，并利用三个链表实现对小波系数幅值的集合划分[27,28]，而链表在实际硬件实现中效率较低。而 CCSDS 算法是以段为单位，各个阶段编码也是独立的，具体实现时具有较大的灵活性。例如在 FPGA 上可以充分实现并行，通过增加流水线则更可以发挥其算法优势，相对于相对 SPIHT 和 JPEG2000 算法，实现更为简单。

(a) marstest (b) spot-1a_b3 (c) spot-panchormatic

图 2.14 空间测试图像举例

Fig. 2.14 Example of test space images

表 2.15 三种算法不同码率下平均性能

Tab. 2.15 Average performance at different rates in different algorithms

Rate (bits/pixel)	PSNR		
	CCSDS	SPIHT	JPEG2000
2.00	41.37	38.72	42.07
1.00	35.76	31.78	36.48
0.50	32.37	23.21	32.85
0.25	29.89	20.68	30.20

根据不同算法，我们对图 2.14 的 3 幅典型的空间图像 marstest、spot-la_b3、spot-panchormatic(分别为 512×512 像素、500×500 像素、1000×1000 像素，均为 8 比特深度)进行了测试。表 2.15 给出了在不同压缩率下三种算法的平均峰值信噪比。可以看出，在表 2.15 的各个码率下，对于空间图像来说，CCSDS 的平均峰值信噪比可以与 JPEG2000 相媲美，且明显高于 SPHIT 算法。在实现上，与 SPHIT 相比较，CCSDS 不存在硬件上不易实现的链表结构。因此，CCSDS 可以在性能与硬件实现复杂度上取得很好的平衡，有利于满足高效的深空探测及近地观测应用。

第三章 FPGA 开发环境

3.1 硬件描述语言

3.1.1 硬件描述语言简介

硬件描述语言 HDL(Hardware Description Language)是一种用形式化方法来描述数字电路和系统的语言。数字电路系统的设计者利用这种语言可以从上层到下层(从抽象到具体)，逐层描述自己的设计思想，用一系列分层次的模块来表示极其复杂的数字系统。然后利用电子设计自动化(EDA)工具逐层进行仿真验证，再把其中需要变为具体物理电路的模块经由自动综合工具转换到门级电路网表，最后再用专用集成电路(ASIC)或现场可编程门阵列(FPGA)自动布局布线工具，把网表转换成具体的电路布线结构[18]。

硬件描述语言发展至今已有 20 多年的历史，并成功地应用于设计的各个阶段：建模、仿真、验证和综合等。到 20 世纪 80 年代，已出现了上百种硬件描述语言，它们对设计自动化曾起到了极大的促进和推动作用。到 20 世纪 80 年代后期，硬件描述语言向着标准化的方向发展。最终，VHDL 和 Verilog HDL 语言适应了这种趋势的要求，先后成为 IEEE 标准。

3.1.2 Verilog HDL 硬件描述语言

Verilog HDL 是硬件描述语言的一种，用于数字电子系统设计。设计者可用它进行各种级别的逻辑设计，可用它进行数字逻辑系统的仿真验证、时序分析、逻辑综合。它是目前应用最广泛的一种硬件描述语言。

Verilog HDL 和 VHDL 作为描述硬件电路设计的语言，其共同的特点在于：能形式化地抽象表示电路的行为和结构；支持逻辑设计中层次与范围的描述；可借用高级语言的精巧结构来简化电路行为的描述；具有电路仿真与验证机制以保证设计的正确性；支持电路描述由高层到低层的综合转换；硬件描述与实现工艺无关；便于文档管理；易于理解和设计重用。

但是 Verilog HDL 和 VHDL 又各有其自己的特点。Verilog HDL 的最大优点为：它是一种非常容易掌握的硬件描述语言，只要有 C 语言的编程基础，通过学习和实际操作，一般很快就可以掌握这种设计。目前版本的 Verilog HDL 和 VHDL 在行为级抽象建模的覆盖范围方面也有所不同。Verilog 较为适合系统级(System)、算法级(Algorithm)、寄存器传输级(RTL)、逻辑级(Logic)、门级(Gate)和电路开关级(Switch)的设计，而对于特大型(千万门级以上)的系统级(System)设计，则 VHDL 更为合适。由于 CCSDS 算法

不是十分复杂，本文选择了 Verilog HDL 语言作为项目的开发语言。

3.2 FPGA 开发平台

3.2.1 FPGA 硬件平台简介

FPGA(Field Programmable Gate Array)即现场可编程门阵列，是 1980 年代中期出现的高密度可编程逻辑器件(PLD)，它是在 PAL、GAL、EPLD 等可编程器件的基础上进一步发展的产物。它是作为专用集成电路(ASIC)领域中的一种半定制电路而出现的，既解决了定制电路的不足，又克服了原有可编程器件门电路数有限的缺点，具有小型化、低功耗、多功能、数字化、标准化、系列化、集成度高、保密性好、可无限次反复编程、并有现场模拟调试验证等优点。FPGA 种类较多，如 Xilinx 公司的 Spartan、Virtex 系列，Altera 公司的 Cydon、stratix、ACEx、APEx 系列等以及 Lattice 公司的 ispLSI、ispMACH4000V/B/C/Z、ispXPGA 系列等。FPGA 采用逻辑单元阵列 LCA(Logic Cell Array)，其内部包括可配置逻辑模块 CLB(Configurable Logic Block)、输出输入模块 IOB(Input Output Block)和内部连线(Interconnect)三个部分。采用 FPGA 设计 ASIC 电路，用户不需要投片生产，就能得到合用的芯片。FPGA 可做其它全定制或半定制 ASIC 电路的中间试验样片。FPGA 内部有丰富的触发器和 1/0 引脚。并且，FPGA 是 ASIC 电路中设计周期最短、开发费用最低、风险最小的器件之一。

本项目采用的 FPGA 硬件开发平台是 Xilinx 公司的一个专门为多媒体应用服务的 MicroBlaze 评估版，采用了 Virtex-II[29]系列的 FPGA。该系列 FPGA 是 Xilinx 公司于 2003 年推出的第四代产品，采用 0.12μm 深亚微米技术工艺和 0.15μm 8 层金属处理技术，提供了 10M PLD 门和 1.5M 位 RAM 等大容量可编程资源，支持基于 IP 核的系统芯片设计实现，可应用于光纤通信、移动通信基站、电视广播等领域。

Virtex-II 系统 FPGA 采用了大量先进的工艺和技术，例如先进的存储器阵列，片上存储器的容量可达 4.5M 位；嵌入式 18 位×18 位高速乘法器模块，支持各种 DSP 功能，新颖的有源互连结构，使器件的连线延时可预计。因而 Virtex-II 系列 FPGA 是一种大容量、高性能的新一代现场可编程门阵列器件。

Virtex-II FPGA 芯片上的硬件资源包括排成矩阵阵列的可构造逻辑模块 CLB，四边的输入/输出模块 IOB 之间的数字时钟管理器 DCM 模块；每隔 4 列或 6 列 CLB(依器件 CLB 的数量而定)，排列着一列可选择 RAM 模块和一列专用乘法器模块，可为有源互连技术的新一代可编程布线资源，将上述所有元件连接在一起。布线矩阵 GRM(General Routing Matix)是一个布线开关阵列，每一个可编程元件接一个开关矩阵，由开关矩阵和 GRM 互连。VirtexII 的结构图如图 3.1 所示。

(1) 可构造逻辑模块(CLB)，与其他 FPGA 一样，也是 Virtex-II 的基本单元，由此构成组合电路和时序电路；

(2) 18K 位可选择 RAM 模块；

(3) 嵌入式乘法器，每列 RAM 模块旁边，排列着一列 18 位×18 位的嵌入式乘法器模块；

(4) 全局时钟多路缓冲器，数字时钟管理器、输入输出模块及有源互连技术等。

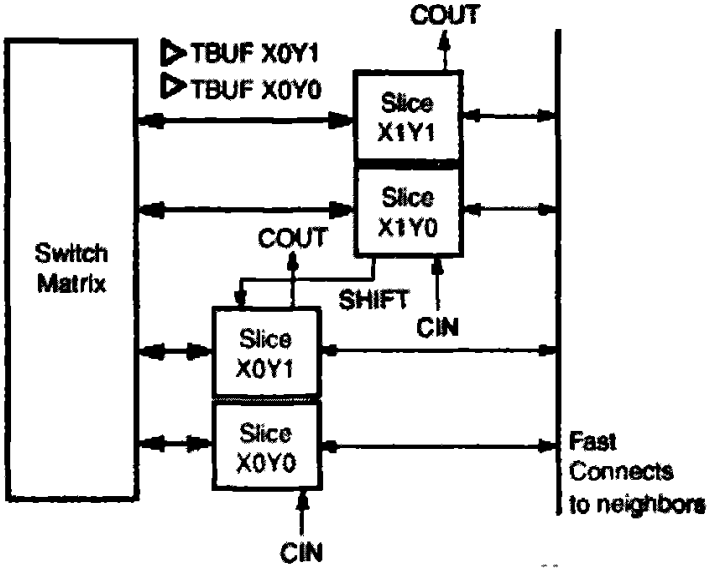


图 3.1 Virtex-II 结构图

Fig. 3.1 The structure of Virtex-II

3.2.2 FPGA 软件平台简介

目前，FPGA 的常用开发工具软件大体上可分为两类：一类是由 FPGA 芯片厂商直接提供的开发软件，另一类是由专业的 EDA 软件开发公司提供的第三方软件。本项目采用的方式是 Xilinx 的集成开发环境 ISE 上进行代码的开发、下载，用仿真工具 ModelSim 进行测试。下面具体介绍这两种软件的特点。

(1) ISE 平台

ISE 是 Xilinx 公司提供的集成化 FPGA 开发软件。在 ISE 中，项目导航器(Project Navigator)是项目管理工具的主体，集成了设计过程中要使用的一系列软件工具，主要包括设计输入(Design Entry)、设计综合(Design Synthesis)、设计约束(Design

Constraints)、设计实现(Design Implement)、设计仿真(Design Simulation)和器件编程(Device Programming)，如图 3.2 所示。

设计输入主要采用硬件描述语言(HDL)、原理图编辑器(ECS, Engineering Schematic Capture)和有限状态机(FSM, Finite State Machine)三种方式。其中硬件描述语言是最为流行的输入方法。

设计综合则是依据逻辑设计描述和约束条件，利用开发工具对设计目标进行优化处理，将 HDL 文件转变为硬件电路实现方案。

设计仿真根据设计阶段不同可分为 RTL 行为级仿真、综合后门级仿真和时序时延仿真 3 大类型。行为级仿真在设计输入之后综合之前进行，主要用来检查代码中的语法错误以及代码行为的正确性；时序时延仿真在综合和布局布线之后进行，能够得到目标器件的详细时序时延信息。

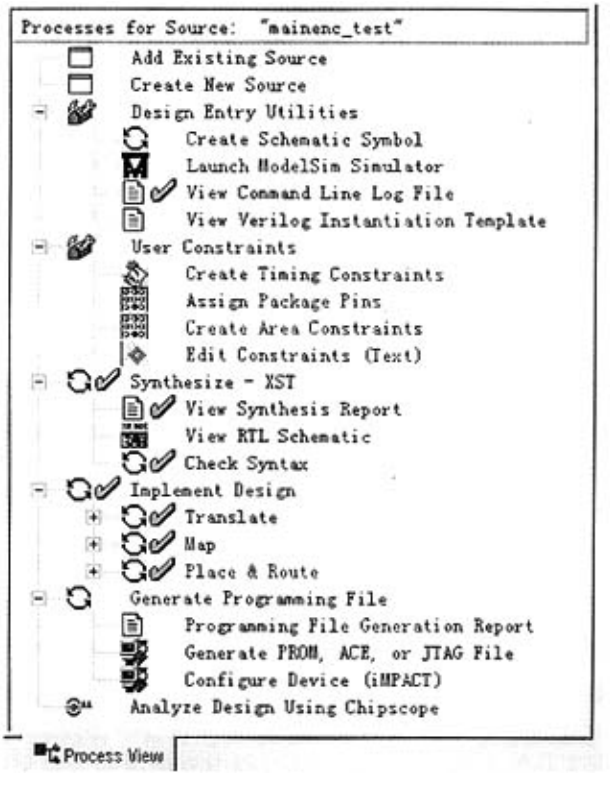


图 3.2　ISE 资源窗口

Fig. 3.2　The view of ISE source window

设计实现包括翻译(Translate)、映射(Map)和布局布线(Place and Route)三个步骤。

器件编程是指在功能仿真与时序仿真正确的前提下，将综合后形成的位流数据下载到具体的 FPGA、PROM 芯片中，又称为芯片配置。ISE 集成了强大的 iMPACT(the intelligent Multi-Purpose Programming And Configuration Tool)配置工具，可以配置 FPGA、CPLD 和 PROM。它支持多种配置模式，其中边界扫描模式最为常用，通过 JTAG 口，iMPACT 可以向 FPGA 下载 BIT 文件，向 CPLD 下载 JEDEC 文件，向 PROM 下载 EXO、MCS 等格式的 PROM 文件。

