南京理工大学

硕士学位论文

基于FPGA的JPEG解码算法的研究与实现

姓名：张艳

申请学位级别：硕士

专业：通信与信息系统

指导教师：钱玲

20090624

摘 要

在多媒体通信技术中，JPEG以其对静止图像的优良的压缩特性获得了广泛应用，成为国际通用的标准。本文在简要介绍JPEG标准和FPGA(Field Programmable Gate Array,现场可编程门阵列）设计流程的基础上，从总体规划的角度提出了整个系统结构的设计思想，对JPEG解码器各部分算法进行了深入的研究，接着对各个模块的设计进行了详细的描述。采用了Verilog硬件描述语言对JPEG基本模式硬件解码器的各主要模块进行设计实现，并给出了功能仿真波形图及测试结果。在JPEG图像解码器中，二维IDCT（离散余弦逆变换）单元在整个解码过程中消耗的时间占了很大的比例，因此提高二维IDCT变换的速率显得很有必要，本文利用行列分解的方法实现二维IDCT变换，有效地减少了二维IDCT单元的运算时间。将反Zig-Zag扫描集成到了反量化器上，节约了反量化和反Zig-Zag扫描的时间。基于JPEG标准中Huffman码表的规律提出并行熵解码的算法，实现了快速Huffman解码。本课题的JPEG解码器的设计与实现，为复杂的图像解码器在FPGA上实现做了探索性的尝试，对其他的图像解码系统的IP核设计以及FPGA实现有着积极的借鉴意义。

关键词：JPEG解码，FPGA，IDCT变换，Verilog硬件描述语言

Abstract

JPEG has been widely used and become one of the international standards for its excellent performance on compressing still image among multimedia communication technologies. On the basis of briefly introducing JPEG standard and FPGA design process, the design principle of the overall decoding structure is proposed. Then, the thesis gives design details of all the major modules of the JPEG decoder which is implemented using Verilog HDL also with the simulation waveforms and the implementing results after the JPEG decoding algorithm was studied deeply. In the JPEG decoder design, it is necessary to increase the speed of two-dimensional IDCT （Inverse Discrete Cosine Transform） module because it consumes a significant portion of the total time in the decoding process. This thesis uses the improved means of ranks decomposition to implement the 2D-IDCT module, which effectively reduces computing time. The inverse Zig-Zag scanning module is integrated into the inverse quantizer, which saves the time of anti-quantization and inverse Zig-Zag scanning. According to the characteristics of Huffman table and JPEG standard, parallel algorithm for entropy decoding is proposed to achieve a fast Huffman decoding.

The design and realization action of the JPEG decoder provides an exploring attempt to implement complex image encoders based on FPGA and a positive reference to the systems IP core design and their FPGA realization. of the other image decoding.

Keywords: JPEG decoding, FPGA, IDCT, Verilog HDL

声　明

本学位论文是我在导师的指导下取得的研究成果，尽我所知，在本学位论文中，除了加以标注和致谢的部分外，不包含其他人已经发表或公布过的研究成果，也不包含我为获得任何教育机构的学位或学历而使用过的材料。与我一同工作的同事对本学位论文做出的贡献均已在论文中作了明确的说明。

研究生签名：_____　　2009年06月24日

学位论文使用授权声明

南京理工大学有权保存本学位论文的电子和纸质文档，可以借阅或上网公布本学位论文的部分或全部内容，可以向有关部门或机构送交并授权其保存、借阅或上网公布本学位论文的部分或全部内容。对于保密论文，按保密的有关规定和程序处理。

研究生签名：_____　　2009年06月24日

1 绪论

1.1 研究背景

1.1.1 图像压缩编码技术的发展和趋势

从20世纪60年代起，随着电子技术和计算机技术的不断提高和普及，应用数字方法进行图像处理（数字图像处理）进入了高速发展时期。图像的数字化处理表示使得图像信号可以高质量地传输，并便于图像的检索、分析、处理和存储。但是，数字图像的表示需要大量的数据空间，因而必须进行数据压缩编码。

图像压缩编码从20世纪40年代末开始进行系统研究以来，迄今已有60多年的历史。早在1948年，Oliver即提出电视信号的线性PCM编码理论。在50年代初发表的电视编码的早期工作中，已提出线性预测理论，测量了亮度信号和差值信号的概率分布[14]。

1958年，Graham首次采用计算机模拟实验的方法，研究静止图像的前值预测DPCM编码法，获得3~4比特/像素图像质量优良的结果。1966年，O'Neal继续图像压缩编码的计算机模拟实验，为DPCM预测编码法作了初步的理论工作。1969年举行了首届《图像编码会议》，并随后出版了名为《图像频带压缩》的会议论文集，至今这个会议仍在举行。

进入20世纪70年代后，有关图像压缩编码的文献日益增多。其中有纯理论的，大量的是计算机模拟实验研究，也有各种类型的硬件系统。图像压缩编码硬件系统的研制随着数字通信系统的迅猛发展而相应地发展，进入20世纪80年代后，图像压缩编码研究与开发又有突出的进展。二进制图像传真机已成为办公室的重要通信设备。地球资源卫星所获得的遥感图像，已经大量地应用到国民经济的各个方面。而在彩色广播电视、会议电视和电视电话的编码方面，已经研制出几十种帧内编码和帧间编码的硬件系统。图像压缩编码的各类应用已出现稳定发展的趋势。与此同时，各类学术会议上有关图像压缩编码的文献报告连接不断。各国（美、西德、日、英、法、中）先后编辑了论文专辑。

经过30多年的基础研究，图像压缩编码研究从20世纪80年代末90年代初进入新的时期。一方面进入到实用化的研究，另一方面继续深入理论研究。

实用化方面的研究集中在国际标准化组织ISO和国际电信联盟ITU联合制订的几个标准上，有力地推动了实用开发工作。于1988年形成草案，1990年通过的ITU-T H.261建议，是图像压缩编码技术走向实用化的重要一步，它是图像压缩编码40年研究成果的结品。进入90年代后相继提出了一系列图像压

缩编码的标准。这些国际建议普遍采用的混合编码技术是当今最实用的高效编码方法，得到了广泛的推广与应用。图像压缩技术的发展趋势是：算法更复杂，压缩率更高，JPEG 的压缩率在 1：20 左右，JPEG 2000 的压缩率将为 1：200 或更高，MPEG 压缩标准也已经过几代发展，从 MPEG-1 到 MPEG-2，到现在的 MPEG-4，压缩算法越来越复杂，运算量越来越大，压缩率也越来越高。

在理论方面，1989 年 Mallat、1990 年 Daubechies 提出多分辨率小波变换方法，该方法很快被应用在图像编码的研究上，原来进行的子带编码、金字塔编码、综合编码都因为有了小波变换才得以实现，因此小波编码方法得到广泛和深入的研究。2000 年 ISO/IEC 新的静态图像压缩编码标准 JPEG 2000 就应用小波编码获得比原 JPEG 更好的质量。另外，1989 年前后提出的面向对象的分析综合，模型基编码和分形编码，目前还在研究之中。

展望未来，随着数字集成电路、计算机科学以及网络通信的进一步发展，国民经济和社会生活方面不断增长的需要，图像压缩编码技术必将会有更多的发展。

1.1.2 图像压缩编码的国际标准

图像压缩编码方法很多，发展也很循序，根据不同应用目的而制定的图像压缩编码的国际标准相继推出。另外，数学、工程技术以及计算机本身体系结构软硬件性能的发展和提高，使得图像压缩编码的理论和技术得到了空前的发展和应用。对压缩编码方法归纳总结使其成果系统化，有助于我们了解其发展方向，也使尚未解决的问题明确化。下面对这些标准作一些简单介绍[13]：

（1）JPEG 标准，JPEG 静止图像压缩编码标准，是由国际标维化组织（ISO）于 1992 年制定的。JPEG 标准是帧内编码，在保证图像质量的前提下，每个象素可以由 24bit 减少到 1bit，压缩比 20～30。JPEG 在实际系统中有广泛的应用前景，如卫星图像、图像文献资料以及新闻图像等的保存和传输。JPEG 的应用面很宽，对所需编码图像的性质，如大小、黑白/彩色、编码方式等均不预先设定。

（2）MPEG 标准，MPEG 动态图像压缩编码标准，是由国际标准化组织（ISO）于 1992 年制定的。MPEG 不仅用上一帧的图像预测当前图像，而且也使用下一帧图像预测当前图像，即双向预测，这是与 H.261 标准有重要区别的地方。MPEG-1 速率为 1.5Mb/s，MPEG-2 速率在 3~10Mb/s 之间，MPEG-4 更适宜在超低速率如电视、电话等方面的应用，MPEG-3 已停用。

（3）H.261/H.263 建议，支持通信业务视听视频编解码标准，是国际电话电报咨询委员会（CCITT）提出的，H.261 于 1990 年得到批准，该建议（标准）是用于会议电视电话的国际标准，既采用了帧内编码，又采用了帧间编码，因

此它的压缩比大致是JPEG标准的三倍。随着多媒体技术应用的发展，H.261建议不断改进，不断完善，相继推出了H.262建议、H.263建议，既适用于低速通信网，也适用于高速通信网。

（4）JPEG 2000标准，新一代静止图像压缩编码标准，同样是由JPEG组织负责制定的。JPEG 2000与传统JPEG最大的不同，在于它放弃了JPEG所采用的以离散余弦变换为主的区块编码方式，而采用以小波变换为主的多解析编码方式。此外，JPEG 2000还将彩色静态画面采用的JPEG编码方式与二值图像采用的JBIG编码方式统一起来成为对应各种图像的通用编码方式。在编码端以最大的压缩质量和最大的图像分辨率压缩图像，在解码端可以从码流中以任意的图像质量和分辨率解压图像，最大可达到编码时的图像质量和分辨率。JPEG 2000应用的领域包括互联网、彩色传真、打印、扫描、数字摄像、遥感、移动通信、医疗图像和电子商务等。

1.1.3 图像压缩处理实现途径

从实时性的的角度来看，为了提高整个图像压缩处理系统的运行速度，通常可以有两类方法：其一，对图像压缩编解码算法的改进和简化，但事实上，很多算法理论已经相当成熟，降低算法实现复杂度的余地并不大；其二，对实现图像压缩处理手段的选择，根据算法的特性及对速度和质量等的需求，有针对性地选择适合的实现手段。目前图像处理的实现主要分以下三种途径进行：

（1）PC机上的软件实现，利用PC机运行速度高的特点，对图像进行压缩处理，实现起来灵活性强，易于管理和维护，也便于计算机进行集中处理。同时，我们也发现，由于图像处理的计算量非常大，计算机本身的结构特性使得单指令单数据的处理过程往往无法满足实时性的要求，因此，该方法更适用于算法的验证。

（2）DSP（数字信号处理器）的实现，由于其内部采用专门的硬件结构来实现某些数字信号处理的常用算法，速度得到了大大的提高。DSP在近些年发展迅速，其应用也涉及图像处理领域。但其体系依然是串行指令执行系统，并且对某些算法的硬件优化并不能满足众多算法的需求，所以DSP的应用仍然存在着一定的局限性。

（3）专用集成电路

专用集成电路是专为某项应用或某个算法而设计的硬件芯片，是近年来世界范围内的研究热点，速度较前两种实现方式快很多、性能也较稳定，所以在一些尖端的科技应用领域中具有非常广泛的应用。但是它的价格很高，设计周期长，灵活性不够。

随着微电子技术和半导体工业的不断创新和发展，FPGA作为可编程的专

用集成电路也得到了快速发展，以硬件描述语言（HDL）开发的FPGA数字系统在越来越多的领域得到应用。在图像处理领域，鉴于FPGA系统具有高速、高可靠性、设计开发周期短、在线可重构性，低功耗，低成本等许多优点，适合于开发各种适应用户要求的、灵活性高的图像压缩处理及传输设备。

1.2 课题研究目标

随着多媒体和网络通信等领域的飞速发展，JPEG 压缩系统的应用也变得越来越广泛，同时也对压缩系统提出了更高的要求，尤其是在一些对实时性能要求比较高的应用场合，它们不仅要求对图像有高的压缩比和图像品质，更加要求有高的实时性，这样对 JPEG 标准的解码提出了很高的要求。目前国外已有不少关于提高解码速度的相关文章，国内研究相对少。

本课题的目标是以FPGA为硬件平台，设计并实现符合JPEG标准基本模式（Baseline Mode）的JPEG图像压缩解码器。该解码器的应用目标主要为各种消费类电子，因此要求该芯片具有价格低廉，处理速度快，扩展性好等特性。基于以上要求，JPEG基本模式解码器采用了高级硬件描述语言Verilog描述，按照Top-Down（自顶向下）和模块化方法设计为可定制的逻辑组件，最后通过FPGA厂商Xilinx提供的软件ISE 9.2完成综合、布局布线以及仿真。

1.3 课题开发环境

本课题基于FPGA领导厂商Xilinx公司的FPGA开发工具ISE进行设计开发，结合ISE自带的综合、仿真工具对主要的模块进行验证。

1.4 课题研究意义

探索利用FPGA对静止图像采用JPEG解压缩方法，使用Verilog HDL语言实现主要模块的设计，对其他图像或视频压缩解码器的FPGA实现以及IP核设计提供一定的参考和帮助。

1.5 论文研究内容及论文结构

本文共分六章，简要介绍各章的主要内容：

第一章绪论。简单说明课题的研究背景、研究目标以及课题所用到的开发环境和课题研究的意义。

第二章 JPEG 图像压缩标准的介绍，包括 JPEG 标准的运行方式和基本构成，有损压缩和无损压缩，JPEG 编码过程以及 JPEG 文件格式。

第三章FPGA技术及其设计方法介绍，简单介绍基于FPGA的设计流程和

方法、Verilog硬件描述语言以及设计工具和仿真工具。

第四章 JPEG 解码器的设计与实现。详细介绍了 JPEG 解码模块的硬件设计过程，分别介绍了读入码流模块、熵解码模块、反量化与反 ZigZag 排序模块、逆离散余弦变换（IDCT）模块的设计与实现。

第五章 JPEG 解码器主要模块的仿真与验证。给出 JPEG 解码主要模块的仿真波形和验证结果，并对结果进行分析。

第六章 总结与展望。对本文的研究工作进行了总结，提出了还需要改进的地方，对该系统的技术实现和算法优化进行了展望。

2 JPEG 图像压缩标准

2.1 JPEG标准概述

2.1.1 JPEG 运行方式和基本构成[15]

JPEG 是"Joint Photographic Expert Group，联合图像专家组"的缩写，是以实现图像数据库、色彩传真、印刷等方面的彩色静止图像编码的标准方式为目标，由 ISO（International Organization for Standardization）和 CCITT（Consultative Committee for International Telegraph and Telephone）的两个组织联合设立的讨论小组。标准方式的正式名称为"Digital compression and coding of continuous-tone still images"，如这个名称表示的一样是具有连续黑白或彩色协调的静止图像的编码技术。两个值（白或黑）的静止图像编码的标准是 JBIG（Joint Bi-level Image coding experts Group），它使用在其他用途上。

作为静态图像压缩的标准算法，JPEG 算法必须满足以下要求：算法独立于图像的分辨率；具有低于 lbit/像素的编码率，并且能够在五秒钟内建立图像，以满足实时要求；在压缩比大约是 2 的情况下能够无失真地恢复原图像；支持顺序编解码和渐进编解码，以及对各种图像成分及数据精度的自适应能力；最后，要求编解码设备简单、易实现。

JPEG 是假想为适用范围非常广泛，通用性很强的技术，所以把算法的功能分为四种运行方式（modes of operation），用户只要从中选择需要的功能即可。这四种运行方式分别是：

（1）基于 DCT 顺序：由 8×8 像素组成的像素块，从左到右进行编码处理并按照从上到下顺序进行扫描。编码处理是由二元 DCT 系数的量化和量化系数的熵编码组成的。

（2）基于 DCT 的扩展：处理的顺序及编码处理的基本构成是与基本 DCT 顺序相同的，但存在多次处理扫描。扫描的顺序与前面相似，负责块内的重要信息（即在第一次扫描中得到粗略的图像）。

（3）无失真：不使用 DCT 变换，对接近像素间的差别进行熵编码，从而不产生失真。

（4）分层（hierarchical）：组合上面三种方式，做成具有多种空间分辨率图像的金字塔结构。

并且，如果这些按照具体实现编码算法的基本结构进行分类的话，可分成 DCT 方式和 Spatial 方式：

（1）DCT 方式：存在于基线（baseline）系统（必备功能）和扩展系统（选

择功能）中。基线系统是所有 JPEG 编码器和解码器中必备的最基本的功能，扩展系统中因为要对应更广泛的应用范围，所以是可选功能。DCT 方式是基于编码和解码，并且是失真的非可逆的过程。

（2）Spatial 方式：对于基线系统和扩展系统被称为独立（independent）功能，编码中使用 DPCM（Differential PCM）和熵编码，与 DCT 方式不具有算法的连续性。该功能是基于编码和解码的无失真、可逆的过程。

2.1.2 失真压缩和无失真压缩

JPEG 标准定义了两种压缩方式：失真压缩和无失真压缩。基于 DCT 变换的压缩方式是有损的，这类方式在高压缩比下，重构的图像也能得到比较好的视觉效果。无失真压缩并不是基于 DCT 变换的，这类压缩方式应用于对图像质量要求很高的场合，解码过程也和 DCT 不相关。

DCT 方式如图 2.1，图 2.2 所示，是在进行了二维 DCT 变换系数量化后熵编码的结构。这种结构的熵编码部分在基线系统中可以使用哈夫曼编码，在扩展系统中可以使用哈夫曼编码以外的其他算术编码。算术编码和哈夫曼编码相比，压缩比率要稍微高些，但是相应的处理也更复杂。DCT 系数量化中所必需的量化表及熵编码中所必需的熵编码表的具体值，根据要进行编码的图像不同而不同，所以没有包含在 JPEG 标准中。但是因为在编码器和解码器中要使用共同的数据，所以通过比特流（bit stream）从编码器传送到解码器。并且随着 DCT 计算中乘法的出现，即使没有进行量化，在进行 DCT 变换和逆变换时，四舍五入产生误差是不同的，由于计算方法等的不同，在解码图像时产生了失真，所以 DCT 方式是不可逆的过程。但是，因为发生的失真对于人类视觉系统来说是察觉不到的，所以即使进行高压缩比压缩，也能得到质量较高的图像。

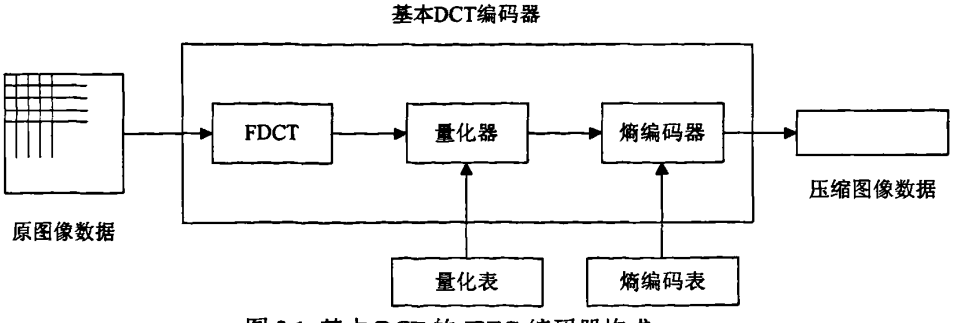

图 2.1 基本 DCT 的 JPEG 编码器构成

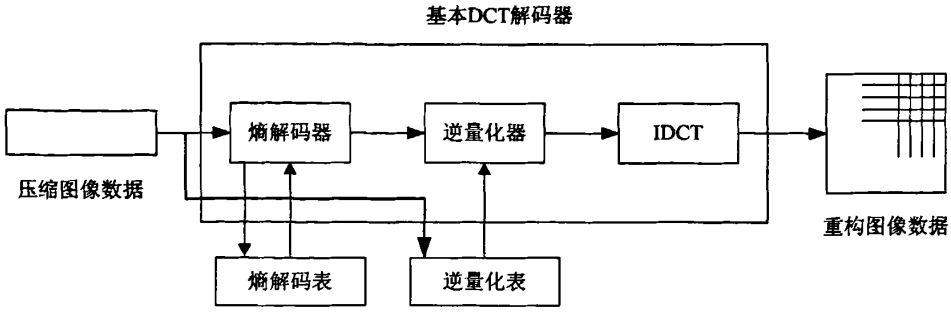

图 2.2 基本 DCT 的 JPEG 解码器的构成

Spatial 方式如图 2.3，图 2.4 所示，是由在二维空间上计算像素之间差值的 DPCM 部分和熵编码部分构成的。

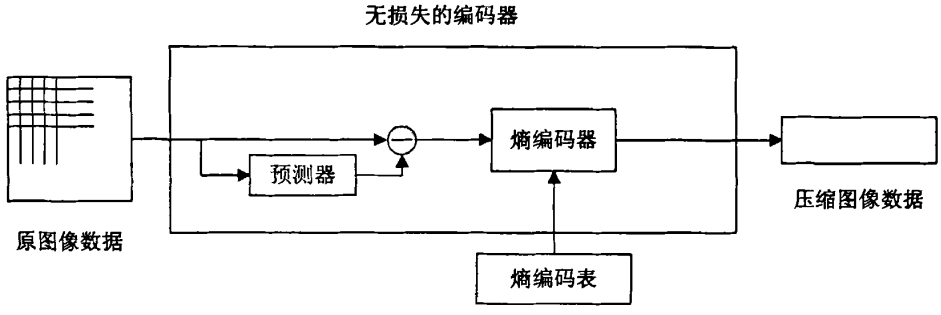

图 2.3 Spatial 方式的 JPEG 编码器的构成

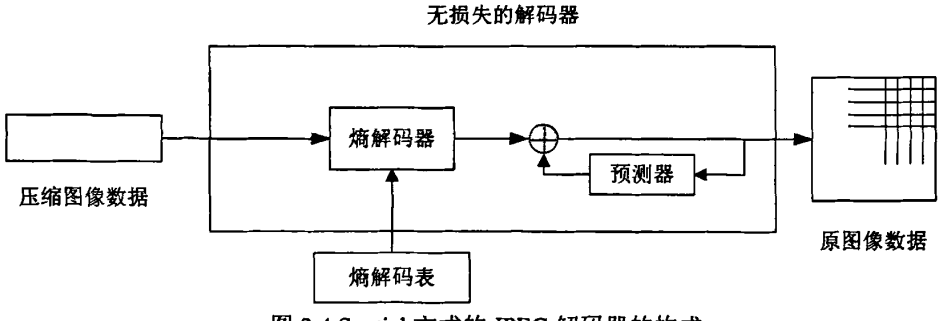

图 2.4 Spatial 方式的 JPEG 解码器的构成

用上面的两种方式，生成压缩率是 2 的指数倍的图像，就可以实现改进方式的分层结构。在分层过程的各层编码中可以使用 DCT 方式或 Spatial 方式，表 2.1 是对上面讲述的 JPEG 编码算法的分类总结。