(2) ModelSim 软件工具

在 FPGA 自顶向下的设计流程中，为了保证设计的正确性，设计仿真包含在设计过程的每一环节中。Model Technology 公司(Mentor Graphics 的子公司)的 ModelSim 是业界较好的硬件描述语言仿真工具[30]。ModelSim 是一个独立的仿真工具，它在工作的时候并不需要其他软件的协助，在 Xilinx 公司的 ISE 集成开发环境中给 ModelSim 仿真软件预留了接口，通过这个接口可以从 ISE 集成环境中直接启动 ModelSim 仿真工具进行仿真。

ModelSim 仿真工具具有以下特点和优点:

① 仿真功能强大，图形化界面友好，ModelSim 6.1 版本在默认条件下具有 11 个窗口：即主窗口、结构窗口、源程序窗口、信号窗口、进程窗口、变量窗口、数据流窗口、波形窗口、存储器窗口、列表窗口和断言窗口。

② 采用直接编译技术，能大大提高 HDL 编译及仿真速度；支持单步调试、断点设置，以快速完成设计调试及验证。

③ 支持 RTL 行为级仿真、综合后门级仿真和时序时延仿真。

④ 可编写 HDL 激励文件或执行组模式进行仿真。

⑤ 具备性能分析与代码覆盖分析功能，利于发现设计瓶颈及方便调试。

3.3 FPGA 基本设计方法和设计流程

3.3.1 TOP-DOWN 设计方法

当构建一个系统时，工程师的最初设计思想是从功能描述开始的。设计工程师首先要考虑规划出能完成某一具体功能、满足自己产品系统设计要求的某一功能模块，利用某种方式(如 HDL 硬件描述语言)把功能描述出来，通过功能仿真(HDL 仿真器)以验证设计思路的正确性。当所设计功能满足需要时，再考虑以何种方式(即逻辑综合过程)完成所需要的设计，并能直接使用功能定义的描述，这就是自顶而下设计方法。

Top-Down 设计流程如图 3.3 所示，其核心是采用 HDL 语言进行功能描述，由逻辑综合(Logic Synthesis)把行为(功能)描述转换成某一特定 FPGA 的工艺网表，送到厂商的布局布线器完成物理实现。在设计过程的每一个环节，仿真器的功能验证和门级仿真技术保证设计功能和时序的正确性。与传统电路设计方法相比，Top-Down 设计方法具体有以下优点：

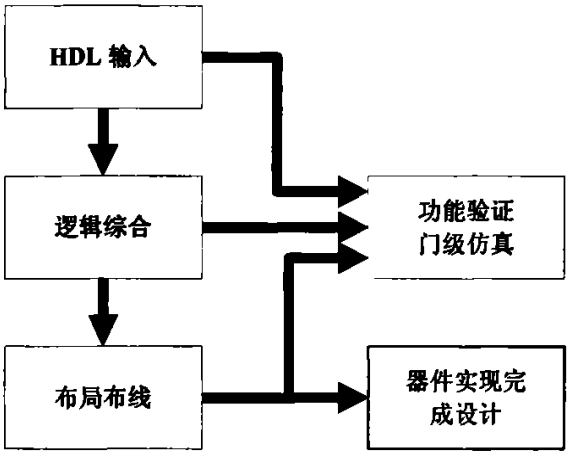


图 3.3 Top-Down 设计方法

Fig 3.3 Top-Down design method

(1) 完全符合设计人员的设计思路，从功能描述开始，到物理实现的完成；

(2) 功能设计可完全独立于物理实现；

(3) 设计可再利用；

(4) 易于设计的更改；

(5) 便于设计、处理大规模、复杂电路；

(6) 设计周期缩短，生产率大大提高，产品上市时间提前，性能明显提高，产品竞争力加强。据统计，采用 Top-Down 设计方法的生产率可达到传统设计方法的 2 到 4 倍。

3.3.2 FPGA 的设计流程

FPGA 设计可以分为设计输入、综合、功能仿真(前仿真)、设计实现、时序仿真(后仿真)、配置下载等六个步骤。下面将结合 ISE 软件的特点来具体介绍一下 FPGA 的开发流程。

(1) 设计输入

设计输入包括使用硬件描述语言 HDL、状态图与原理图输入三种方式。HDL 设计方式是现今设计大规模数字集成电路的较好形式。HDL 语言描述在状态机、控制逻辑、总线功能方面较强，使其描述的电路能在特定综合器作用下以具体硬件单元形式较好的实现；而原理图输入在顶层设计、数据通路逻辑、手工最优化电路等方面具有图形化强、单元节俭、功能明确等特点。常用方式是以 HDL 语言为主，原理图为辅，进行混合设计以发挥二者各自特色。在进行混合设计时，设计者应当严格遵循自顶向下和自底向上的结构化设计方法。自顶向下是指设计小组在设计之初对系统进行充分的分析，明确技术条件指标，并将这些指标提炼为算法，转化为结构描述，然后将系统划分为容易实现的子系统，划分之后在进行时序调度和资源分配，不断深入，直至最终解决问题。自底向上则是指设计小组在系统划分的基础上按模块进行设计。每个小组成员独立的完成各自的模块设计，进而构成整个 FPGA。小组成员中可能各自习惯的 HDL 语言不同，但对整体而言并无影响，模块在调用时只需要将被调用模块视作黑箱，无需关心其具体设计。

对于 ISE 来说，底层模块可以采用 Verilog HDL 或 VHDL 语言描述，上层模块可以考虑使用图形方式描述。

(2) 设计综合

综合就是针对给定的电路实现功能和实现此电路的约束条件，如速度、功耗、成本及电路类型等，通过计算机进行优化处理，获得一个能满足上述要求的最优或者接近最优的电路设计方案。综合包括分析、综合和优化三个步骤。以 HDL 描述为例，分析是采用标准的 HDL 语法规则对 HDL 源文件进行分析并纠正语法错误；综合是以选定的 FPGA 结构和器件为目标，对 HDL 和 FPGA 网表文件进行逻辑综合；优化则是根据用户的设计约束对速度和面积进行逻辑优化，产生一个优化的 FPGA 网表文件，以供 FPGA 布局和布线工具使用。综合与优化可以分两步独立进行，在两步之间进行约束指定，如时钟的确定、通路与端口的延时、模块的算子共享、寄存器的扇出等。如果设计模型较大，可以采用层次化方式进行综合，先综合下级模块，后综合上级模块。在进行上级模块综合时设置下级模块为 Don't Touch，使设计与综合过程合理化。综合后形成的网表可导入 FPGA 设计厂商提供的可支持第三方设计输入的专用软件中，就可进行后续的 FPGA 芯片的实现。综合完成后可以输出报告文件，列出综合状态与综合结果，如资源使用情况、综合后层次信息等。

在 ISE 中，综合模块(Synthesis)负责综合的过程。

(3) 仿真验证

从广义上讲，设计验证包括功能与时序仿真和电路验证。仿真是指使用设计软件包对已实现的设计进行完整测试，模拟实际物理环境下的工作情况。前仿真是指仅对逻辑功能进行测试模拟，以了解其实现的功能是否满足原设计的要求，仿真过程没有加入时序信息，不涉及具体器件的硬件特性，如延时特性；而在布局布线后，提取有关的器件延迟、连线延时等时序参数，并在此基础上进行的时序仿真称为后仿真，它是接近真实器件运行的仿真。

由于 ISE 的仿真功能相对较弱，当安装 ModelSim 后，可以设置好输入参数调用 ModelSim 仿真器进行仿真。

(4) 设计实现

实现是利用 FPGA 厂商的实现工具把综合后逻辑映射到目标器件结构的资源中，决定逻辑的最佳布局，选择逻辑与输入输出功能连接的布线通道进行连线，并产生相应文件(如配置文件与相关报告)。通常可分为如下五个步骤。

① 转换(Translate)：将多个设计文件进行转换并合并到一个设计库文件中。

② 映射(Map)：将网表中逻辑门映射成物理元素，即把逻辑设计分割到构成可编程逻辑阵列内的可配置逻辑块与输入输出块及其它资源中的过程。

③ 布局与布线(Place and Route)：布局是指从映射取出定义的逻辑和输入输出块，并把它们分配到 FPGA 内部的物理位置，通常基于某种先进的算法来完成；布线是指利用自动布线软件使用布线资源选择路径试着完成所有的逻辑连接。可以使用约束条件操作布线软件，完成设计规定的性能要求。在布局布线过程中，可同时提取时序信息形成报告。

④ 配置(Configure)：产生 FPGA 配置时的需要的位流文件。

在 ISE 中，以上功能都可以在 Process 窗口的下拉菜单中实现。

(5) 时序分析

在设计实现过程中，在映射后需要对一个设计的实际功能块的延时和估计的布线延时进行时序分析；而在布局布线后，也要对实际布局布线的功能块延时和实际布线延时进行静态时序分析。静态时序分析允许设计者详尽地分析所有关键路径并得出一个有次序的报告，而且报告中含有其它调试信息。静态时序分析器可以用来检查设计的逻辑和时序，以便计算各通路性能，识别可靠的路径，检测建立和保持时间的配合。

最终的时序分析也由 ModelSim 完成， ISE 也可以进行一些辅助的分析。

(6) 下载验证

下载是在功能仿真与时序仿真正确的前提下，将综合后形成的位流下载到具体的FPGA芯片中，也叫芯片配置。因FPGA大多支持IEEE的JTAG标准，所以使用芯片上的JTAG口是常用下载方式。将位流文件下载到FPGA器件内部后进行实际器件的物理测试即为电路验证，当得到正确的验证结果后就证明了设计的正确性。电路验证对ASIC最后的投片生产具有较大意义。

下载工具可由ISE的生成可执行文件(Generate Programming File)负责。它会生成可执行的二进制文件，通过JTAG口将其下载到硬件芯片中。同时应注意正确配置相应的引脚。

第四章 CCSDS 编码器的 FPGA 设计与实现

4.1 CCSDS 编码器的整体结构设计

本文我们采用 Xilinx 公司 Vertex-II 系列的 XC2V2000 芯片。在算法的总体规划中，首先要考虑片上资源的限制。该芯片片上内存为 1Mbits，10K 的 Slices，考虑到 CCSDS 图像算法的复杂度和中间数据的存储，暂时把输入图像暂时定义成 32×32。由于该算法是对小波变换后的系数再进行分段，段的大小可以人为指定。对于大幅图像来说，只要有足够的存储空间，通过修改上层的模块即可做到对大图像的压缩。

CCSDS 图像压缩算法的编码系统功能框图如图 4.1 表示，采集到的原始图像数据由串口输入到片内的 RAM 中，再将此 RAM 中的数据送至编码模块，编码模块完成编码功能，将编成的码流送入码流缓冲区。最终将码流缓冲区的输入由串口送回 PC 端进行验证[31]。

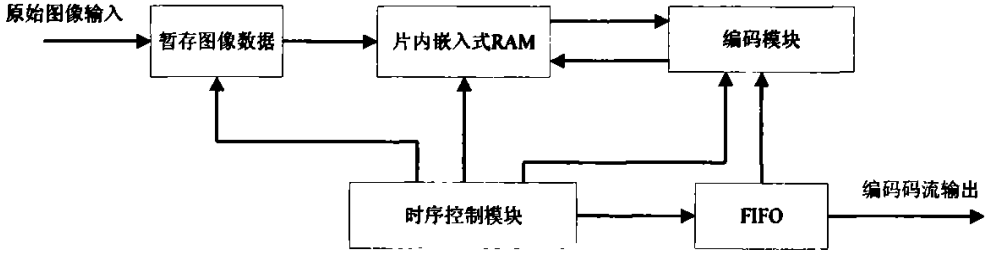


图 4.1 编码系统功能框图

Fig. 4.1 Functional schematic of coding system

4.1.1 内存设计

内存的设计和应用是实现 CCSDS 图像压缩算法的关键技术之一。XC2V2000 芯片共有片内内存 1Mbits。由于 ISE 已经把设计好的内存样式以 IP Core 的形式集成到开发环境中，我们只需调用 IP Core 即可完成内存的实现。IP Core 共提供给我们 3 种适合的内存样式：分布式内存、单端口内存、双端口内存。分布式内存分配较灵活，但不能实现太大块的内存[32]。单端口内存在一个时钟周期只能读或只能写，读写不能同时进行。双端口内存则在一个时钟周期内可以进行同时读写。实际应用时，由于受到片上 slice 资源的限制，很多数据不能全部存放到寄存器中，需要把中间的数据存入内存。为了减少频繁访问内存所带来的开销，故选择双端口内存作为设计的方式。调用 IP core 后生成的内存配置如图 4.2 所示。