表 2.1 JPEG 编码算法分类

	DCT 方式		Spatial 方式
	基线系统	扩展系统	
编码器基本构成	DCT+量化+熵编码		DPCM+熵编码
输入图像精度	8 比特	8 比特或 12 比特	2～16 比特
运行方式	顺序	顺序或扩展	顺序
熵编码	哈夫曼编码（2AC，2DC 表）	哈夫曼编码（4AC，4DC 表）或算术编码	哈夫曼编码（4AC，4DC 表）或算术编码

一般情况下，基于 DCT 的基本系统编码算法满足大多数的应用场合，本文讨论 DCT 方式的 JPEG 解码算法。

2.2 JPEG编码过程

2.2.1 色彩空间转换

自然界中所有的颜色都可以用红（Red）、绿（Green）和蓝（Blue）三种颜色波长的不同强度组合而得，这就是人们常说的三基色原理。RGB 是比较常见的彩色图像记录格式，RGB 颜色空间的优点在于直接诉求于人颜色色彩的感应，表述直观，容易理解。但是研究表明，人眼对色度信号的敏感程度远远小于对亮度信号的敏感程度，这时我们考虑将 RGB 色彩空间转变到亮度/色度空间，如 YUV、YCrCb，可以对色度信号进行二次取样，去掉一些不必要的色度信号量从而提高数据压缩率。

色彩空间转换这部分内容本来不属于 JPEG 编码标准中，但在 JPEG 标准中规定，信源图像可以为彩色也可以为黑白，如果是彩色图像，应由亮度分量和色度分量构成，所以在处理之前，应先进行色彩空间的转换，即将 RGB 色彩空间转换到 YUV 或 YCrCb 色彩空间。YUV 用于彩色电视制式，与 YUV 颜色空间类似的一种称为 YCrCb 色彩空间，主要适用于计算机用的显示器。本文选用更适合图形压缩的 YCrCb 颜色模型。YCrCb 与 RGB 信号的转换关系如下：

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.500 & -0.4187 & -0.0813 \\ -0.1687 & -0.3313 & 0.500 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{2.1}$$

因为 JPEG 里的数据都是无符号 8 比特数据，因此色度信号在这里都被加上 128。在通过对色度分量进行二次采样来减少色度分量后，肉眼将察觉不到图像质量的变化。

JPEG 定义了最小编码单位（Minimum Coded Unit），简称 MCU，其值不能大于 10。通常我们对图像数据的取样有三种方式：YCbCr 4：4：4、YCbCr 4：2：2 和 YCbCr 4：1：1。它们所代表的含义是指 Y、Cb 和 Cr 三个成分的数据取样比例。举例而言，如果 Y 取四个数据单元，即水平取样因子（Horizonal Sampling Factor）Hy 乘以垂直取样因子（Vertical sampling Factor）Vy 的值等于 4（Hy·Vy =4），而 Cb 和 Cr 各取一个数据单元，即 HCb.·HCr=1，这样的部分取样称为 YCbCr 4：1：1，由于部分取样的关系，被压缩的数据量就大大减少。但这并不意味着部分取样比越高压缩比越大，因为图像质量将大大降低。

$$\text{JPEG 定义的 } MCU = \sum_{i=1}^{Ns} Hi \cdot Vi \le 10 \text{。} \qquad (2.2)$$

其中的 Ns 为图像成分的个数，Hi 为分量 i 的水平抽样因子，Vi 为分量 i 的垂直抽样因子。

若图像仅有一个分量，则数据以非交错的方式处理，MCU 相当于一个单元；若图像有多个分量的话，则数据就以交错的方式来处理，而 MCU 就是由数个不同的分量中的数据单元组成，其排列方式与 Hi 和 Vi 有着密切的关系。

2.2.2 离散余弦变换（DCT）

离散余弦变换（DCT，Discrete Cosine Transform）是一种经典谱分析方法。它属于离散傅立叶变换的一种特殊情况，即在变换后的傅立叶级数中只包括余弦项。由不同的正交基向量，可以得到不同的正交变换，各种正交变换都能在不同的程度上减少随机向量的相关性，而且信号经过大多数的正交变换后，能量会相对集中在少数变换系数上。当信号的统计特性符合一阶平稳马尔柯夫过程，而且相关系数接近 1 时（许多图像信号都可以足够精确地用此模型描述），DCT 十分接近信号的最佳变换，变换后的系数分布比较集中。即使信号的统计特性偏离这一模型，DCT 的性能下降也不显著。

自 1974 年提出离散余弦变换概念以来，DCT 已被大量应用于数字信号处理的各个领域，特别是图像压缩领域，已经被证明在是图像和视频压缩编码中非常有效的技术之一。很多图像、视频压缩标准都采用了 DCT 变换，目前的图像压缩标准，如 JPEG、H.26X 系列、MPEG 系列等都采用了基于 DCT 的图像压缩算法。

DCT 将一组空间域的像素点转换成频域中的系数，该变换是可逆的。由于人的视觉系统对高频的数据不敏感，因此若对高频部分略微进行消减，人的眼

睛也不容易辨认出来。通过此变换达到了缩小图像信号频带，减少图像数据容量的目的。

在JPEG的DCT中最初通常是将原始图像数据分成许多8×8的数据单元（data units），然后经过零偏置（zero shift）将每个字节的值从0～255转为-128～127，再做DCT处理。DCT将每个数据单元的值转换为64个系数，第一系数称为直流系数（DC coefficient），而其余63个系数则称为交流系数（AC coefficient）。变换后可见，仅左上角的少量低频系数数值较大，其余系数的数值很小，这样就可能只编码和传输少数系数而不严重影响图像质量了。

以下就是JPEG使用的二维DCT公式：

$$Z(u,v)=\frac{1}{4}C(u)C(v)[\sum_{i=0}^{7}\sum_{j=0}^{7}x(i,j)\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}] \tag{2.3}$$

$$\begin{cases}u,v=0\text{时}, & C(u),C(v)=1/\sqrt{2}\\ u,v\neq 0\text{时}, & C(u),C(v)=1\end{cases}$$

i，j代表图像数据矩阵内某个数值的坐标位置；$x(i,j)$代表图像数据矩阵内的(i,j)位置上的像素值，$i,j,u,v=0,1,2...,7$。

由于DCT的运算量相对较大，几乎占到了整个压缩/解压过程的40%，如果直接进行DCT计算，工作量很大。对一个8×8的图像块做DCT变换，需要1024次乘法和896次加法，这难以满足实时处理的需求。因此，必须采用好的快速算法，才能提高处理速度，增强实时性。

2.2.3 量化（Quantization）

严格说DCT本身并不能进行码率压缩，因为64个样值经过变换后仍然得到64个系数。经DCT变换后，比特数增加了，直流分量的最大值是原来256的64/8倍，即0~2047，交流分量的范围是-1024~1023。量化是经过DCT变换后的频率系数进行量化，将DCT系数按比例缩小，并取其最接近的整数值的处理过程称为量化。量化的作用是在保持一定质量前提下，丢弃图像中对视觉效果影响不大的信息。量化的目的是减小非“0”系数的幅度以及增加“0”值系数的数目。量化是图像质量下降的最主要原因。

JPEG标准中采用线性均匀量化器，量化过程为对64个DCT系数除以量化步长并四舍五入取整，量化步长由量化表决定。量化表元素因DCT系数位置和彩色分量的不同而取不同值。量化表为8×8矩阵，与DCT变换系数一一对应。量化表一般由用户规定（JPEG标准给出了参考值），并作为编码器的一个输入。量化表中元素为1到255之间的任意整数。其值规定了其所对应DCT系数的量化步长。DCT变换系数除以量化表中对应位置的量化步长并合入小数部分后，

多数变为零，从而达到了压缩的目的。

特别是按人眼的生理特征对低频分量和高频分量设置不同的量化表，会使大多数高频分量的系数变为零。在高空间频率段，将出现大量连续的零。由于大多数图像的高频分量较小，相应于图像高频成分的系数经常为零，加上人眼对高频成分的失真不太敏感，所以可用更粗的量化；对低频分量采用较细的量化，而对高频分量采用较粗的量化。即根据不同的要求设置不同的量化等级，从而降低数码率。

对于有损压缩算法，JPEG 算法使用均匀量化器，其量化步距是按照系数所在的位置和每种颜色分量的色调值来确定。因为人眼对亮度信号比对色差信号更敏感，因此使用了两种量化表：如表 2.2，表 2.3。此外，由于人眼对低频分量的图像比对高频分量的图像更敏感，因此图中的左上角的量化步距要比右下角的量化步距小。

表 2.2 亮度量化值表

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

表 2.3 色度量化值表

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

如果不使用这两种表，也可以把自定义的量化表替换它们。这两张表依据心理视觉阈制作，对 8bit 的亮度和色度的图像的处理效果不错。当然也可以使用任意的量化表。量化表是定义在 JPEG 文件的 DQT 标记之后。一般为 Y 值定义一个，为 C 值定义一个。量化表是控制 JPEG 压缩比的关键。这个步骤除掉了一些高频量，损失了很多细节。但事实上人眼对高空间频率远没有低频敏感，所以处理后的视觉损失很小。

2.2.4 Z 字形扫描（Zig-Zag Scan）

量化后的系数要重新排序，目的是为了增加连续的“0”系数的个数，就是 0 的游程长度，方法是按照 Z 字形的样式编排，如图 2.5 所示。后续的游程长度编码（Run-Length Encoding，RLE）指的是由 Z 字构成的数据流中各个字符连续重复出现而形成字符串的长度，即游程长度编码是指一个码可同时显示码的值和前面几个“0”。这样就可以把 Z 字形读出的优点显示出来了。因为 Z 字形读出在大多数情况下出现连续“0”的机会比较多，尤其在最后，如果都是“0”，在读到最后一个数后只要给出块结束（EOB）码，就可以结束输出，因此节省了很多码率。Z 字形扫描后就把一个 8×8 的矩阵变成了一个 1×64 的矢量，频率