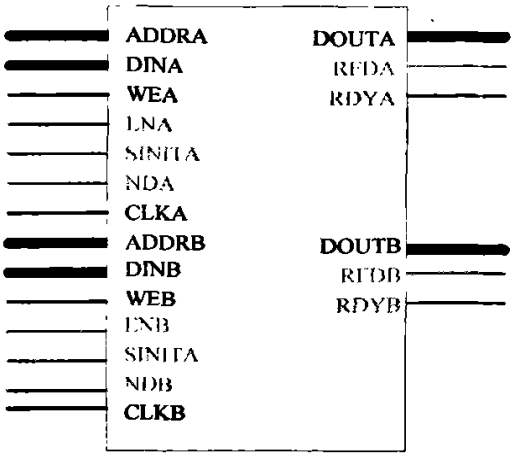


图 4.2 双端口内存配置图

Fig. 4.2 Dual-port memory configuration

4.1.2 打码流模块设计

CCSDS 另一个重要的模块就是打码流模块，其作用是把 CCSDS 编码器编出的码字写入码流缓冲区，之后再由外部设备读出并传递出去。由于每个码字不定长，且要保持码流数据的连续性，因此需要重点设计。C 代码提供的方法是按比特写入临时缓冲区，当缓冲区满时写入码流区域。这样虽然可以保证正确性和码流的连续性，但耗费了大量的指令周期，因此需要重新设计。本文采用的方法是扩大临时缓冲区的长度，将所打的码字按码字长度全部放入缓冲区，当缓冲区达到码流区域一个字长时再写入码流区域，同时将临时缓冲区进行调整以留出下一个码字空间。

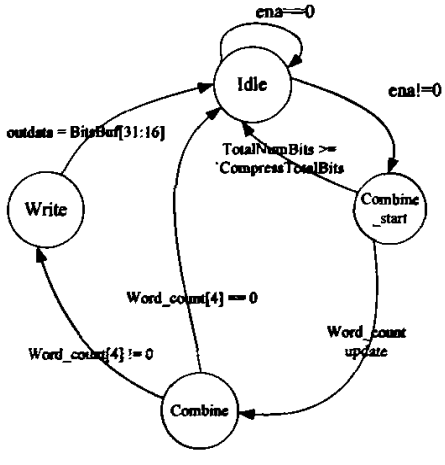


图 4.3 Putbits 状态转移图

Fig. 4.3 Putbits state transfer chart

Putbits 状态转移图如图 4.3 所示。第一个状态是 Idle 状态，该状态等待 ena 使能信号。当得到使能信号后，进入 Combine_start 状态。Combine_start 中，首先判断已打入的比特数是否已经满足压缩比特数的限制。如满足则返回 Idle 状态继续等待，如不满足则进行比特的组合，和相应的移位位数计算。接下来进入 Combine 状态，该状态进行比特的与操作，把缓存区的比特与新加入的比特拼接在一起。同时还要判断在比特缓存区的数据是否满足一个字的字长，满足一个字长则进入 Write 状态，否则返回 Idle 状态，并给出 done 信号。在 Write 状态中，需要把缓存区的高 16 位写入码流缓冲区中，同时更新比特缓存区和一些记录状态的信息，最后给出 done 信号。在本文的 FPGA 实现中，向码流缓冲区打入一个码字只需最多 4 个状态，也就是最多 5 个指令周期，大大减少了算法延时。在 CCSDS 算法中，需要多次调用打码流 Putbits 模块，该模块的高效执行对整体的优化贡献很大。图 4.4 是 Putbits 模块的时序仿真图。

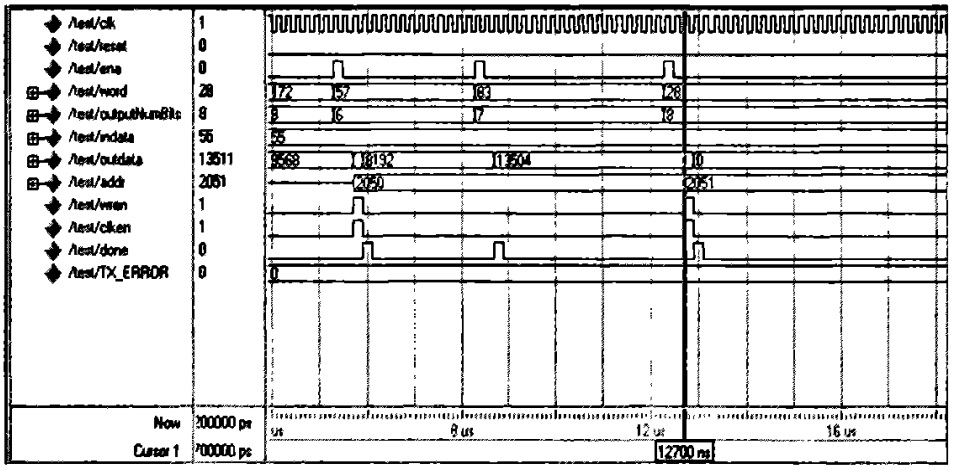


图 4.4 Putbits 仿真图

Fig. 4.4 Putbits simulation chart

4.2 各功能模块设计

CCSDS 编码系统设计的重点在于各个模块的算法移植。根据完成的功能不同，整个算法可分为离散小波变换模块、直流编码模块、交流比特深度编码以及位平面编码。其中位平面编码在每一个位平面上又可分为 stage0、stage1、stage2、stage3、stage4 等 5 个阶段。其编码流程如图 4.5 所示，接下来几节将详细介绍各编码模块的实现方式。

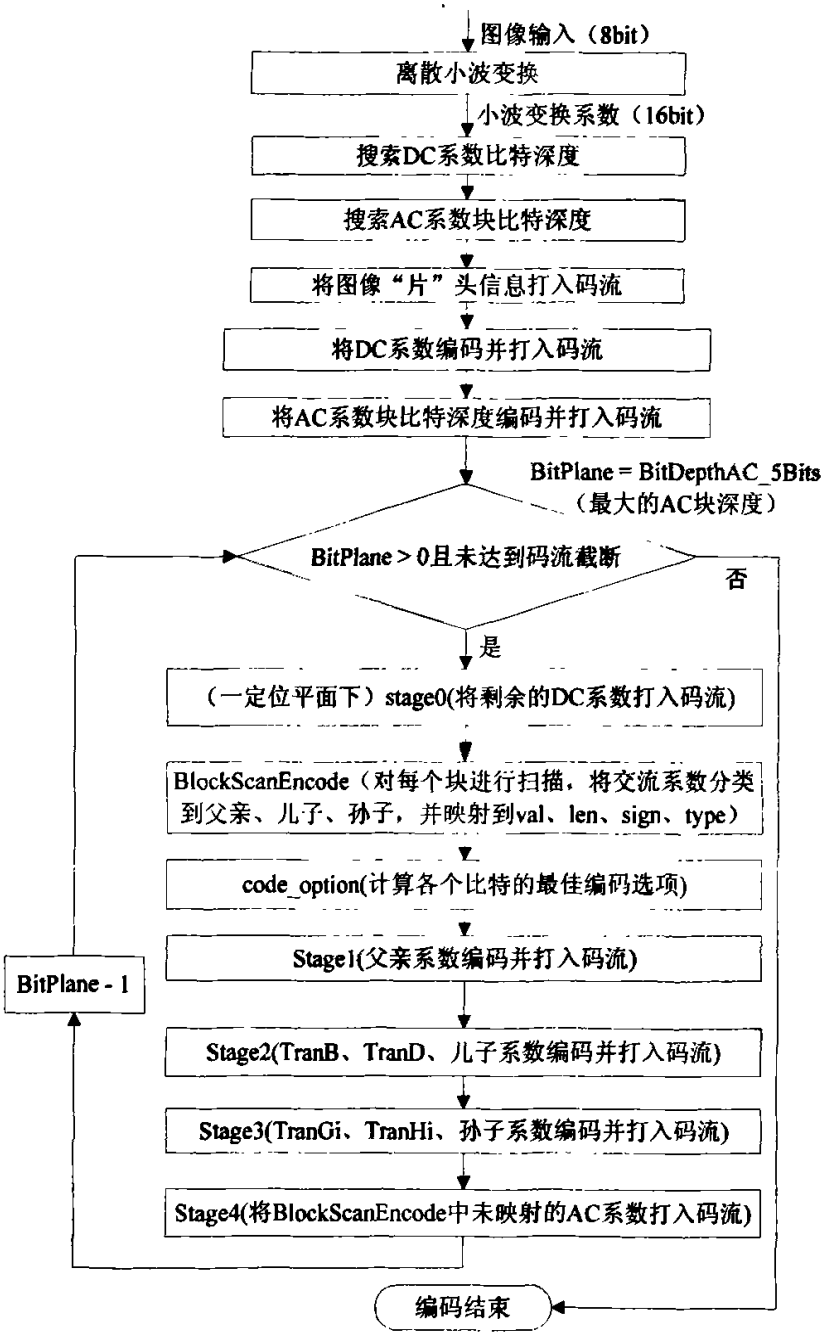


图 4.5　CCSDS 图像压缩算法的编码流程图

Fig. 4.5　Flow chart of CCSDS image compression algorithm

4.2.1 离散小波变换模块的设计

小波变换集中了 CCSDS 编码算法中的绝大部分计算，因此成为提高性能的关键，本文选取了一种并行度和稳定性相对较高的实现方案。本文采用了三级小波变换串行，每一级行列变换并行的方式。三级小波变换串行的结构比较简单，即顺次执行三级小波变换，每级变换结果经过地址映射后存入内存。

其中，行、列变换的设计方案分别如图 4.6，4.7 所示：

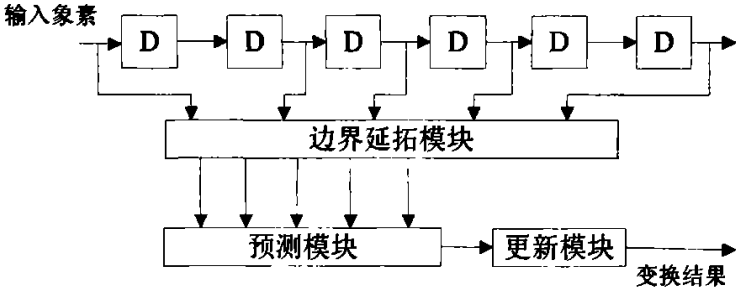


图 4.6 行小波变换的结构

Fig. 4.6 Structure of row DWT

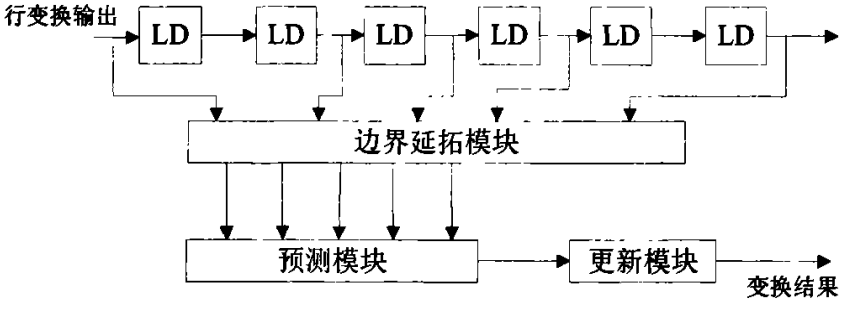


图 4.7 列小波变换的结构

Fig. 4.7 Structure of column DWT

最终的 DWT 算法的性能如表 4.1 所示。

表 4.1 DWT 算法的性能

Tab. 4.1 Performance of the DWT

最高工作频率(MHz)	Slices(个)
104.199	1620

4.2.2 直流编码模块的设计

直流编码模块由编码参数计算模块和编码模块两部分组成，如图 4.8 所示。其计算模块的各功能单元介绍如下：

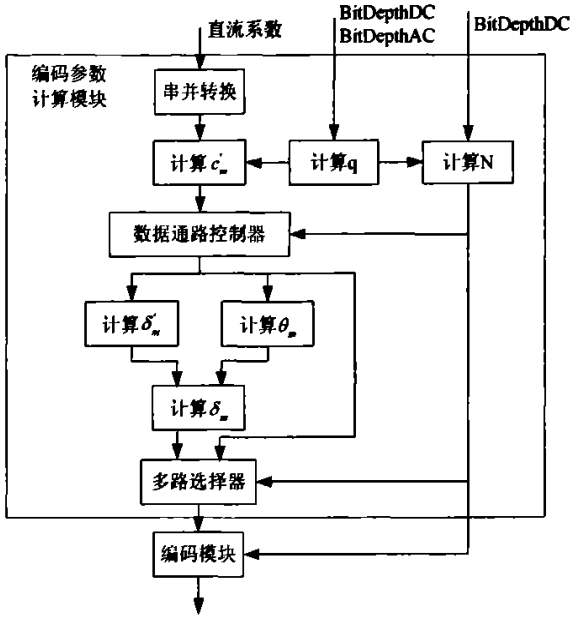


图 4.8 直流系数编码器框图

Fig. 4.8 Diagram of DC coder

串并转换单元将从内存读取的直流系数存放在寄存器组中。由于“段”的大小是 32×32，所以寄存器组的大小设置为 16。当 16 个直流系数读取完毕以后，随后的各功能单元可实现并行计算。

在对直流系数进行量化之前，首先要计算量化步长 q。其输入参数为直流系数比特深度和交流系数比特深度。然后利用量化步长 q 的值对寄存器组中的 16 个直流系数进行量化。由于可以并行计算，在一个时钟周期内就能得到量化后的直流系数 c_m' 序列。而 N 的计算是和量化步骤同时进行的，也只需一个时钟周期。

N 的值用来作为数据通路控制器的输入，当 N=1 时，直接对量化后的直流系数 c_m' 序列进行编码；当 N>1 时，则会选择另外一条计算通路。这条计算通路先计算量化后的直流系数 c_m' 序列的差分序列 $\delta_m' = c_m' - c_{m-1}'$，和 c_m' 序列的计算一样，也是在一个时钟周期内

完成 16 个元素的计算，同时并行计算参数θ_m。两步计算结果用来计算序列δ_m，16 个元素在一个时钟周期内得到结果。

多路选择器由参数 N 控制，当 N=1 时，选择量化后的直流系数$c_m^{'}$序列进行编码；当 N>1 时，对δ_m序列进行编码。

编码模块的框图如图 4.9 所示，通过参数 N 对两条数据通路进行选择：当 N=1 时，按图 2.12 的编码格式对量化后的直流系数$c_m^{'}$序列进行 uncoded 编码；当 N>1 时，先计算编码选项标识符 k，为了减少计算量，一般采取次最优的方法，即启发式方法，然后按图 2.13 的编码格式对δ_m序列进行变长编码。

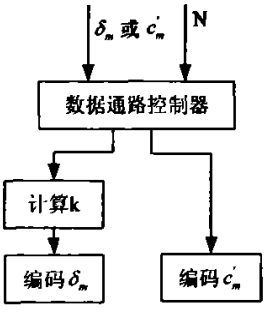


图 4.9　编码模块框图

Fig. 4.9　Diagram of coder module

最终的编码码流需要写入内存，每编码一个参数就写一次内存，所以编码模块并没有像编码参数计算模块那样采取并行的方式，而是以串行化的方式进行，这样能节省资源。表 4.2 给出了实现直流编码模块的性能数据，包括最高工作频率和占用的 slices 数目。

表 4.2　DCencoder 算法的性能

Tab. 4.2　Performance of the DCencoder

最高工作频率(MHz)	Slices(个)
101.164	965

4.2.3　位平面编码模块的设计

位平面编码主要由如下几个部分组成：首先将参数 *BitDepthAC_5Bits* 作为位平面编码开始的起点。之后判断该位平面的直流系数是否已打入码流，如还有未编码的直流系

数，则调用 stage0 模块将剩余的直流系数打入码流。stage0 的实现比较简单，主要是调用上文提到的 Putbits 模块即可。需要注意的是由于已知该码字的长度为 16 比特，因此先对要打的比特进行打包后再进行打码流操作比较节省时间。

接下来是比较大的模块 BlockScanEncode，该模块的作用是对小波系数进行扫描。扫描的顺序是按照小波变换块内该比特对图像恢复的贡献度的大小进行的。依照第二章所描述的，首先是扫描“双亲”系数，然后是“孩子”系数，最后是“孙子”系数。其位置见图 4.10 所示。扫描的同时对该位平面的比特按组映射到 *Val*、*Len*、*Sign*、*Type* 四个关键字中，作为位平面编码的基础。

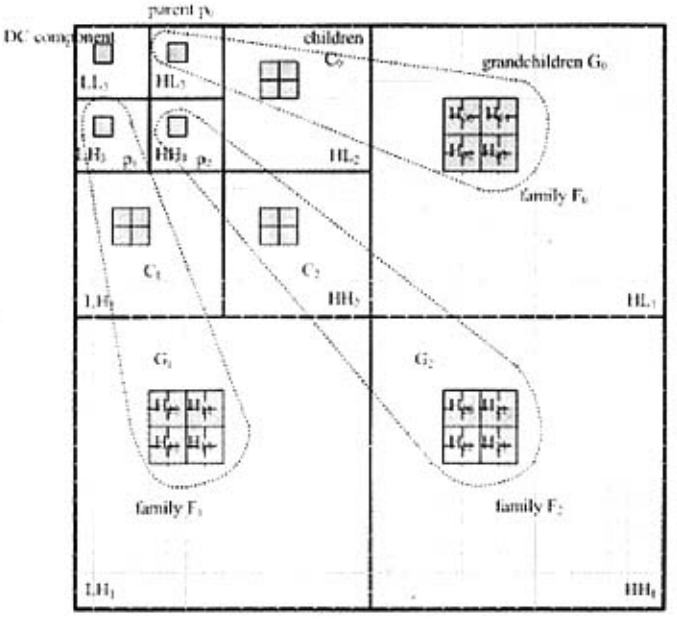


图 4.10　BlockScanEncode 模块系数图解

Fig. 4.10　Schematic of module BlockScanEncode

“双亲”系数在图中的 P0、P1、P2 三个位置上。首先要判断当前三个位置是否已经编码，这个信息存储在 *HitHistoryTypeP* 中。每次调用 BlockScanEncode 模块时需要把这个信息从内存中读出、判断、更新，然后再存回内存中，这样就可以保证其信息正确性。在映射的过程中，*Type* 的值为宏定义 ENUM_TYPE_P，其值为 1。*Len* 则为该位平面未编码的比特个数，如都未编码则 *Len* = 3。*Val* 是 P0、P1、P2 所组合的值。*Sign* 是 P0、P1、P2 符号所组合的值。这样就完成了 Parents 位置的系数映射。

在映射 Children 系数之前需要计算一些转移字符的值，由于 CCSDS 处理的对象是空间数据，经过小波变换后在某一位平面上会出现许多连零的情况。因此需要事先计算后面的数据是否全为零，若全为 0 则可以跳过此位平面，只需给一个全零的标志即可。在具体实现中，在计算“孩子”系数和“孙子”系数时，需要做类似的运算，即判断某一块是否全为 0，故一次遍历将后面的转移字符一并进行计算，存入 BitFlag 寄存器中供下面的模块使用。转移字符的计算是调用 BlockCodingCalcAllFlag 模块实现的，该模块将后续的小波系数进行遍历，每连续的 4 个比特给出是否全为 0 的标志。由于频繁访问内存需要浪费大量的指令周期，这样实现后可以减少对内存的访问次数。得到了所有的 BitFlag 标志后，即可得出 TranB 和 TranD。TranB 的值为 1 说明该位平面上“孩子”、“孙子”系数的值不全为 0，然后进行 TranD 的计算。对于 TranD，*Val* 的值为 3 比特，每一比特对应一个家庭。如果某一比特为 1 则说明该“家庭”在当前位平面上有不为 0 的值。如图 4.11 是 TranD 模块的时序仿真图。

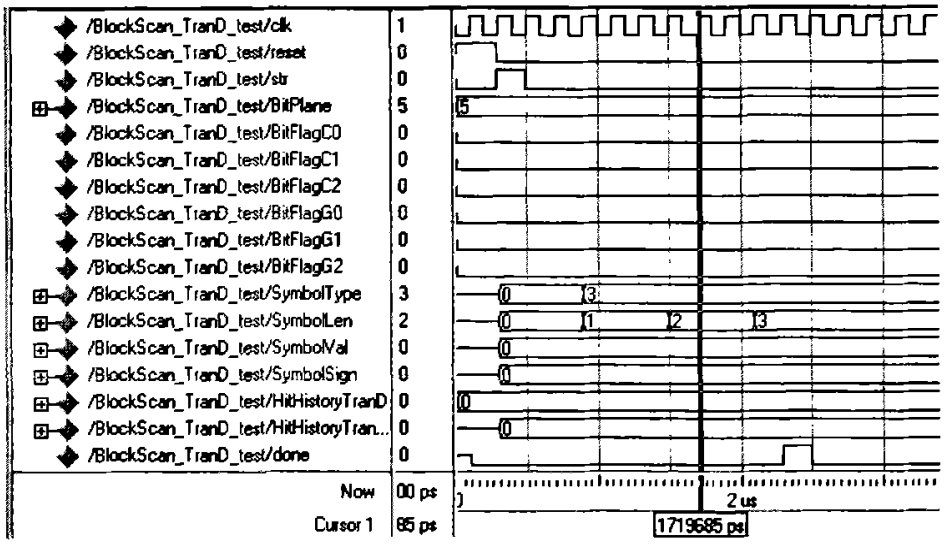


图 4.11 TranD 模块仿真图

Fig. 4.11 Module TranD simulation chart

然后来是扫描“孩子”系数，首先要读出标志位 *HitHistoryTypeCi*，该标志位的作用同 *HitHistoryTypeP* 类似，是判断该系数是否已经在以前的位平面上编码。如果已经编码则把该位平面的值存入寄存器 *ChildrenRefSymbol* 中去，这些信息将在 stage4 中打入码流。对于“孩子”系数的映射，*Type* 的值为宏定义 ENUM_TYPE_CI，*Val* 的值为在当前位平面未经过编码的系数的组合，*Sign* 的值是这些系数的符号位，*Len* 则为未编

码系数的个数。得到这四个关键字的同时，还需要更新标志寄存器的值。如图 4.12 是 TypeCi 的模块的时序仿真图。

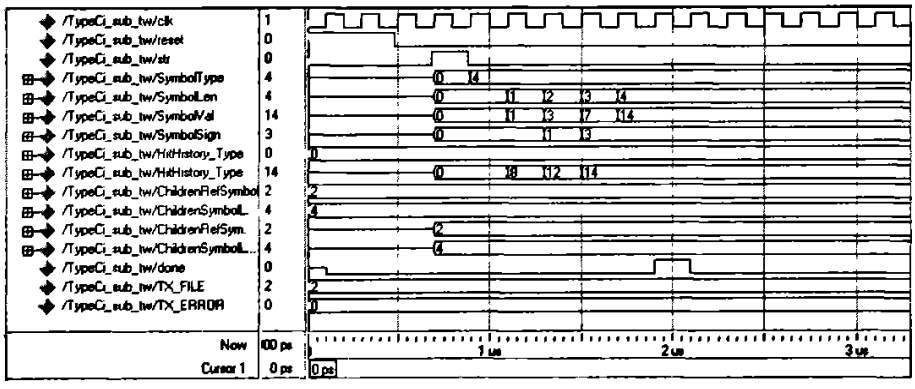


图 4.12　TypeCi 模块仿真图

Fig. 4.12　Module TypeCi simulation chart

接下来是“孙子”系数的映射。由图可知，对于块中的一个“家庭”，“孙子”系数有 16 个之多。因此需要对“孙子”系数进行更细的划分，具体方法是把 16 个系数分成连续的 4 组，记作 H_{ij}。i 表示第几个家庭，j 表示该家庭的某个“孙子”系数。划分后，我们首先对“家庭”信息 i 进行映射。其具体方法与 TranB、TranD 类似，计算该“家庭”的“孙子”系数是否全部为 0，记录在 4 个关键字中。关键字 *Val* 共有 4 比特，分别对应每个“家庭”划分的 4 组。由于映射的是转移字符，*Sign* 的值是始终为 0 的。如图 4.13 是 TranHi 的模块的时序仿真图。