较低的系数放在矢量的顶部。

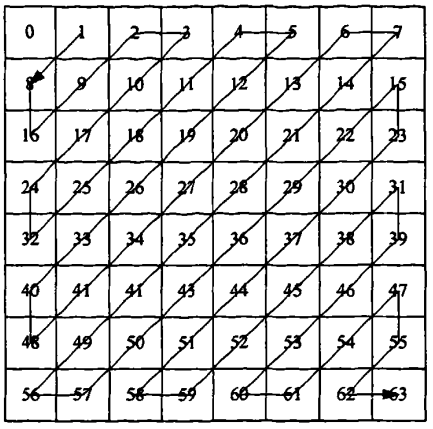

图 2.5 Z 字形扫描图

2.2.5 熵编码（Entropy Encode）

在经过 DCT 编码以及量化之后，已大大降低了图像信息的相关性。熵编码器的作用是进一步降低表达信息所需的数据量。熵编码器的这个作用是通过三种方式来实现的：差分脉冲编码（Differential Pulse Code Modulation，DPCM），游程长度编码和哈夫曼（Huffman）编码。

2.2.5.1 游程长度编码和差分脉冲编码

变换系数经量化后，大多数低频部分的系数非零，而大多数高频部分的系数变为零。相邻 8×8 块之间的直流分量（DC 系数）一般有很强的相关性，即 JPEG 标准对 DC 系数采用差分脉冲编码方法，对相邻像素块之间的 DC 系数的差值进行编码。其余 63 个交流分量（AC 系数）使用游程长度编码，从左上角开始沿对角线方向，以 Z 字形进行扫描直至结束。

（1）直流系数的编码

直流系数可由前面的公式得到：

$$Z(0,0)=\frac{1}{4}C(0,0)[\sum_{i=0}^{7}\sum_{j=0}^{7}x(i,j)\cos(0)\cos(0)] \tag{2.4}$$

其中 $C(0,0)=1/2$，$Z(0,0)$ 就是直流系数，即一块图像样本的平均值，包含了原始8×8图像块里的很多能量。8×8图像块经过DCT变换和量化之后得到的直流系数有两个特点，一是系数的数值比较大；二是相邻图像块的DC系数值变化不大。根据这两个特点，JPEG算法使用了差分脉冲编码技术，对相邻图像块之间量化DC系数的差值（*Diff*）进行编码。

$$Diff=DC(i)-DC(i-1) \tag{2.5}$$

JPEG从 $i=1$ 开始对DC编码，所以 $DC(0)=0$。将当前 *Diff* 值加在上一个

$DC(i-1)$ 值上得到当前值 $DC(i)$。

（2）交流系数的编码

游程长度编码是压缩一个文件最简单的方法之一。它的做法就是把一系列的重复值（例如图像象素的灰度值）用一个单独的值再加上一个计数值来取代。比如有这样一个字母序列aabbbcccccddd它的游程长度编码就是2a3b5c3d。这种方法实现起来很容易，而且对于具有长重复值的串的压缩编码很有效。

在JPEG编码中，使用游程长度编码对交流系数进行编码；量化AC系数的特点是1×64矢量中包含有许多“0”系数，并且许多“0”是连续的，因此，使用非常简单和直观的游程长度编码对它们进行编码。

JPEG使用了2个字节表示AC系数的中间格式符号：

符号1（游程，尺寸）

符号2

其中，1个字节符号1的高4位来表示连续“0”的个数，而使用它的低4位来表示编码下一个非“0”系数所需要的位数，跟在它后面的符号2是量化AC系数的数值。假设有一组矢量（64个的后63个）是：

23，44，0，0，0，0，19，0，-20，-8，0，0，1，0，0，0，0，0，...，0

经过RLE压缩后如下：

（0，23）；（0，44）；（4，19）；（1，-20）；（0，-8）；（2，1）；EOB

EOB是一个结束标记，表示后面都是“0”了。实际上，我们用（0，0）表示EOB。但是，如果这组数字不以“0”结束，那么就不需要EOB。注意（15，0）表示了16个连续的“0”。

2.2.5.2 哈夫曼编码

为了对DPCM编码后的直流（DC）系数和游程长度编码后的交流（AC）系数作进一步的压缩，需要对数据进行哈夫曼编码。

哈夫曼编码是1952年由Huffman提出的编码方法，基本思想是可变长最佳编码定理，即根据源数据符号出现的概率大小进行编码，出现概率大的符号分配越短的码字，出现概率越小的符号分配越长的码字，从而达到用尽量少的比特数表示数据源。

哈夫曼编码的具体步骤如下：

①统计数据源符号出现的概率，得到不同概率的信息符号；

②将数据源符号按概率从大到小排列；

③把两个最小概率相加作为新符号的概率，并按②重排；

④重复①、②，直到概率为1；

⑤在每次合并信源时，将合并的信源分别赋0和1；

⑥寻找从每一信源符号到概率为1处的路径，记录路径上的0和1；

⑦从树根开始写出每一符号的0、1。

哈夫曼编码器可以使用很简单的查表方法进行编码。这种可变长度的哈夫曼表可以事先进行定义。编码时，每个矩阵数据的DC值与63个AC值，将分别使用不同的Huffman编码表，而亮度与色度也需要不同的Huffman编码表，所以一共需要4个编码表，才能顺利地完成JPEG编码工作。

（1）DC编码

JPEG指出连续块的DC系数之间有很紧密的联系，因此可对8×8块的DC值的差别进行编码（Y，Cb，Cr分别有自己的DC）。由于$Diff = DC(i) - DC(i-1)$，所以这一块的$DC(i)$就是：$DC(i) = DC(i-1) + Diff$。$Diff$转换成中间符号VLC（SSSS）和VLI（VVVV），VVVV就等于$Diff$，SSSS表示$Diff$的值所需的位数，这可从DC系数的VLI编码表得到。

DC是采用差分脉冲编码调制的差值编码法，也就是在同一个图像分量中取得每个DC值与前一个DC值的差值来编码。DC采用差分脉冲编码的主要原因是由于在连续色调的图像中，其差值多半比原值小，对差值进行编码所需的位数，会比对原值进行编码所需的位数少许多。例如差值为5，他的二进制表示为101，如果差值为-5，则先改正为正整数5，再将其二进制转换成1的补数即可。所谓1的补数，就是将每位转换；若值为0，便改成1；值为1，则变成0。差值5应保留的位数为3，表2.4即列出差值所应保留的位数与差值内容及VLI编码。

表2.4 DC系数的VLI编码表

SSSS	DIFF值	VLI编码
0	0	
1	-1，1	0，1
2	-3，-2，2，3	00，01，10，11
3	-7...-4，4...7	000...011，100...111
4	-15...-8，8...15	0000...0111，1000...1111
5	-31...-16，16...31	00000...01111，10000...11111
6	-63...-32，32...63	000000...011111，100000...111111
7	-127...-64，64...127	0000000...0111111，1000000...1111111
8	-255...-128，128...255	00000000...01111111，10000000...11111111
9	-511...-256，256...511	000000000...011111111，100000000...111111111
10	-1023...-512，512...1023	0000000000...0111111111，1000000000...1111111111
11	-2047...-1024，1024...2047	00000000000...01111111111，10000000000...11111111111

在差值前端另外加入一些差值的哈夫曼码值，例如亮度差值为5（101）的位

数为3，则哈夫曼码值应该是100，两者连接在一起即为100101。下列两份表格分别是亮度和色度DC差值的编码表。根据这两份表格内容，即可为DC差值加上哈夫曼码值，完成DC的编码工作。

表 2.5 亮度 DC 差值的 Huffman 编码表

SSSS	编码位数	Huffman 编码
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

表 2.6 色度 DC 差值的 Huffman 编码表

SSSS	编码位数	Huffman 编码
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	11	11111111110

JPEG从0开始对DC编码，所以DC（0）=0，然后再将当前 *Diff* 值加在上一个值上得到当前值。

例如Y矩阵中的一个8×8块的DC值为10，二进制为1010，码长为4，查表亮

度DC差值的Huffman编码表的哈夫曼识别码长为3，识别码为101，则其对哈夫曼编码为1011010，负数的编码是取其绝对值的反码进行编码，如DC为-10，则其绝对值为10，二进制仍为1010，反码为0101，哈夫曼编码为1010101。

（2）AC 编码

AC 编码方式与 DC 略有不同，在 AC 编码之前，首先得将 63 个 AC 值按 Z 字形排序，将 AC 系数转换成中间符号，中间符号表示为 VLC（RRRR/SSSS）和 VLI（VVVV），其中 VVVV 等于各个 AC 值，SSSS 表示 AC 值所需的位数，RRRR 是指在非零的 AC 之前，其值为零的 AC 个数，AC 系数的范围与 SSSS 的对应关系与 DC 差值位数与差值内容对照表相似。

如果连续 0 的 AC 个数大于 15 时，则用（15/0）来表示连续 16 个系数全为零，输出 ZRL（Zero Run Length）；而（0/0）称为 EOB（end of block），用来表示其后所剩余的 AC 系数皆等于 0，以中间符号值作为索引值，从相应的 AC 编码表中找出适当的哈夫曼码值，再与 AC 值相连即可。为了提高存储效率，JPEG 里并不直接保存数值，而是将数值按实际值所需要的位数分成 10 组，如表 2.7 所示。

表 2.7 AC 系数的 VLI 编码

SSSS	AC 系数	VLI 编码
1	-1，1	0，1
2	-3，-2，2，3	00，01，10，11
3	-7...-4，4...7	000...011，100...111
4	-15...-8，8...15	0000...0111，1000...1111
5	-31...-16，16...31	00000...01111，10000...11111
6	-63...-32，32...63	000000...011111，100000...111111
7	-127...-64，64...127	0000000...0111111，1000000...1111111
8	-255...-128，128...255	00000000...01111111，10000000...11111111
9	-511...-256，256...511	000000000...011111111，100000000...111111111
10	-1023...-512，512...1023	0000000000...0111111111，1000000000...1111111111

符号 SSSS 用于表达实际值所需要的位数，例如色度信号：

（0，31）；（0，45）；（4，23）；（1，-30）；（0，-8）；（2，1）；（0，0）

只处理每对数右边的那个：

31 实际值所需要的位数为 5，实际保存值为 11111，所以被编码为（5，11111）；

45 同样的操作，编码为（6，101101）；

23 同样的操作，编码为（5，10111）；

-30 同样的操作，编码为（5，00001）；

-8 同样的操作，编码为（4，0111）；

1 同样的操作，编码为（1，1）。

前面的那串数字就变成了：

（0，5），11111，（0，6），101101；（4，5），10111；（1，5），00001；（0，4），0111；（2，1），1；（0，0）

括号里的数值正好合成一个字节 RRRR/SSSS。后面被编码的数字表示范围是-32767…32767。合成的字节里，高 4 位是前续 0 的个数，低 4 位描述了后面数字实际值所需要的位数。例如某一组亮度的中间符号为 5/3，AC 值为 4，首先以 5/3 为索引值，从亮度 AC 的 Huffman 编码表中找到 1111111110011110 哈夫曼码值，于是加上原来 100（4）即是用来取[5，4]的 Huffman 编码 1111111110011110100，[5，4]表示 AC 值为 4 的前面有 5 个零。继续刚才的例子，对每一对 RRRR/SSSS 查找色度 AC 系数的 Huffman 表。