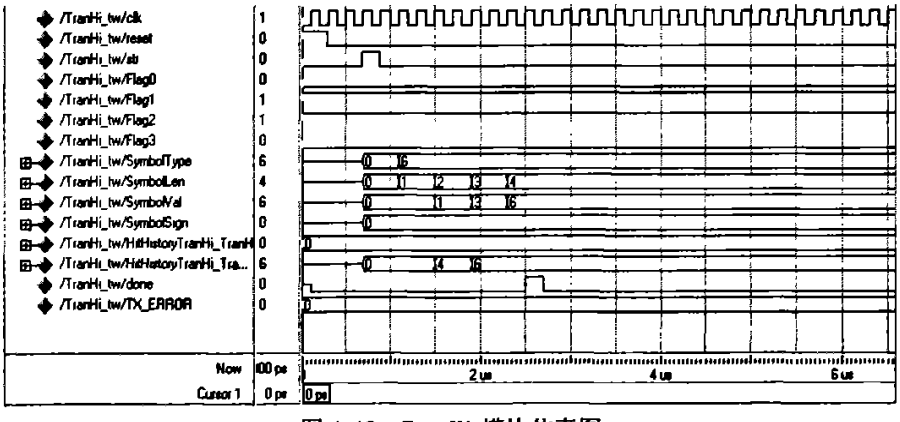


图 4.13　TranHi 模块仿真图

Fig. 4.13　Module TranHi simulation chart

最后根据 TranHi 的值来映射“孙子”系数。当 TranHi 的某一比特为 0 时，则说明要映射的孙子系数的 4 个关键字全为 0，由于该信息已经在 TranHi 中体现，就不需要计算。“孙子”系数的计算方法与“孩子”系数类似，但计算量更大，因此我们定义了一组寄存器来存储中间传递信息的数据。值得注意的是，由于 CCSDS 算法的特点，“孙子”系数是图像的高频部分，其系数值连续为 0 的情况非常多，实际映射的比特数较少。因此在 Verilog 语言实现时应尽量现根据 TranHi 判断“孙子”系数的情况，如不需要计算则尽快结束模块调用，减少算法的时延。对于“孙子”系数来说，*Type* 的值为 ENUM_TYPE_HIJ，*Len* 的值为在当前位平面未编码系数的个数，*Val* 的值为这些系数的组合，*Sign* 则为系数的符号位。如果该位平面上有非 0 系数，且已经在之前位平面上编码的则编入 *GrandChildrenRefSymbol* 中，并在 stage4 中打入码流。如图 4.14 是 TranHi 的模块的时序仿真图。

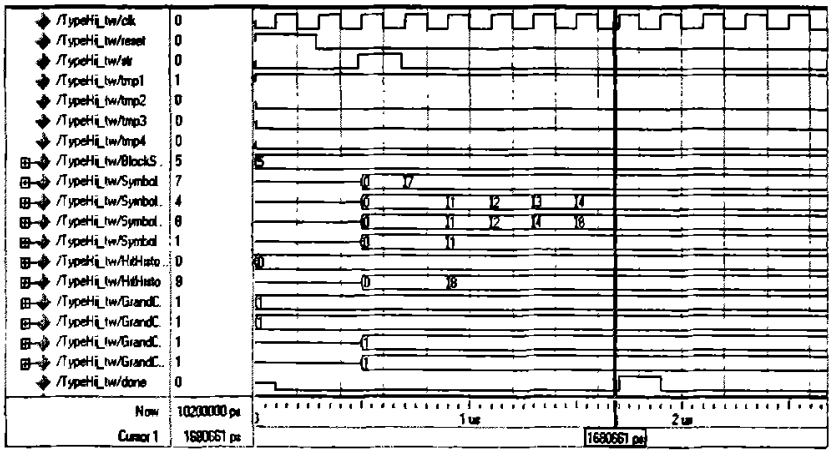


图 4.14 TypeHij 模块仿真图

Fig. 4.14 Module TypeHij simulation chart

当把块中的所有系数映射完毕后，还需要把在本位平面更改的状态信息(如是否系数已经在位平面上编码)存回内存，以备下次进入 BlockScanEncode 模块时参考。这个功能由 SaveBlockData 模块实现，实现方法即将标志寄存器的值存入内存相应的位置即可。最后 BlockScanEncode 模块的性能如表 4.3 所示。

表 4.3 BlockScanEncode 模块的性能

Tab. 4.3 Performance of the module BlockScanEncode

最高工作频率(MHz)	Slices(个)
176.866	2112

将块系数扫描映射到4个关键字后，我们需要把这些信息编码到码流中去。为了提高压缩的倍数，我们还需要利用一些先验的知识，按照出现的概率大小分配不同长度的码字，并且选择一组最佳的编码方式，使总编码长度最短。这些功能都在 code_option 模块中实现。code_option 中的 PROCESS_map 状态进行码字的映射，即把 *Val* 映射到 *Pattern*。之后 COUNT_bit 状态计算采用不同 option 时码字的长度。最后 OPT_sel 状态比较几种 option 总长度的大小，并给出 *Option_0*、*Option_1*、*Option_2* 分别对应2比特、3比特、4比特最佳的编码选项。

有了映射后的 *pattern* 值和编码选项，我们就可以根据图 2.11-2.13 进行查表得到要打入码流中的码字。在具体实现时，如果按照表中的方法直接对应的话，硬件则会生成一个很大的译码器，占用很多 slices 资源，并且较复杂的结构还会导致不可预知的布局布线延迟，不利于系统的稳定性。为此，我们从映射的关系着手，找出两者在每一比特的相关性，尽量用与或操作和比特拼接操作完成，这样既节省了资源又提高了性能。

```
case(MappedPattern)
4'd0, 4'd1:
begin
    codeword[1:0] <= {1'b1,MappedPattern[0]};
    len<=4'd2;
end
4'd2,4'd3:
begin
    codeword[2:0] <={1'b0, MappedPattern[1:0]};
    len<=4'd3;
end
4'd4,4'd5:
begin
        codeword[3:0] <={1'b0,1'b0,1'b1, MappedPattern[0]};
    len<=4'd4;
end
4'd6,4'd7,4'd8,4'd9,4'd10,4'd11:
begin
      codeword[5:0] <={{3{1'b0}},{MappedPattern[1]^MappedPattern[2]},{~MappedPattern[1]},
MappedPattern[0]};
       len <=4'd6;
  end
  default:
  begin
   codeword[6:0] <= {{3{1'b0}},MappedPattern[3:0]};
```

```
len<=4'd7;
end
```

上例实现的是 4 比特长度码字，code_option 为 1 时的映射方法。可以看出对应不同的 *Symbol* 值，映射后 *len* 的长度分为 5 种情况。我们把每一种对应到一类映射的方式上，如当 *Symbol* 为 6 至 11 时，其映射的码字长度为 6 比特。前三个比特为 0，第四比特是 *Symbol* 第二比特与第三比特异或的值，第五比特是 *Symbol* 第二比特取反后的值，最后一比特是 *Symbol* 第一比特的值。这样我们就很方便的实现了这一部分的码字映射，而不需要很大的译码器。同理，表的其他部分也可以按照此方式实现。由于这个码字的映射过程在 stage1 至 stage3 都要用到，采用这种译码的方式大大提高了位平面编码的整体性能。

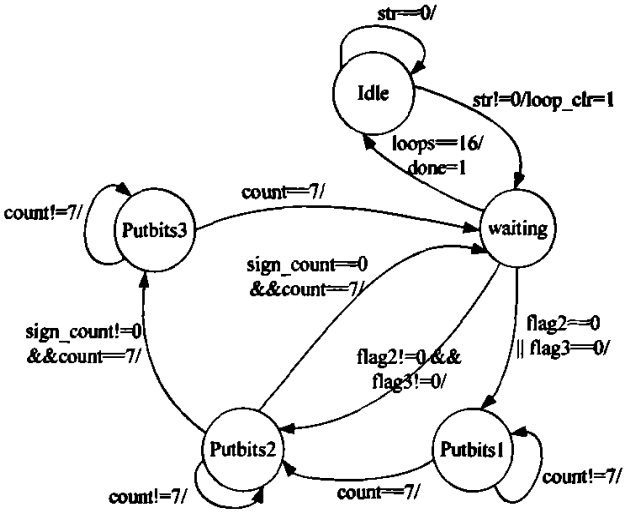


图 4.15　stage1 模块状态转移图

Fig. 4.15　Module stage1 state transfer chart

接下来要进行 stage1 到 stage3 的编码，在这三个部分将把“双亲”系数、“孩子”系数和“孙子”系数以及一些转移字符打入码流。首先进入的是 stage1 模块，该模块的状态转移图如图 4.15 所示。状态转移图中斜杠 (/) 左边的内容表示输入，右边的内容表示输出。由于 stage 编码需要较多的控制变量，所以有限状态机的输出也较多，在给出的状态转移图中并没有给出全部的输出变量，而只是给出了较关键的几个。stage 编码一共有 5 个状态，分别是 Idle，waiting，Putbits1，Putbits2 和 Putbits3。Idle 是空闲状态，在此状态下等待使能信号；waiting 是等待状态，主要是负责块编码字的计数以及码字的

映射，映射的方法即上面介绍的方法；Putbits1 状态将最优编码选项写入编码码流，该选项仅在第一次出现时打入码流，因此还需要寄存器存储该信息。我们分别用 flag2 和 flag3 分别表示 2 比特编码字和 3 比特编码字是否第一次出现，等于 1 表示已经出现，等于 0 表示还未出现。；Putbits2 状态将 $types_b[P]$写入编码码流；Putbits3 状态将 $signs_b[P]$写入编码码流。为了保证 putbits 模块能够结束，需要等待 7 个时钟周期，所以用一个计数器 count 来统计。sign_count 是一个块中符号的个数。状态转移过程描述如下：当使能信号 str 等于 1 时，跳转至 waiting 状态，并将 loop_clr 置 1，即将编码字计数器清零。如果编码字是第一次出现，则跳转至 Putbits1 状态编码最优编码选项参数；如果编码字已经出现过，则跳转至 Putbits2 状态，编码 $types_b[P]$参数。Putbits1 状态在写入编码码流以后无条件地跳入 Putbits2 状态。如果符号的个数不为 0，则从 Putbits2 状态跳入 Putbit3 状态，对符号进行编码，完成以后，无条件跳入 waiting 状态。在 waiting 状态检测编码字计数器 loops 是否为 16，即一个“群”是否编码完毕，如果是，则跳入 Idle 状态，等候下一个“群”的编码，并将 done 信号置成高电平，表示一个“群”的编码已经完成，否则继续下一个块的编码。

图 4.16 给出了 stage1 模块的时序仿真图，其中 str 是 stage1 模块的使能信号，done 信号是模块的结束标志。ena 标志是 putbits 模块的使能信号。word 和 length 是 putbits 模块的输入参数，分别对应码字和长度。表 4.4 是 stage1 模块的性能。

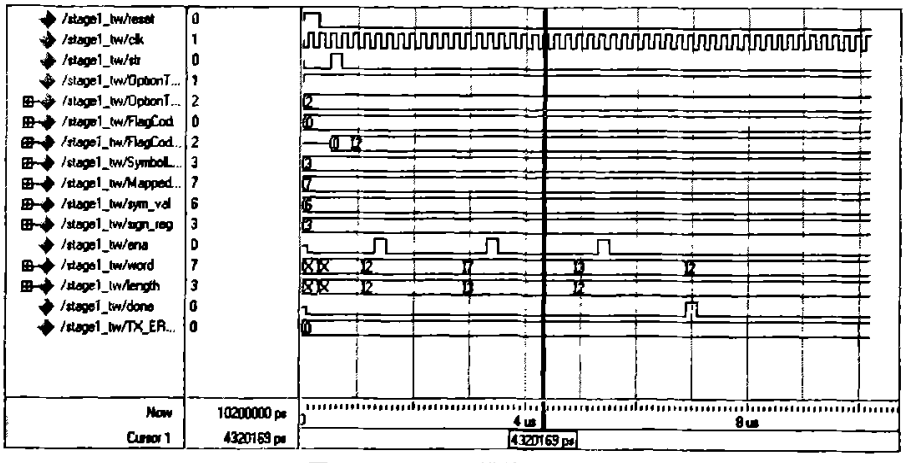


图 4. 16　stage1 模块仿真图

Fig. 4.16　Module stage1 simulation chart

表 4.4 stage1 模块的性能

Tab. 4.4 Performance of the stage1

最高工作频率(MHz)	Slices(个)
193.087	476

stage2 模块对象是“孩子”系数以及相关的转移字符 TranB、TranD。模块首先遍历 BlockScanEncode 模块计算的数据，如果 *Type* 类型为 ENUM_TRAN_B、ENUM_TRAN_D、ENUM_TYPE_CI 则进行映射成码字，打入码流。由于在 BlockScanEncode 模块中 *Type* 的值是递增的，因此当 *Type* 的值大于 ENUM_TYPE_CI 时则说明进入到“孙子”系数的范围内，可以结束遍历以节省时间。stage2 模块的整个流程与 stage1 类似，所不同的是需要调用到 4 码字长度的映射关系表，因此较 stage1 稍微复杂一些。模块时序仿真图与模块性能如图 4.17 和表 4.5。

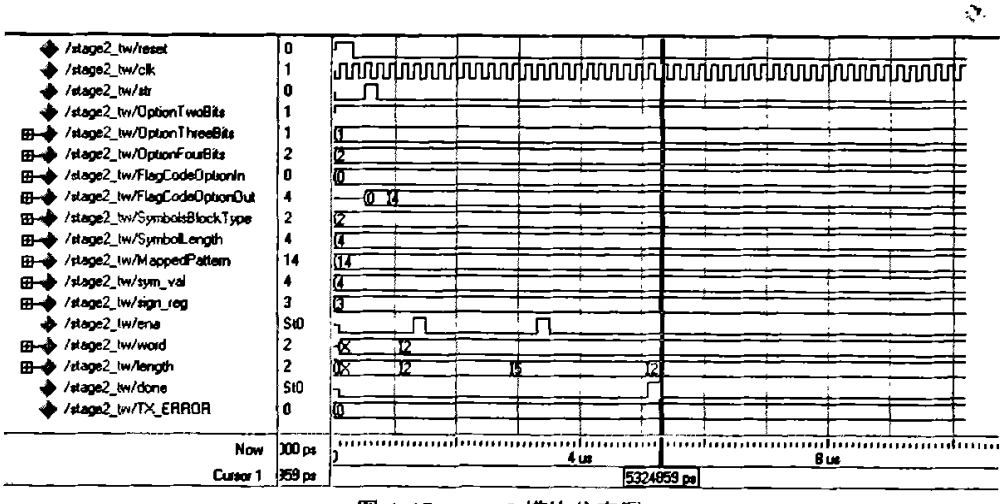


图 4.17 stage2 模块仿真图

Fig. 4.17 Module stage2 simulation chart

表 4.5 stage2 模块的性能

Tab. 4.5 Performance of the stage2

最高工作频率(MHz)	Slices(个)
198.531	511

stage3 模块对象是“孙子”系数以及相关的转移字符 TranGi、TranHi。模块同样要先遍历 BlockScanEncode 模块计算的数据，此时已编码的类型已经清 0。当遍历到 *Type*

类型为 ENUM_TRAN_Gi、ENUM_TRAN_Hi、ENUM_TYPE_Hij 时进行码字的映射并打入码流。stage3 模块的整个流程与 stage2 的原理相同，只是编码的对象不同而已。因为 *Type* 的值是按大小顺序排列的，实现 stage3 模块的重点是如何快速搜索到有效的数据，这可以通过快速的跳转实现，或者加入标志寄存器为 stage3 指向开始的位置。stage3 中实际打入码流的数据不多，其性能也与 stage2 相近。

最后我们给出了 CCSDS_encoder 模块，该模块按顺序调用上面介绍的各子模块，从而实现了整个 CCSDS 算法。经过后续对整体的进一步优化后，该算法的性能如表 4.6 所示。

表 4.6　CCSDS 算法的性能

Tab. 4.6　Performance of the CCSDS

最高工作频率(MHz)	Slices(个)
101.113	8968

第五章　系统验证及性能分析

整个编码系统的验证分为软件平台验证和硬件平台验证两部分。软件平台验证包括功能仿真和时序仿真，各个模块的功能仿真在设计各功能模块时完成，接下来将各个功能模块连接成一个系统，进行整体的功能仿真。在软件平台验证正确的前提下，再进行硬件平台验证，这部分工作需要在 Macro Blaze 开发板上进行，从而保证编码系统在实际硬件中的正常运行。

5.1　软件平台验证

软件平台验证是在 ModelSim 平台下进行的，其测试流程如图 5.1 所示。主要包括五个步骤：(1) 从外部文件读入图像数据；(2) 等待编码系统使能信号；(3) 数据进入 CCSDS 编码模块，完成编码后将完成信号置高电平；(4) 系统工作结束，将码流数据写入外部文件；(5) 比较输出数据与 C 语言程序的标准输出，验证系统结果的正确性。

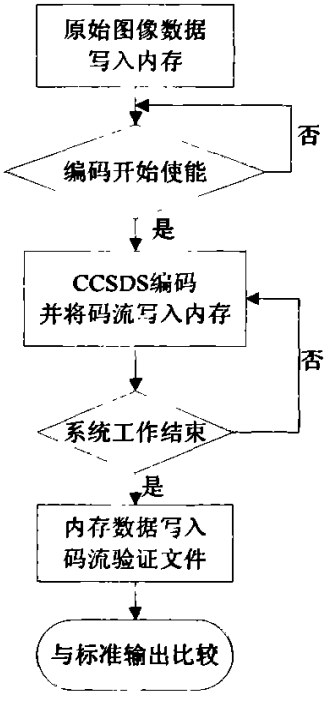


图 5.1　ModelSim 软件验证流程

Fig. 5.1　Flow of ModelSim software verification

由此可见，系统结果的验证最终归结于编码码流的比较，其要求是比特一致。也就是 ModelSim 软件平台下编码后产生的码流，应该与 VC6.0 平台下经过相应步骤产生的码流完全一致。比特一致的验证可由比较软件完成，其原理是将两幅图片按比特做差，其值为 0。由于软件运行时间较慢，并且软件仿真结果的正确并不能保证硬件平台的正确性，所以在软件平台上，只验证了数量较少的几幅测试图像。在保证几个典型的测试图像软件正确后，将重点放在硬件平台上。

5.2 硬件平台验证

5.2.1 硬件平台设计

硬件系统的验证与软件验证类似，只是平台从 ModelSim 上转变为下载到 FPGA 芯片上验证。其硬件验证系统如图 5.2 所示。

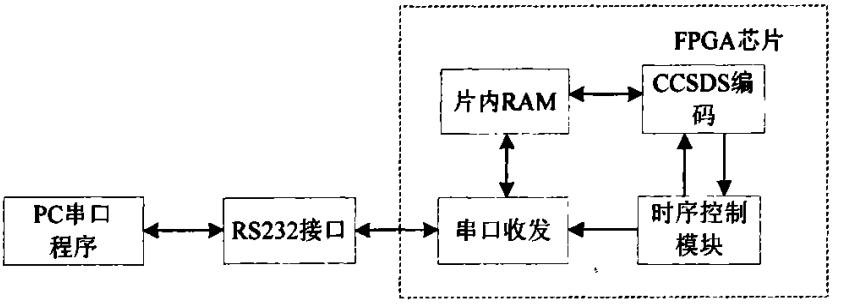


图 5.2 硬件验证系统

Fig. 5.2 Hardware verification system

硬件验证系统的具体工作过程分为 4 个主要部分：(1) 在 PC 端产生原始图像数据，通过 RS232 串口[33]传给 FPGA 开发板，写入内存；(2) CCSDS 编码模块完成图像数据的编码压缩；(3) FPGA 通过 RS232 串口把编码后的码流数据回传，由 PC 进行软件解码；(4) 解码得到的图像数据与标准 C 语言版本编码器的解码输出比较，达到验证目的。硬件验证平台与软件验证平台的最大区别就是增加了一个串口接收和发送模块，这个模块是 FPGA 开发板和 PC 机进行通信所必须的。这样做只需改变 PC 串口程序的输入图像文件名，重新编译 PC 机的程序，而整个 FPGA 编码系统只需完全编译一次，就能验证多幅图像，大大节省了验证时间。实验证明，完成一幅 256×256 图像的硬件验证工作不到一分钟，并且其中大部分时间是消耗在 RS232 串口数据的传输上。

为了使验证工作更加可视化，使用了 RS232 串口软件，如图 5.3 所示。它不仅可以设置各种传送速率的串口数据接收和发送，还提供数据的实时显示功能。为了不影响验

证速度，实际中并没有对接收的数据进行实时显示，而是直接写到临时文件中。CCSDS图像压缩算法的解码模块读取该临时文件中保存的码流数据，完成解码。

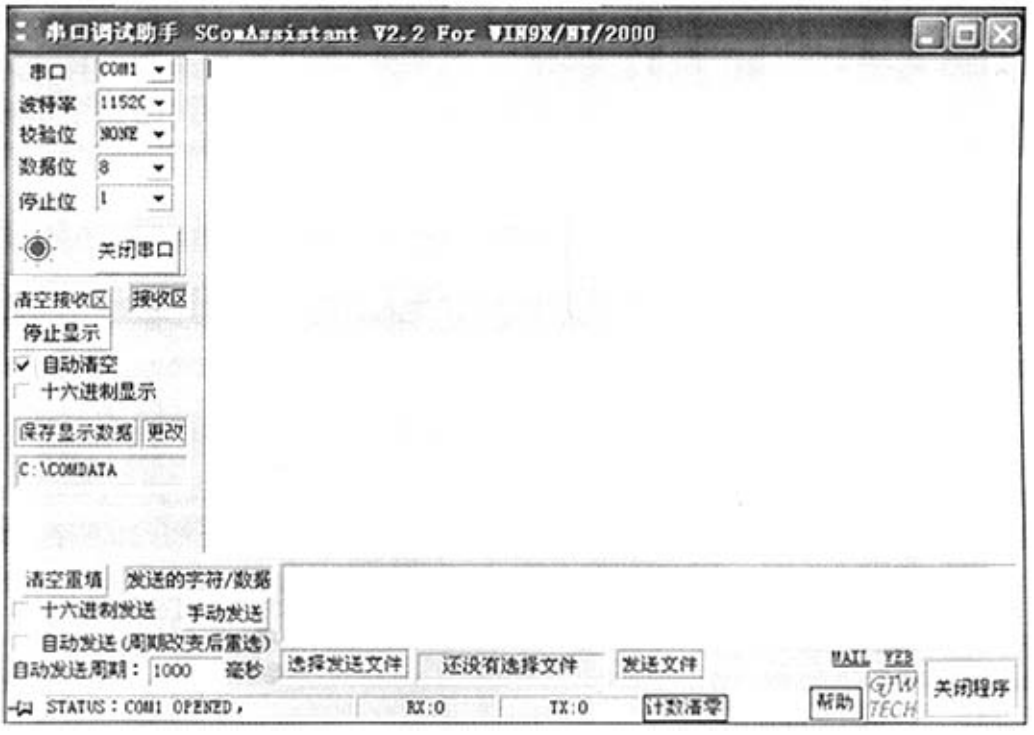


图 5.3 RS232 串口通信软件

Fig. 5.3 RS232 serial port communication software

5.2.2 硬件平台验证结果及分析

本文通过大量的图像数据验证最终硬件系统的正确性，主要包括常规图像和太空图像两类。这里给出一幅典型常规测试图像，256×256 像素、8 比特的 Lena 图像。如图 5.4 所示，(a)、(b)、(c)分别对应原始图像、经过 FPGA 编码再由软件解码的图像和经过 C 语言软件编解码的图像。另给出一幅太空图像，512×512 像素、8 比特深度的火星图像。如图 5.5 所示，其图像排列顺序与图 5.4 一致。从图 5.4 和图 5.5 可以看出，经过硬件编码的图像质量与软件的毫无差别。在所验证的大量图像中，FPGA 实现的 CCSDS 图像压缩系统能达到用软件实现图像编码的质量。

(a) 原始 Lena 图像

(a) Original Lena image

(b) 软件编码 Lena 图像

(b) Software coding Lena image

(c) 硬件编码 Lena 图像

(c) Hardware coding Lena image

图 5.4 Lena 图像硬件验证结果

Fig. 5.4 Hardware verification results for Lena

表 5.1 列出了 FPGA 系统中各个主要功能模块和整个系统的资源使用情况以及达到的性能，其中各模块的最大频率是在对该模块单独编译时所得的结果。总体编译时由于模块间的优化，性能还会提高一些。由于内存相互共享的原因，这里只给出小波变换和总体的内存开销。

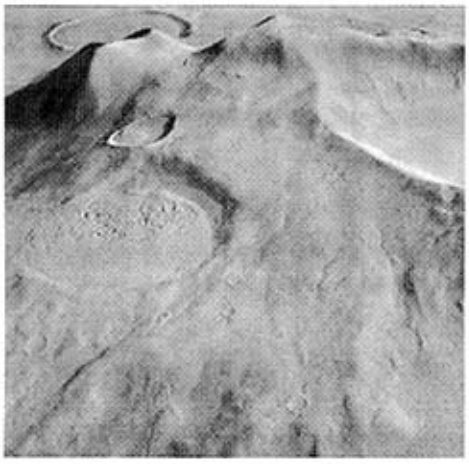
表 5.1 系统及各模块性能

Tab. 5.1 Performances for system and modules

模块名称	占用 slice 数目	使用 slice 的比率	最大频率(MHz)	片内内存(16bit)
DWT	1620	15%	104.199	2048
DCencoder	965	7%	101.164	-
BlockScanEncode	2112	19%	176.866	-
Stage0	191	1%	197.531	-
Stage1	476	4%	193.087	-
Stage2	512	4%	198.531	-
Stage3	512	4%	198.531	-
Stage4	425	3%	160.372	-
total	8968	83%	101.113	4096