查找色度 AC 系数的 Huffman 表，则：

0/5 Huffman 编码为 11001；

0/6 Huffman 编码为 111000；

4/5 Huffman 编码为 1111111110011001；

1/5 Huffman 编码为 11111110110；

0/4 Huffman 编码为 1011；

2/1 Huffman 编码为 11011；

0/0 EOB Huffman 编码为 00。

那么最后对于前面的例子表示的 63 个系数按位流写入 JPEG 文件中就是这样的：11001 11111 111000 101101 1111111110011001 10111 11111110110 00001 1011 0111 11011 1 00

在 JPEG 文件中，一般有两个 DC 差分 Huffman 编码表，一个是亮度 DC 差分的 Huffman 编码表，一个是色度 DC 差分的 Huffman 编码表；有两个 AC 系数的 Huffman 编码表，一个是亮度 AC 系数 Huffman 编码表，一个是色度 AC 系数 Huffman 编码表。

如果色度 DC 和上一块 DC 的差值 Diff 是-511，就编码成（9，000000000）。查找表色度 DC 差分的 Huffman 编码表，9 的色度 DC 差分 Huffman 编码表是 111111110。那么在 JPEG 文件中，DC 的二进制表示为 11111111 0000000000。

它将放在 63 个 AC 的前面，上个例子的最终位流如下：

111111110 000000000 11001 11111 111000 101101 1111111110011001 10111 11111110110 00001 1011 0111 11011 1 00

实现上述步骤即完成了单幅图像的 JPEG 压缩。

2.3 JPEG文件格式

JPEG委员会在制定JPEG规格时，定出了各种标记符来识别图像压缩数据信息，本课题所研究的是支持JFIF的交换格式的JPEG图像文件。JPEG/JFIF文件格式使用Motorola格式，而不是Intel格式，就是说，如果是一个字的话，高字节在前，低字节在后。

JPEG文件是由一个个段（segments）构成的，每个段长度小于等于65535。每个段从一个标记字开始。标记字都是0xFF打头的，以非0字节和0xFF结束。例如“0xFFDA”，“0xFFDA”，“0xFFC0”。每个标记有它特定意义，这是由第二字节指明的。例如，SOS（Start of Scan=0xFFDA）指明应该开始解码。另一个标记DQT（Define Quantization Table=0xFFDB）就是说它后面有64个字节的量化表。在处理JPEG文件时，如果碰到一个0xFF，而它后面的字节不是0，并且这个字节没有意义，那么遇到的0xFF字节必须被忽略（一些JPEG里，常用0xFF做某些填充用途）。如果在做Huffman编码时碰巧产生了一个0xFF，那么就用0xFF0x00代替。就是说在JPEG图形解码时碰到FF00就把它当作FF处理。另外在Huffman编码区域结束时，碰到几个位没有用的时候，应该用1去填充，然后后面跟FF。

常用的标记有SOI、APP0、DQT、SOF0、DHT、DRI、SOS、EOI。其中SOI等都是标记的名称，在文件中标记是以标记代码形式出现。这里仅列出几个最常用标记的标记代码、表示的意义和占用字节长度[24]。

（1）SOI （Start of Image），图像开始标志符，标记代码为0xFFD8，占2个字节。

（2）APP0 （Application），应用程序保留标记0，标记代码为0xFFE0。

（3）DQT（Define Quantization Table），量化标志段，段内具体信息如图2.6所示。

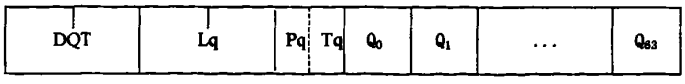

图2.6 量化标志段信息

DQT：量化表段的标示符，2个字节，标记代码为0xFFDB；

L_q：量化表段的段长度，2个字节；

P_q：量化表元素精度，半个字节，值为0表示每个量化表的值为8比特，为1表示16比特；

T_q：量化表编号，半个字节，通常为0~3；

Q_i：量化值，通常为一个字节，根据量化精度不同而不同，量化值按Z字形顺

序排列。

（4）SOF0（Start of Frame），帧图像开始标志段，段内具体信息如图 2.7 所示。

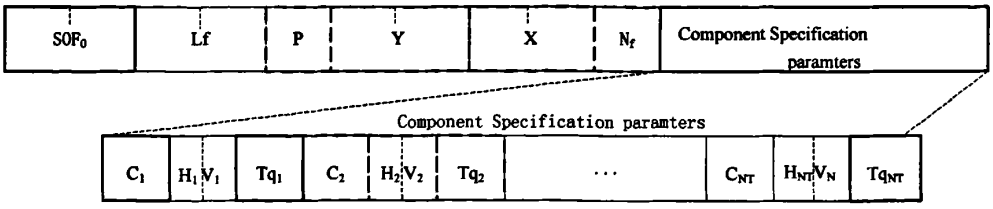

图 2.7 帧图像开始标志段信息

SOF：帧信息段标示符，2 个字节。标记代码从 0xFFC0 到 0xFFCF（0xFFCC 除外），表示不同的编码方式，SOF0 的标记代码为 0xFFC0 表示 Baseline 编码方式；

L_f：帧信息段段长度，2 个字节；

P：采样精度（单位：比特），1 字节；

Y：表示图像中最大的行数，应该等于所有分量中最大垂直采样的行数，2 字节；

X：表示图像中每行最大的点数，应该等于所有分量中最大水平采样的行数，2 字节；

N_f：图像的分量个数，1 字节，通常为 1 或 3，1 表示灰度图，3 表示彩色图；

C_{NT}：图像分量的编号，1 个字节，通常为 0 到 2，不能有重复的；

$H_{NT}V_{NT}$：第 N_f 个分量所使用的水平采样因子和垂直采样因子，1 字节；

Tq_{NT}：第 N_f 个分量所使用的量化表编号，1 字节。

（5）DHT（Define Huffman Table），Huffman 表标志段，段内具体信息如图 2.8 所示。

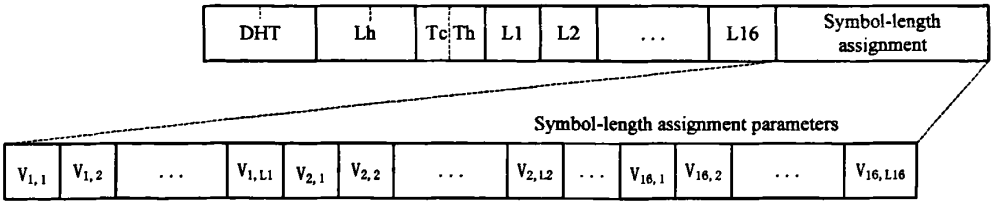

图 2.8 Huffman 表标志段信息

DHT：Huffman 表段标识符，标记代码为 0xFFC4，2 个字节；

L_h：Huffman 表段长度，2 个字节；

T_c：为 0 表示 DC 表或者无损表，为 1 表示 AC 表，半字节；

T_h：Huffman 表编号，半字节；

L_i：表示长度为 i 的 Huffman 编码的个数，1 个字节，i=1...16；

$V_{i,j}$：每个 Huffman 编码所表示的值，1 字节。

（6）SOS（Start of Scan），扫描开始标志段，段内具体信息如图 2.9 所示。

SOS：扫描数据段标识符，2 个字节，标记代码为 0xFFDA；

L_s：扫描数据段段长度，2 个字节；

N_s：扫描数据段中图像分量的个数，1 个字节；

Cs_j：扫描数据图像分量编号，指明图像分量中哪一个将成为扫描数据中第 j 个分量，每个 Csj 应等于一个 SOF 中说明的 Ci，1 个字节；

Td_j：Huffman DC 表编号，半个字节；

Ta_j：Huffman AC 表编号，半个字节；

S_s：1 个字节；

S_e：1 个字节；

$A_h|A_l$：1 个字节，这后三参数与 Baseline 编码无关，通常是 0x00，0x3F，0x00。

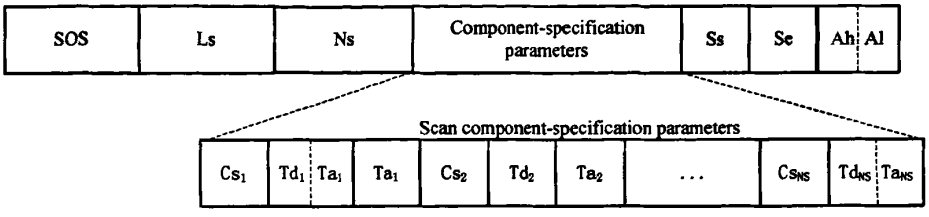

图 2.9 扫描开始标志段信息

（7）EOI （End of Image），图像结束标志符 2 字节，标记代码为 0xFFD9。

3 FPGA 技术及其设计方法

自 1985 年 Xilinx 公司推出第一片现场可编程逻辑门阵列（Field Programmable Gate Array， FPGA）至今，FPGA 已经历了 20 余年的发展历史。在这 20 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：FPGA 从最初的 1200 个可利用门，发展到 20 世纪 90 年代的 25 万个可利用门，进入 2000 年以后，国际上 FPGA 的著名厂商 Altera 公司及 Xilinx 公司又陆续推出了数百万个可利用门的单片 FPGA 芯片，将 FPGA 的集成度提高到了一个新的水平。

纵观 FPGA 的发展历史，之所以具有巨大的市场吸引力，根本在于 FPGA 不仅可以解决电子系统小型化、低功耗、高可靠性等问题，而且其开发周期短、开发软件投入少、芯片价格不断降低，促使 FPGA 越来越多地取代了 ASIC 市场，特别是对于小批量、多品种的产品需求，使 FPGA 成为首选。

目前，FPGA 的主要发展方向是：随着大规模现场可编程逻辑器件的发展，系统设计进入了“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；国际上的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；特别引人注目的是所谓 FPGA 动态可重构技术的的开拓，将推动数字系统设计观念的巨大转变。

本章将对 FPGA 的基本原理，设计方法和流程，开发工具，描述语言及系统方案的设计作一个大体地介绍。

3.1 FPGA简介

3.1.1 FPGA 的基本原理[7]

FPGA 同 CPLD（Complex Programmable Logic Device，复杂可编程逻辑器件）一样是可编程逻辑器件，它是在 PAL、GAL 等逻辑器件的基础之上发展起来的。同以往的 PAL（Programmable Array Logic，可编程阵列逻辑）、GAL（Generic Array Logic，通用阵列逻辑）等相比，FPGA 的规模更大，它可以替代几十甚至几千块通用 IC 芯片。这样的 FPGA 实际上就是一个子系统部件，这种芯片受到全世界范围内电子工程设计人员的广泛关注和普遍欢迎。经过 20 余年的发展，许多公司都开发出了多种可编程逻辑器件，比较典型的是 Xilinx 公司的 FPGA 器件系列和 Altera 公司的 CPLD 器件系列，它们开发较早，占据了较大的 PLD 市场，目前全球范围的 PLD/FPGA 产品有 60%以上都是由 Altera 和 Xilinx 公司提出的，可以说是 Altera 和 Xilinx 共同决定了 PLD(Programmable

Logic Device，可编程逻辑器件）技术的发展方向。当然还有许多其它类型的器件，如 Lattice、Vantis、Actel、Quicklogic、Lucent 等。

简化的 FPGA 基本由 6 部分组成：可编程输入/输出单元、基本可编程逻辑单元、嵌入式块 RAM、丰富的布线资源、底层嵌入功能单元和内嵌专用硬核。

通常 PLD 器件可分为两种结构。一种是基于乘积项（Product-Term）的 PLD 结构，采用这种结构的 PLD 芯片有 Altera 的 MAX7000、MAX3000 系列、Xilinx 的 XC9500 系列和 Lattice、Cypress 的大部分产品。另一种是基于查找表（Look-Up-Table，LUT）的结构，采用这种结构的 PLD 芯片也可称之为 FPGA..。查找表本质上就是一个 RAM。目前 FPGA 中多使用 4 输入的 LUT，所以每一个 LUT 可以看成是一个有 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后，PLD/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表，找出地址对应的内容，然后输出即可。

3.1.2 FPGA 的特点

FPGA 芯片从某种角度来看是一种特殊的 ASIC 芯片，它们除了具有 ASIC 的特点之外，还具有以下优点：

（1）规模越来越大。随着 VLSI（Very Large Scale IC）工艺的不断提高，单一芯片内部可以容纳上百万个晶体管，FPGA 芯片的规模也越来越大，单片逻辑门数已逾百万。芯片的规模越大所能实现的功能就越强，同时也更适于实现片上系统（SOC）。

（2）开发过程投资小。FPGA 芯片在出厂前都做过百分之百的测试，而且 FPGA 设计灵活，发现错误时可直接更改设计，减少了投片风险，节省了许多潜在的花费。所以不但许多复杂系统使用 FPGA 完成，甚至设计 ASIC 也要把实现 FPGA 功能样机作为必要的步骤。

（3）一般可以反复地编程、擦除。在不改变外围电路的情况下，设计不同片内逻辑就能实现不同的电路功能。所以，用 FPGA/CPLD 试制功能样机，能以最快的速度占领市场。甚至在有些领域，因为相关领域协议发展太快，设计 ASIC 可能跟不上技术的更新，只能用 FPGA/CPLD 完成系统的研制与开发。

（4）开发工具智能化、功能强大。现在 FPGA 开发工具种类繁多，智能化高、功能强大。应用各种工具可以完成从输入、综合实现到配置芯片等一系列功能。还有很多工具可以完成对设计的仿真、优化、约束、在线调试等功能。这些工具易学易用，可以使设计人员更能集中精力进行电路设计，快速将产品推向市场。

（5）新型 FPGA 内嵌 CPU 或 DSP 内核，支持软硬件协同设计，可以作为

片上可编程系统的硬件平台。

FPGA 既继承了 ASIC 的大规模、高集成度、高可靠性的优点，又克服了普通 ASIC 设计周期长、投资大、灵活性差的缺点，逐步成为复杂数字硬件电路设计的理想首选。

3.2 FPGA的设计流程[25]

FPGA 设计大体分为设计输入、功能仿真、综合、综合后仿真、实现、布线后仿真与验证和下板调试等主要步骤，如图 3.1 所示：

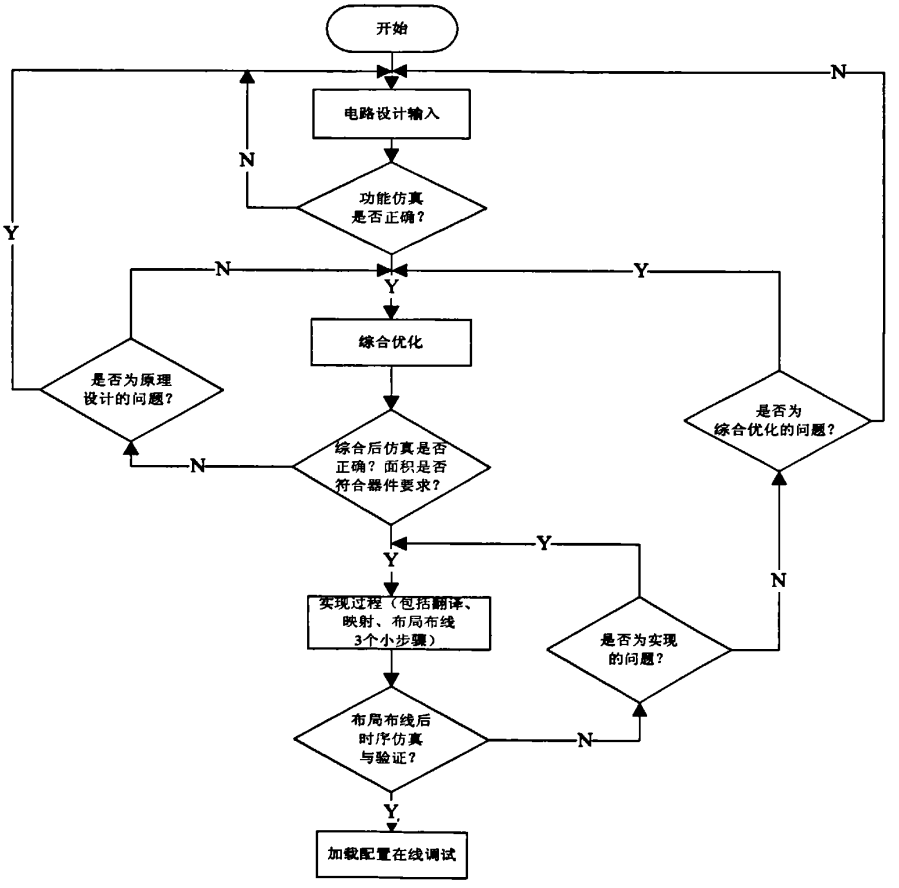

图 3.1 FPGA 的设计流程

（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良好形式，除了 IEEE 标准中的 VHDL 与 Verilog HDL 两种形式外，有些 FPGA 厂家还推出了专用语言，如 Quartus 下的 AHDL。HDL 描述语言在状态机、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工具（如 Synopsys 公司的 FPGA Compiler II 或

FPGA Express）作用下以具体硬件单元较好地实现；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。

（2）功能仿真

电路设计完成后，要用专用的仿真工具对设计进行功能仿真，验证电路功能是否符合设计要求。功能仿真有时也被称为前仿真。常用的仿真工具有Model Tech公司的ModelSim， Synopsys公司的VCS，Cadence公司的NC-Verilog和NC-VHDL，Aldec公司的Active HDL、VHDL/Verilog HDL等。通过仿真能及时发现设计中的错误，加快设计进度，提高设计的可靠性。

（3）综合优化

综合优化（Synthesize）是指将HDL语言、原理图等设计输入翻译成由与、或、非门，RAM，触发器等基本逻辑单元组成的逻辑连接（网表），并根据目标与要求（约束条件）优化所生成的逻辑连接，输出edf和edn等文件，供FPGA厂家的布局布线器进行实现。常用的专业综合优化工具有 Synplicity 公司的Synplify/Synplify Pro、Amplify，Synopsys公司的FPGA Compiler II，Mentor公司旗下的Exemplar Logic公司出品的LeonardoSpectrum等。另外，FPGA厂商的集成开发环境也带有一些综合工具，如Xilinx ISE中的XST等。

（4）综合后仿真

综合完成后需要检查综合结果是否与原设计一致，需要做综合后仿真。在仿真时，把综合生成的延时文件反标到综合仿真模型中去，可估计门延时带来的影响。综合后仿真虽然比功能仿真精确一些，但是只能估计门延时，而不能估计线延时，仿真结果与布线后的实际情况还有一定的差距，并不十分准确。这种仿真的主要目的在于检查综合器的综合结果是否与设计输入一致。

（5）实现

综合结果的本质是一些由与、或、非门、触发器，RAM等基本逻辑单元组成的逻辑网表，它与芯片实际的配置情况还有较大差距。此时应该使用 FPGA厂商提供的工具软件，根据所选芯片的型号，将综合输出的逻辑网表适配到具体FPGA器件上，这个过程就叫做实现（Implementation）过程。Xilinx的实现过程分为翻译（Translate）、映射（Map）和布局布线（Place & Route）等3个步骤。因为只有器件开发商最了解器件的内部结构，所以实现步骤必须选用器件开发商提供的工具软件。

（6）时序仿真与验证

布局布线之后首先应该做时序仿真。时序仿真中应该将布局布线的时延文

件反标到设计中，使仿真既包含门延时，又包含线延时信息。与前面各种仿真相比，这种后仿真包含的延时信息最为全面、准确，能较好地放映芯片的实际工作情况。有时为了保证设计的可靠性，在时序仿真后还要做一些验证。验证的手段比较丰富，可以用 ISE 内嵌时序分析工具完成静态时序分析（STA，Static Timing Analyzer），也可以用第三方验证工具（如 Synopsys 的 Formality 验证工具，PrimeTime 静态时序分析工具等）进行验证。需要强调的是，对于一般的逻辑设计，可能选择或者忽略复杂的验证手段，但是布局布线后时序仿真是必选项，只有通过时序仿真检查布局布线结果有无时序违规，才能在较大程度上保证设计的稳定与可靠性。

（7）调试与加载配置

设计开发的最后步骤就是在线调试或者将生成的配置文件写入芯片中进行测试，在 ISE 中对应的工具是 iMPACT。

在任何仿真或验证步骤出现问题，就需要根据错误的定位返回到相应的步骤更改或者重新设计。

3.3 软件开发工具

Xilinx 公司作为全球最大的可编程逻辑器件生产厂商，生产的 FPGA/CPLD 产品在全球范围得到广泛运用，其推出的软件开发工具 ISE 功能强大，易于使用，通过它可以完成整个 FPGA/CPLD 的开发过程。ISE 的界面风格简洁，有可视化编程技术，具有很大的辅助设计功能，可以大大减少设计者的工作量，提高了设计效率和设计质量，是开发以 Xilinx 公司 FPGA 器件作为核心的现代数字电路系统的首选工具。