(a) 原始空间图像 marstest

(a) Original space image marstest



(b) 软件编码空间图像 marstest　　(c) 硬件编码空间图像 marstest

(b) Software coding space image marstest　　(c) Hardware coding space image marstest

图 5.5 空间图像硬件验证结果

Fig. 5.5 Hardware verification results for space image

第六章 总结与展望

随着遥感技术及其应用的不断发展，卫星遥感图像越来越多地应用于国民经济和国防的各个领域。然而，由于图像数据传输往往受到数据传输带宽和星上功率的限制，遥感图像大量数据必须经过压缩才能适应图像传输和存储的需求，数据压缩技术在遥感领域也越来越受到重视。空间遥感图像分辨率高、空间局部相关性较弱、纹理丰富复杂，信息量大，码速率高，为适应遥感数据传输和存储的需要，对压缩技术的要求除压缩比和失真度外，特别要求实时性好，可靠性高。

本文根据遥感图像的特点和压缩要求，在研究了 CCSDS 空间压缩算法的基础上，参考已有的 C 代码，实现了在 FPGA 硬件平台上的代码移植，主要工作有：

(1) 深入研究了 CCSDS 图像压缩算法，分析了小波变换模块，直流、交流系数编码模块，位平面编码模块等。根据硬件平台的特点，修改已有的 C 代码以适于硬件实现。

(2) 将 CCSDS 图像压缩算法与 JPEG2000 和 SPIHT 进行了比较分析。分析结果表明，CCSDS 在压缩性能和硬件实现复杂度上取得了较好的平衡，适合空间数据的高速实时处理。

(3) 选用 Xilinx 公司的 XC2V2000 芯片为实现平台，利用硬件描述语言 Verilog HDL 实现 CCSDS 图像压缩算法，并对重要功能模块如位平面编码模块进行优化。利用双端口内存提高内存访问效率，优化打码流操作。根据位平面编码的特点，优化其码字映射表，在硬件上高速有效的实现。

(4) 在 ModelSim 上对该系统进行模块级和系统级的功能仿真、时序仿真和验证。设计 FPGA 与 PC 通信的串口模块，用于硬件验证系统，节约验证时间，提高工作效率。对该系统进行大量的硬件测试，验证了 FPGA 硬件实现的压缩编码系统的有效性和可靠性。

由于研究的时间相对较短，作者的知识储备不足，对 CCSDS 图像压缩算法的理解还有许多方面不够深刻。近期 CCSDS 组织又给出了最新的解释说明，有待继续学习和研究。对于 FPGA 在图像压缩算法实现方面，作者的经验相对较少，一些更高级的技巧还不能全部掌握，硬件资源占用相对较大。对于该算法进一步的研究和完善工作，作者仅提出一些个人粗浅的认识：

(1) CCSDS 图像压缩算法在 C 代码中有一定冗余，可以结合解码进一步进行修改。

(2) 在硬件实现时，应对 CCSDS 图像压缩算法引入流水线机制，以提高其并行性，充分利用 FPGA 的特点。

(3) CCSDS 图像压缩算法各个模块的 FPGA 设计并没有进行更深入的优化，如

BlockScanEncode 模块占用的资源较多，在性能上可以有更大的提高，可以在下一步的工作中进行完善。

(4) 提高各个模块的并发性，充分利用 CCSDS 图像压缩算法的特点以提高性能。

(5) 进行一些前端设计，如 IEEE1394 总线传输，以便实际使用时与高速摄像机实现数据互通。

参 考 文 献

[1] 张雪松，倪国强，周立伟. 基于 JPEG 标准实时图像编码系统的研究. 北京理工大学学报, 1998, 18(2):217-221.

[2] 张益真, 刘涛. Visual C++实现 MPEG/JPEG 编解码技术. 北京:人民邮电出版社, 2002.

[3] Santa-Cruz D, Ebrahimil T. JPEG2000 still image coding versus other standards[A]. SPIE' s 45th Annual Meeting, Applications of Digital Processing XXIII, Vol. 4115:446-454.

[4] Charilaos, Athanassios, Touradj. The JPEG2000 still image coding system:an overview. IEEE Transactions on Circuits and Systems for Video Technology, 2000, 46(4):1103-1127.

[5] Said A, Pearlman W A. A new, fast and efficient image codec based on set partitioning in hierarchical trees. IEEE Transactions on Circuits and Systems for Video Technology, 1996, 6(3):243-250.

[6] CCSDS. CCSDS 122.0-B-1, Image Data Compression. 2005.

[7] Pen-Shu Yeh, Armbruster P, Kiely A. The new CCSDS image compression recommendation. IEEE Conference on Aerospace, 2005:4138-4145.

[8] 任晓东, 文博. CPLD/FPGA 高级应用开发指南. 北京:电子工业出版社, 2003.

[9] 褚振勇, 翁木云. FPGA 设计及应用. 西安:西安电子科技大学出版社, 2002.

[10] 孙航. Xilinx 可编程逻辑器件的高级应用与设计技巧. 北京:电子工业出版社, 2004.

[11] Fry T W, Hauck S A. SPIHT image compression on FPGAs. IEEE Transactions on Circuits and Systems for Video Technology, 2005, 15(9):1138-1147.

[12] Dawood A, Williams J. On-board satellite image compression using reconfigurable FPGAs. 2002 IEEE International Conference on Field-Programmable Technology, 2002:306-310.

[13] Corsonello P, Perri S, Staino G. Low bit rate image compression core for onboard space applications. IEEE Transactions on Circuits and Systems for Video Technology, 2006, 16(1):114-128.

[14] 肖山竹, 陈尚锋, 孙广富. 动态可重构的高速小波图像压缩系统. 微处理机, 2003, 3:34-37.

[15] 王家文, 曹宇. 图形图像处理. 北京:国防工业出版社, 2004.

[16] 赵学军. 基于小波变换的图像压缩. 图像处理学报, 2006, 3:203-205.

[17] 王剑. 基于 MATLAB 的小波变换在图象压缩中的应用. 计算机工程与应用 2003, 1:57-61.

[18] 夏宇闻. Verilog 数字系统设计教程. 北京:北京航空航天大学出版社, 2003.

[19] Mallat S. A theory for multiresolution signal decomposition:the wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989, 11(7):674-693.

[20] Swenldens W. The lifting schema:a construction of second generation wavelets. Applied and Computational Harmonic Analysis, 1996, 3(2):186-200.

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

[22] 沈兰荪,卓力.小波编码与网络视频传输.北京:科学出版社,2005.
[23] 刘文耀.小波图像编码与专用 VLSI 设计.北京:电子工业出版社,2006:74.
[24] 刘军伟,饶妮妮.提升小波变换的 FPGA 设计与实现.微计算机信息.2005,21(10-2):132-134.
[25] 章毓晋.图像工程(上册)-图像处理和分析.北京:清华大学出版社,1993:153-156.
[26] CCSDS.CCSDS 120.0-G-1, Lossless Data Compression.1997.
[27] 彭进业,俞卞章.一种改进的图像自适应零树编码方法.电子学报,2002,30(4):591-593.
[28] Ramaswamy V N,Namuduri K R,Ranganathan N.Performance analysis of wavelets in embedded zerotree-based lossless image coding schemes.IEEE Transactions on Signal Processing.1999,47(3):100-105.
[29] 王诚,薛小刚,钟信潮.Xilinx ISE 使用详解.北京:人民邮电出版社,2005.
[30] 李国锁.基于 FPGA 的图像压缩系统的设计与实现:(硕士学位论文).长沙:国防科学技术大学,2006.
[31] 吴斌.CCSDS 图像压缩和 AES 加密算法研究及其 FPGA 实现:(硕士学位论文).大连:大连理工大学,2007.
[32] 王镇道,陈迪平,文康益.JPEG 图像压缩算法的 IP 核设计.计算机应用,2005,25(5):1076-1077.
[33] 李现勇.Visual C++串口通信技术与工程实践.北京:人民邮电出版社,2002.

攻读硕士学位期间发表学术论文情况

雷震霖，殷福亮. 空间图像 CCSDS 压缩算法研究与 FPGA 实现. 大连理工大学研究生网络学刊，大连：大连理工大学，2007. 学位论文第 2 章和第 4 章部分内容.

致　谢

首先要感谢我的导师殷福亮教授。本论文从选题、论证、研究到最后的完成，自始至终都凝聚着殷老师的心血。殷老师精深的理论功底、渊博的学识、严谨的作风、敏锐的思维和诲人不倦的师者风范将使我受益终生。殷老师不仅在理论学习和实践项目上给予我很多教诲和帮助，而且在生活和做人处世方面更令我受益匪浅。我将铭记导师的谆谆教诲，用它来指导我今后的人生道路，在此，我谨向殷老师表示真挚的谢意。

我还要感谢教研室的陈喆老师，他深厚的理论知识、丰富的实践经验、一丝不苟的治学态度和助人为乐的精神每时每刻都在影响着我，成为我学习的榜样。陈老师不仅在实践项目上给予我许多信任和帮助，而且在平常的生活中给了我不少启迪，使我终生受益。

感谢中科院空间中心的顾晓东老师，他在整个项目中给予了指导和帮助。在与顾老师的交流中，我进一步加深了对项目的理解以及对背景应用的认识。并特别感谢顾老师在中科院空间中心调试时提供的帮助。

论文和学业的完成，离不开所有教授我知识的老师们，特别是在大学本科和硕士研究生阶段，电信学院的老师们：郭成安老师、邱天爽老师、马晓红老师、李建华老师和魏东兴老师等，是他们无私地教授我信号处理、通信等方面的专业知识，使得我逐步进入到这个日新月异的领域，感受到科学的魅力，在此，我对这些曾经教育和帮助过我的老师们表示深深的敬意。

我要感谢同组的杨瑞娜和高超同学。在将近一年的团队学习和生活过程中，他们俩不仅在实际工作中帮助我、信任我，而且在生活中给我带来了无数笑声与欢乐，使得我们这个团队既积极进取，又自然愉快，在这种团队氛围中，我们三个人紧密合作，最终出色地完成了毕业设计任务。同时感谢张雷，张志乐同学在硬件调试过程中的及时援助，使得整个工程可以顺利跑通，得以按时进行演示。

我还要感谢通信与信号处理研究所的全体同学，是他们给了我轻松愉快、积极向上的学习环境。我要特别感谢我的组长吴斌师兄，是他们给了我 DSP 实际项目经验。我还要感谢 ZSP 项目组的师弟师妹，是他们尽心尽力地配合我完成了项目。我还要感谢金乃高师兄，从他那里学到了认真负责的态度。感谢同寝室张锐和杨志峰同学在生活中对我的帮助。感谢 2005 届的所有同学和朋友，我们的友谊地久天长。

最后我要感谢我的父母，他们含辛茹苦地抚养我成人，教给我做人的道理，给我家庭的温暖，默默无闻地支持我，他们是我今生最大的幸福！

空间图像CCSDS压缩算法研究与FPGA实现

作者： 雷震霖
学位授予单位： 大连理工大学

相似文献(10条)