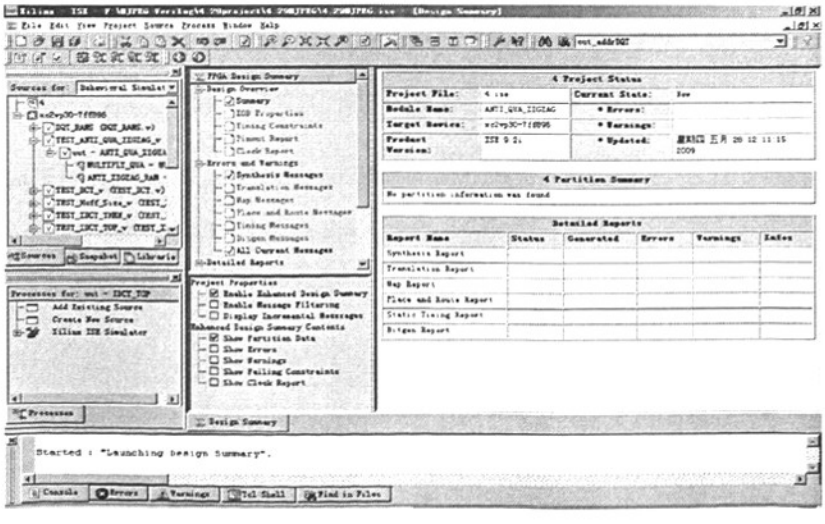

图3.2 ISE主界面

ISE 的主要功能包括设计输入、综合、仿真、实现和下载，涵盖了 FPGA 开发的全过程，从功能上讲，其工作流程无需借助任何第三方 EDA 软件。

设计输入：ISE 提供的设计输入工具包括用于 HDL 代码输入和查看报告的 ISE 文本编辑器（The ISE Text Editor），用于原理图编辑的工具 ECS（The Engineering Capture System），用于生成 IP Core 的 Core Generator，用于状态机设计的 StateCAD 以及用于约束文件编辑的 Constraint Editor 等。

综合：ISE 的综合工具不但包含了 Xilinx 自身提供的综合工具 XST，同时还可以内嵌 Mentor Graphics 公司的 Leonardo Spectrum 和 Synplicity 公司的 Synplify，实现无缝链接。

仿真：ISE 本身自带了一个具有图形化波形编辑功能的仿真工具 HDL Bencher，同时又提供了使用 Model Tech 公司的 Modelsim 进行仿真的接口。

实现：此功能包括了翻译、映射、布局布线等，还具备时序分析、管脚指定以及增量设计等高级功能。

下载：下载功能包括了 BitGen，用于将布局布线后的设计文件转换为位流文件，还包括了 iMPACT，功能是进行设备配置和通信，控制将程序烧写到 FPGA 芯片中去。

3.4 Verilog HDL语言概述[10]

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。Verilog HDL 是硬件描述语言中的一种，用于数字电子系统设计。设计者可以使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。Verilog HDL 是目前应用最广泛的一种硬件描述语言。

Verilog HDL 是 1983 年由 GDA（GateWay Design Automation）公司的 Phil Moorby 首创的。Phil Moorby 后来成为 Verilog-XL 的主要设计者和 Cadence 公司（Cadence Design System）的第一个合伙人。在 1984-1985 年间，Moorby 设计出了第一个名为 Verilog-XL 的仿真器；1986 年，他对 Verilog HDL 的发展又作出了另一个巨大的贡献——提出了用于快速门级仿真的 XL 算法。

随着 Verilog-XL 算法的成功，Verilog HDL 语言得到迅速发展。1989 年，Cadence 公司收购了 GDA 公司，Verilog HDL 语言成为 Cadence 公司的私有财产。1990 年，Cadence 公司决定公开 Verilog HDL 语言，于是成立了 OVI（Open Verilog International）组织，负责促进 Verilog HDL 语言的发展。基于 Verilog HDL 的优越性，IEEE 于 1995 年制定了 Verilog HDL 的 IEEE 标准，即 Verilog HDL 1364-1995；2001 年发布了 Verilog HDL 1364-2001 标准。

Verilog 的一个显而易见的特点是：使用它做数字电路的设计时，只需要根据标准的 Verilog 语言规范描述复杂的电路系统，再用一般软件的模块化思想描述系统的规格和功能，接着利用软件将所写的 Verilog 代码编译成电路。这样大大节省了人工将电子元件拼凑成电路的时间。具体来讲，应用 Verilog 进行工程设计的优点有以下几个方面：

（1）Verilog 支持几乎所有的数字系统设计层次的建模，包括算法、行为级建模、寄存器传输（RTL）级建模、门级建模、开关级建模。

（2）Verilog 支持行为功能建模方式、结构建模方式、数据流建模方式（对应于 Verilog 中的 assign 命令）。Verilog 行为描述语言作为一种结构化和过程化的语言，其语法结构非常适合于算法级和 RTL 级的模型设计。

（3）Verilog 语言是并发的，即具有同一时刻执行多任务的能力。一般来讲，编程语言是非并行的，但在实际硬件电路中许多操作都是在同一时刻发生的。

（4）Verilog 语言有时序的概念，在硬件电路中，从输入到输出总是有延迟存在的，为了描述这些特征，Verilog 语言需要建立时序的概念。因此使用 Verilog 不仅能描述硬件电路的功能，还能描述时序要求。

（5）Verilog 的层次描述在描述中显式进行时序建模。Verilog 提供了强大的硬件建模能力，内嵌任务和函数。Verilog 可以通过 PLI（编程语言结构）与 C 语言接口提供原语描述能力。

（6）硬件描述语言和人类语言一样，它的流行与传播也需要人能够提供一个广阔的应用领域或者平台。Verilog 之所以受到推崇，也是基于大量的 EDA 工具对 Verilog 语言的良好支持。

4 JPEG 解码器的设计与实现

4.1 JPEG解码器的总体结构

本文中，JPEG 解码器的设计采用自顶向下的方法，划分为读入码流模块、熵解码模块、反量化反 Zig-Zag 模块、IDCT 模块 4 个部分，对每个模块分别进行设计，最后再进行组合，完成整个模块的设计。在模块设计中，基于 Huffman 码字本身的特点和 JPEG 标准，提出一种快速判断码长的算法，同时提出 Huffman 码字分级结构，根据该结构快速查表获得 Huffman 解码的算法及其模块设计；采用行列分解的方法实现 IDCT 硬件算法，设计出 IDCT 模块。由于色彩空间转换并不属于真正的 JPEG 标准中，故本解码器设计不包括色彩空间转换部分。

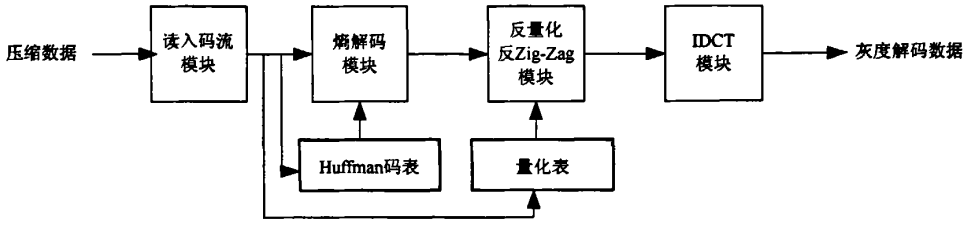

图 4.1 JPEG 解码器总体结构

JPEG 解码器总体结构如图 4.1 所示。读入码流模块从传输码流中获取数据，经过处理后输出给熵解码模块。熵解码模块接收压缩码流，完成图像编码数据的熵解码。反量化和反 Zig-Zag 模块对熵解码后的数据进行处理和排序后输出给 IDCT 模块，IDCT 模块输出 8 比特灰度解码数据。

本文在采用自顶向下的设计方法的同时，充分利用流水线改善系统性能。下面将对系统内部各模块的具体设计思想和实现过程进行介绍。

4.2 读入码流模块设计

按照上述的 JPEG 文件数据存储方式，JPEG 图像文件大体上可以分为两部分：标记段和压缩数据。其中标记段给出了诸如图像长度、宽度、量化表和 Huffman 表等重要信息，不同信息存在于不同的标记段中。在 JPEG 图像解码过程中需要从标记段中提取解码需要的各种信息，以便于后续对压缩数据的解码。当标记段解码完后，开始进入实质解码阶段，对压缩数据进行解码。

（1）逐字节读取 JPEG 文件内容。当读到 SOI 标记时，说明其确实是 JPEG 文件，继续向下读取。

（2）读取至量化表标记，若有，则将其所描述的量化表输出至存储量化表

的 RAM 中，若无，则调用解码系统中所默认的标准量化表，以准备进行量化。

（4）读取至哈夫曼表标记，将哈夫曼表的内容解压提取出来，存入相应结构中，并注意表明该哈夫曼表的类型和索引，以分别供 AC 和 DC 解码使用。

（5）读取至扫描开始，通过以上获得的各类信息对数据流以比特为单位进行解码，当遇到文件结束符时，立刻停止解码，并将所得信息输出。

在对 JPEG 所附带的文件信息进行解析，并提取了其中的大量有用信息，比如亮度 DC、亮度 AC 等 4 张哈夫曼表、量化信息以后，要进行的是对图像正文的信息数据所进行的熵解码。硬件实现利用 FSM（有限状态机）来进行控制。如下图所示：

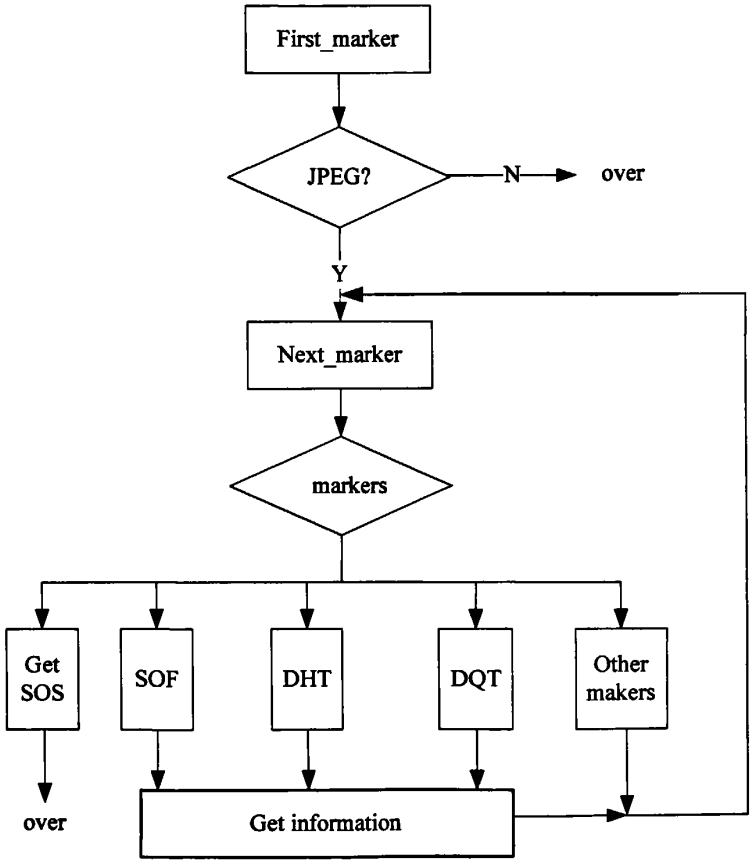

图 4.2 读入码流模块设计状态转换图

读入码流模块端口表示如下：

表 4.1 读入码流模块端口表

名称	宽度	功能
in_clk	1	系统时钟
in_rst_n	1	系统同步复位
in_en	1	使能信号
out_data	8	码流输出
out_weDQT	1	输出给量化表 RAM 的写信号
out_weDHT	1	输出给 Huffman 表 RAM 的使能信号
out_rdy	1	连接熵解码模块，表示输出压缩码流