1.学位论文 邓丽娟 图象压缩系统中熵编解码器的FPGA设计及实现 2005

随着移动终端、多媒体、Internet网络、通信，图像扫描技术的发展，以及人们对图象分辨率，质量要求的不断提高，用软件压缩难以达到实时性要求，而且会带来因传输大量原始图象数据带来的带宽要求，因此采用硬件实现图象压缩已成为一种必然趋势。而熵编码单元作为图像变换，量化后的处理环节，是图像压缩中必不可少的部分。研究熵编解码器的硬件实现，具有广阔的应用背景。本文以星载视频图像压缩的硬件实现项目为背景，对熵编码器和解码器的硬件实现进行探讨，给出了并行熵编码和解码器的实现方案。熵编解码器中的难点是huffman编解码器的实现。在设计并行huffman编码方案时通过改善Huffman编码器中变长码流向定长码流转换时的控制逻辑,避免了因数据处理不及时造成数据丢失的可能性，从而保证了编码的正确性。而在实现并行的huffman解码器时，解码算法充分利用了规则化码书带来的码字的单调性，及在特定长度码字集内码字变化的连续性，将并行解码由模式匹配转换为算术运算，提高了存储器的利用率、系统的解码效率和速度。在实现并行huffman编码的基础上，结合针对DC子带的预测编码，针对直流子带的游程编码，能够对图像压缩系统中经过DWT变换，量化，扫描后的数据进行正确的编码。同时，在并行huffman解码基础上的熵解码器也可以解码出正确的数据提供给解码系统的后续反量化模块，进一步处理。在本文介绍的设计方案中，按照自顶向下的设计方法，对星载图像压缩系统中的熵编解码器进行分析，进而进行逻辑功能分割及模块划分，然后分别实现各子模块，并最终完成整个系统。在设计过程中，用高级硬件描述语言verilogHDL进行RTL级描述。利用了Altera公司的QuartusII开发平台进行设计输入、编译、仿真，同时还采用modelsim仿真工具和Symplicity的综合工具，验证了设计的正确性。通过系统波形仿真和下板验证熵编码器最高频率可以达到127M，在62.5M的情况下工作正常。而熵解码器也可正常工作在62.5M，吞吐量可达到2500Mbps，也能满足性能要求。仿真验证的结果表明：设计能够满足性能要求，并具有一定的使用价值。

2.学位论文 王纲毅 基于提升机构的二维离散小波的FPGA设计 2005

在卫星遥感设备中，随着遥感技术的发展和对传输式观测卫星遥感图像质量要求的不断提高，航天遥感图像的分辨率和采样率也越来越高，由此引起高分辨率遥感图像数据存储量和传输数据量的急剧增长，然而卫星信道带宽有限。为了尽量保持高分辨率遥感图像所具有的信息，必须解决输入数据码率和传输信道带宽之间的矛盾。所以星载高分辨率遥感图像数据的高保真、实时、大压缩比压缩技术就成了解决这一矛盾的关键技术。FPGA器件为实现数据压缩提供了一种压缩算法的硬件实现的一个理想的平台。FPGA器件集成度高，体积小，通过用户编程实现专门应用的功能。它允许电路设计者利用基于计算机的开发平台，经过设计输入，仿真，测试和校验，直到达到预期的结果，减少了开发周期。小波变换能够适应现代图像压缩所需要的如多分辨率、多层质量控制等要求，在较大压缩比下，小波图像压缩质量明显好于DCT变换，因此小波变换成为新一代压缩标准JPEG2000的核心算法。同时，小波变换的提升算法结构简单，能够实现快速算法，有利于硬件实现，因此提升小波变换对于采用FPGA或ASIC来实现图像变换来说是很好的选择。本文针对卫星遥感图像的数据流，主要研究可以对卫星图像进行实时二维小波变换的方案。针对提升小波变换的VLSI结构和FPGA设计中的关键技术，从边界延拓、滤波器结构、整数小波、定点运算、原位运算等方面进行了研究和讨论，并且完成了针对卫星遥感图像的分块二维9/7提升小波变换的FPGA实现。采用VerIlog语言对设计进行了仿真验证，并将仿真结果同matlab仿真结果进行了比较，比较结果表明该方案能实现对卫星遥感图像数据流的二维提升小波变换的功能。同时QuartusII综合结果也表明，系统时钟能够工作在很高的频率，可以满足高速实时对卫星图像的小波变换处理。

3.期刊论文 杨隽.周诠.张敏瑞.YANG Jun.ZHOU Quan.ZHANG Minrui BP神经网络图像压缩算法乘累加单元的FPGA设计 -现代电子技术2009,32(19)

提出一种基于三层前馈BP神经网络实现图像压缩算法的方案,该方案采用可重载IP核和VHDL代码相结合的设计方式.对方案中重要单元一束累加单元进行了FPGA设计,该模块设计采用流水线处理方式,增大了数据吞吐量,减小了系统延时,提高了时钟频率,并完成了该单元的行为级功能仿真.仿真结果验证了FPGA设计的可行性.

4.学位论文 吴斌 CCSDS图像压缩和AES加密算法研究及其FPGA实现 2006

遥感图像是深空探测和近地观测所得数据的重要载体，在军事和社会经济生活领域发挥着重要作用。由于遥感图像数据量巨大，它的存储和传输已成为遥感信息应用中的关键问题。图像压缩编码技术能降低图像冗余度，从而减小图像的存储容量和传输带宽，它的研究对于遥感图像应用具有重要的现实意义。CCSDS图像压缩算法是空间数据系统咨询委员会(CCSDS)提出的图像数据压缩算法。该算法复杂度较低，并行性好，适合于硬件实现，能实现对空间数据的实时处理，从而广泛应用于深空探测和近地观测。对于直接关系到军事战略、经济建设等方面的遥感图像的传输，必须对它进行加密处理。AES加密算法是由美国国家标准和技术研究所(NIST)于2000年发布的数据加密标准，它不但能抵抗各种攻击，保证加密数据的安全性，而且易于软件和硬件实现。本论文对CCSDS图像压缩算法和AES加密算法进行了研究，完成的主要工作包括：

(1)研究了CCSDS图像压缩算法的原理和结构，用C语言实现了算法的编解码器，并与SPIHT算法和JPEG2000算法的性能进行了比较。

(2)研究了AES加密算法的原理和结构，用C语言实现了算法的加解密器。

(3)介绍了实现CCSDS图像压缩算法和AES加密算法的FPGA设计所选择的软件开发工具、开发语言和硬件开发平台。

(4)给出了CCSDS编码器的FPGA实现方法和实现性能。

(5)给出了AES加密器的FPGA实现方法和实现性能。

本文设计的CCSDS图像压缩和AES加密FPGA系统运用了流水线设计、高速内存设计、模块并行化设计和模块串行化设计等技术，在系统速度和资源面积上取得了较好的平衡，达到了预期的设计目的。

5.学位论文 申申 基于帧内预测的静态图像编码器FPGA设计 2008

随着计算机技术和Internet的迅速发展，静态图像的压缩技术得到了迅速的发展和广泛的应用。JPEG是现有的应用广泛的静态图像压缩标准，JPEG2000作为新近的静态图像编码标准和JPEG相比有更高的压缩效率，且具有许多新的特点。

采用帧内预测的H.264/AVC帧内编码是H.264/AVC视频编码标准中的重要技术。它在编码图像质量和编码复杂度方面都与JPEG2000编码标准相比更有竞争力，是一种有效的静态图像编码解决方案。但是，和JPEG2000编码器相比，H.264/AVC编码器的缺点在于，在带宽受限的情况下，无法有效的控制编码图像的比特数，以满足传输的需要。

本文首先提出了一种针对H.264/AVC帧内编码的比特数控制策略。其次，设计了一种基于FPGA平台的采用帧内预测编码的静态图像编码硬件系统。通过采用宏块级流水线，使系统处理速度加倍，并且提出了一种宏块流水线重启的方法，以保持系统在比特数控制下工作的性能。系统还采用了可配置的4路并行帧内预测器，并通过亮度、色度预测穿插进行的方式提高了预测器的利用率。系统还采用了一种并行4×4二维DCT/Hadamard集成变换结构。除此之外，本文还设计了CABAC熵编码硬件结构，包括二值化处理器，上下文模型选择器以及全流水的算术编码器。通过对系统各功能模块进行仿真，结果正确，符合设计要求。

6.期刊论文 高珍.邓甲昊.孙骥.宋崧.GAO Zhen.DENG Jia-hao.SUN Ji.SONG Song 微型无人机5/3提升小波改进算法FPGA设计 -微计算机信息2009,25(17)

数字图像无线传输是微型无人机完成特殊侦察任务的重要组成部分.针对微型无人机的特点及其对数字图像压缩的实时性要求,重点提出了一种提取图像重要信息的内嵌延拓5/3提升小波改进算法,且基于该改进算法设计了一种二维数字图像小波变换的硬件结构并进行仿真实验分析.结果表明,该结构算法简单、占硬件资源少、硬件利用率高;与传统小波变换相比有效地缩短了运算时间、提高了变换速度,基本满足微型无人机对数字图像传输的实时性要求.

7.学位论文 卞俊锋 JPEG2000标准中算术编码的FPGA设计与码率控制算法的研究 2005

JPEG2000是由ISO/ITU-T组织下的IECJTC1/SC29/WG1小组制定的下一代静止图像压缩标准，其优良的压缩特性使得它将具有广泛的应用领域。JPEG2000算法非常复杂，图像编码过程占用了大量的处理器时间开销和内存开销，因而通过对JPEG2000算法进行优化并采用硬件电路来实现JPEG2000标准的部分或全部内容，对加快编码速度从而扩展其应用领域有重要的意义。

本文的研究主要包括两方面的内容，其一是JPEG2000算术编码器算法的研究与硬件设计，其二是JPEG2000码率控制算法的研究与优化算法的设计。在研究算术编码器过程中，首先研究了JPEG2000中基于上下文的MQ算术编码器的编码原理和编码流程，之后采用有限状态机和二级流水线技术，并在不影响关键路径的情况下通过对算术编码步骤优化采用硬件描述语言对算术编码器进行了设计，并通过了功能仿真与综合。实验证明该设计不但编码速度快，而且流水线短，硬件设计的复杂度低且易于控制。

在研究码率控制算法过程中，首先结合率失真理论建立了算法的数学模型，并验证了该算法的有效性，之后深入分析了该数学模型的实现流程，找出影响算法效率的关键路径。在对算法优化时采用黄金分割点算法代替原来的二分查找法，并使用了码块R-D斜率最值记忆和码率误差控制算法。实验证明，采用优化算法在增加少量系统资源的情况下使得计算效率提高了60％以上。之后，分析了率失真理论与JPEG2000中PCRD-opt算法的具体实现，又提出了一种失真更低的比特分配方案，即按照“失真/码长”值从大到小通道编码顺序进行编码，通过对该算法的仿真验证，得出在固定码率条件下新算法将产生更少的失真。

8.期刊论文 徐晓东.周以齐.郝群.闫龙.XU Xiao-dong.ZHOU Yi-qi.HAO Qun.YAN Long 二维9/7整数离散小波变换的FPGA设计 -光学技术2008,34(5)

采用流水线设计技术和模块化设计思想,提出一种基于CCSDS新推荐标准的图像数据 压缩算法中二维9/7整数离散小波变换的实时并行实现结构.结构主要由行变换、列变换和 行缓存等模块组成,行、列变换可并行进行.使用FPGA内嵌的BlockRAM作为行缓存器,减少 了外部存储器的使用、访问以及时间延迟.结果表明,该实现结构能够减少运算量和电路规模,获得较高的吞吐率,增加硬件资源利用率,提高变换速度.

9.期刊论文 陈维波.邓家先.王海荣.Chen Weibo.Deng Jiaxian.Wang Hairong 基于FPGA的快速9/7整形离散小波变换系统设计 -电子科技2010,23(5)

CCSDS图像数据压缩标准中采用9/7整形离散小波变换为核心算法,该算法结构简单,易于硬件设计实现.文中基于FPGA设计实现了9/7整数离散小波变换系统,设计中使用内部RAM存储方式,减小了对存储器的需求量,同时采用基于行的列变换方式,行、列变换同时进行,提高了运行速度,仿真和综合结果显示该设计需要的硬件资源少,运行速度快.

10.学位论文 徐长远 基于FPGA的静止图像压缩系统的研究——JPEG编码器的设计 2007

基于FPGA的静止图像压缩系统的研究-JPEG编码器的设计电力电子与电力传动数字图像在人们生活中的应用越来越广泛，由于原始图像数据量比较大，因此数字图像压缩技术逐渐成为图像应用的一个核心环节。在数字图像压缩领域，国际标准化组织于1992年推出的JPEG标准应用最为广泛。

本文基于FPGA设计了JPEG图像压缩系统，通过改进算法，优化结构，在合理的利用硬件资源的条件下，有效的挖掘出算法内部的并行性。改进了DCT变换算法，设计了并行查找表结构的乘法器，采用了流水线优化算法来解决时间并行性问题，提高了DCT模块的运算速度。依据Huffman编码表的规律性，采用并行查找表结构，用较少的存储单元完成了Huffman编码运算，同时提高了编码速度。整个设计通过EDA软件进行了逻辑综合及功能与时序仿真。综合和仿真结果表明，本文提出的算法在速度和资源利用方面均达到了较好的状态，可满足实时JPEG图像压缩的要求。

设计了一个硬件开发平台，对JPEG图像压缩系统进行了验证。硬件平台上使用ADV7181B来实现AD转换；使用TI公司TMS320C6416型DSP芯片实现了系统配置以及通过PCI接口与上位机PC的实现数据交换；使用Microsoft VC++6. 0开发平台开发了系统控制软件平台，实现对整个压缩系统的控制。