4.3 熵解码模块设计

4.3.1 Huffman 解码原理

Huffman 算法是变长编码，压缩后产生的码字长度不固定。传统的解码方法必须逐位读入码流，先判断码字长度，再进行解码，效率相对较低。

本文针对 JPEG Huffman 编码表特点，结合现有的 Huffman 解码算法，提出了一种新的解码电路，能在一个时钟内判断出 Huffman 编码长度，而不用逐位操作判断，比传统方法节约了大量的判断时间；并使用 pipeline 结构，加快硬件解码速度。

4.3.1.1 Huffman 编码码字特性

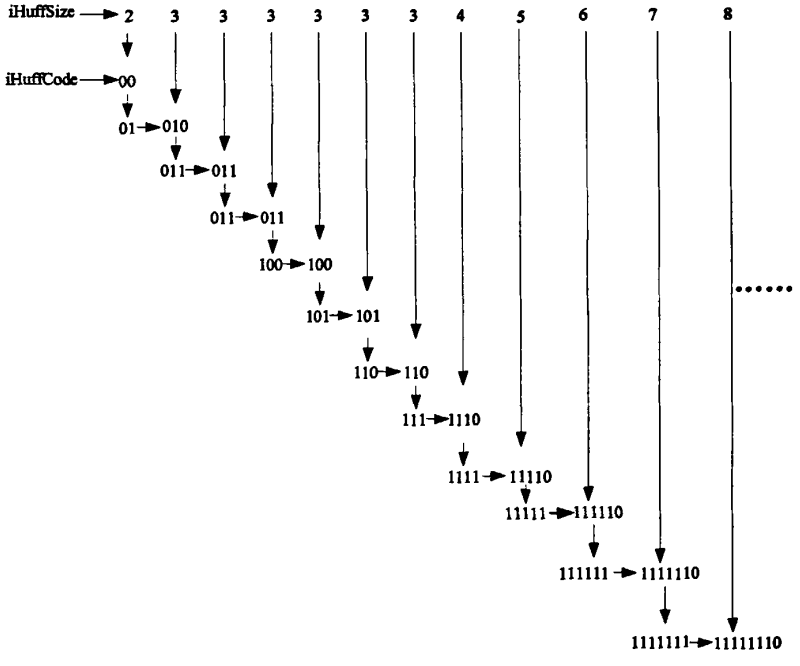

图4.3 JPEG Huffman编码产生流程图

图4.3 JPEG Huffman编码产生流程图，通过对JPEG Huffman编码产生流程图的观察，可以发现编码具有一定的规律。一是具有相同码长的码字是连续增长的；二是对于码长相连的两个码字集合中的码字存在一定的过渡规律，码长小的那个码字集中，最大的码字加1后，再在末尾补充若干个0以达到下一个码长的位数。这样形成的码字构成了下一码长的码字集中最小的码字。

由这个特性，很容易构造出Huffman编码的解码算法，即逐位读入码流，然后判断是否小于该长度的Huffman编码，如果小于则得出编码长度，就可以得到Huffman编码，否则继续读入下一位，然后继续上述判断。确定码长是解码过程的关键部分，而确定码长的算法效率影响着整个解码算法的效率。

4.3.1.2 Huffman变长解码器的解码算法

近年来，在一些实时应用场合中，人们对 Huffman 解码芯片的解码速度的要求越来越高，解码算法的研究也逐渐引起了人们的注意。尤其在一些嵌入式系统中，Huffman 解码模块往往成为限制系统速度的瓶颈。这主要是由于 Huffman 算法是变长编码，压缩后产生的码字长度不固定，因而在硬件实现上存在一定难度。

目前的针对硬件实现的 Huffman 解码算法主要有以下几种：

（1）二元树搜寻法，二元树搜寻法是逐位读入码流，然后判断是否存在此码长的码字，如果没有则继续读入一位码流来判断，直到找到正确的码字进行解码。此方法最节省存储空间，但解码效率低下，速度很慢。

（2）直接表对应法，此法是直接从码流中读取最大码长位数的数据，然后直接从存储器中查询匹配的码字，这种方法的解码速度比较快，但需要的存储空间比较大。如果此码流中最大码长为 k，则解码时需要耗费 2^k 的存储空间。

（3）分类群组搜寻法，是二元树搜寻法和直接表对应法的结合，它将查找表划分成几组子表，这样存储空间效率有较大的改善，但解码速度比直接表对应法小。

（4）范式哈夫曼表（Canonical Huffman Table）法，通过对 JPEG 码表的观察，可以发现码表具有一定的规律。这在上一小节内容中已作介绍，不再赘述。基于此规律我们可以对查找表进行重新构造，将码字按码长大小从小到大排列，并且将同一码长集中的码字同样按从小到大排列。然后将新查找表的码字所对应的原始符号（码表中需编码符号）数进行存储。另外在解码时，为了能够进行码长的判断，需要额外存储信息有每个码长集合中的最小编码以及它在存储器中的存储地址。从上述可以看出存储器利用率相对于其它方法有很大提高，和二元树搜寻法相当。另一方面，范式哈夫曼表法的解码主要分为三步，码长的判断、存储地址的计算以及存储数据的读取，这解码速度相对于二元树

搜寻法有很大提高。范式哈夫曼表法的存储器利用效率高，实现简单，适合较多的解码场合。

本文主要参考范式哈夫曼表的算法思路，在码长判定的方法上略加改进，使我们不必逐一比特探测码长，改进的电路能够在一个时钟周期判断码长，大大提高了整个 Huffman 解码器的运行速率。

下面我们来详细了解范式哈夫曼表算法的具体思想：

在解码算法中，将解码分解为检测 Huffman 码字长度，然后用码字长度作为查表依据得到解码输出。

首先，由码字特性可知，对应于特定长度的所有码字构成的集合中，将其单调排列时，其变化是连续的。因此，对于一个特定长度的码字，可以用该长度对应的最小码字和该码字相对于最小码字的偏差来表示。如对于码字长为 L 的特定码字 code，可以用 Min（L）和 code 相对于 Min（L）的偏差 offset 来表示：

$$code=Min（L）+offset \tag{4.1}$$

这一点可以用来在硬件实现时利用偏差寻址。将解码符号的存储结构按码字单调递增的顺序安排，则由最小码字的地址和偏差 offset 得到解码符号。解码结果的地址可由下得到：

$$\begin{aligned} &offset+Min（L）'s\ address=（code–Min（L））+Min（L）'s\ address \\ &\qquad =code+（Min（L）'s\ address-Min（L））\end{aligned} \tag{4.2}$$

后面括号内的值（对应于各个不同的长度）可预先存储在一个存储组织中，我们假设称其为偏差地址表存储器。

对于一个熵解码器，需包括 3 个独立的存储体：其一是存储各长度对应的最小码字，其二存储的是各长度对应的最小码字的地址，其三是存储解码符号。

举例以一个较小的Huffman解码器为例说明解码工作流程。表4.2表示一个Huffman表，表4.3表示依照表4.2经统计得到的各码字长度最小值表，表4.4根据码表得到的符号存储地址表，表4.5根据码表得到的偏差地址表。

表4.2 Huffman码表

符号	码字	长度	符号	码字	长度
S7	00	2	S6	111010	6
S22	01	2	S2	111011	6
S12	100	3	S10	111100	6
S1	1010	4	S21	111101	6
S19	1011	4	S20	1111100	7
S11	11000	5	S3	1111101	7
S14	11001	5	S13	1111110	7
S4	11010	5	S15	11111110	8
S5	11011	5	S16	111111110	9
S8	111000	6	S17	1111111110	10
S9	111001	6	S18	1111111111	10

表4.3 各码字长度最小值表

2	00
3	100
4	1010
5	11000
6	111000
7	1111100
8	11111110
9	111111110
10	1111111110

表4.4 符号存储表

地址	符号	长度	地址	符号	长度
0	S7	2	11	S6	6
1	S22	2	12	S2	6
2	S12	3	13	S10	6
3	S1	4	14	S21	6
4	S19	4	15	S20	7
5	S11	5	16	S3	7
6	S14	5	17	S13	7
7	S4	5	18	S15	8
8	S5	5	19	S16	9
9	S8	6	20	S17	10
10	S9	6	21	S18	10

表4.5 偏差地址表

码字长度	偏差系数
2	0
3	−2
4	−7
5	−19
6	−47
7	−109
8	−236
9	−491
10	−1002

将每个长度对应的最小编码存储在表 4.3 中，对于到来的比特流，可以通过与表 4.3 中的值依次比较，得到码字的长度 L，然后根据长度 L 在表 4.5 中得到地址偏差，从而计算出其对应码字的地址 offset＋the minimum code's address=（code–minimum code）+the minimum code's address=code+（the minimum code's address-minimum code）。根据地址在表 4.4 找到相应的码字，从而得到相应的符号，得到解码输出。其中表 4.5 存储的地址偏差即为等式中的（the minimum code's address-minimum code）一项。由于第一级确定码字的长度只需与各个长

度对应的最小码字做比较，而第二级，在确定长度为 L 的码字集中，更可以根据集内码字的单调性，直接通过地址偏差运算得到相应的地址，从而得到相应的符号，完成解码，所以大大减小了匹配的工作量，从而提高了解码效率。

当到来的码流为 11111011101011011.....先探测码字长度，取出各长度位对应的前缀码与表 4.3 中的值比较：

2bit：11>00

3bit：111>100

4bit：1111>1010

5bit：11111>11000

6bit：111110>111000

7bit：1111101>1111100

8bit：11111011<11111110

故得到码字的长度为 7bit，于是得到码字为 code＝1111101，同时根据长度 7 在表 4.5 中查表得到偏差为△＝－109，则可得其对应的符号的地址为 address ＝code＋△＝125－109＝16，根据地址 16 查表 4.4 即可得到符号 S3，即为相应的解码输出。

同上面的过程可以得到接下来的输出为S4，S5... ...由此可以看出，范式Huffman表算法存储器利用率高，但检测Huffman码字长度部分需要经过多次比较才能得到编码长度，本文在检测Huffman码字长度方法中加以改进，从而进一步提高了解码速率，这将在关键模块Get_Length模块中详细描述。

4.3.2 熵解码器的模块设计

4.3.2.1 熵解码器的总体设计

整个模块主要由两个模块构成，Huffman 解码模块，DPCM_RLD（差分脉冲及游程长度解码）模块。输入数据送入 Huffman 解码模块，经过内部电路判定待解码部分属于 AC 子带或者 DC 子带，从而查相应的 Huffman 表进行解码，然后将解码后的数据存入 FIFO 中，DPCM_RLD 模块从 FIFO 中取出数据进行相应的解码，然后发往后续模块。整个模块的结构框图如图 4.4 所示。

图4.4 熵解码器模块框图

熵解码器模块接口说明如表4.6所示：

表4.6 熵解码器模块接口说明

名称	宽度	功能
in_clk	1	系统时钟
in_rst_n	1	系统同步复位
in_din	8	压缩数据
in_en	1	使能信号
out_Y_	1	输出亮度分量指示
out_data	11	熵解码输出
out_DC	1	直流分量指示
out_rdy	1	输出数据有效

4.3.2.2 Huffman 解码模块设计

Huffman 解码实现电路结构图如下：

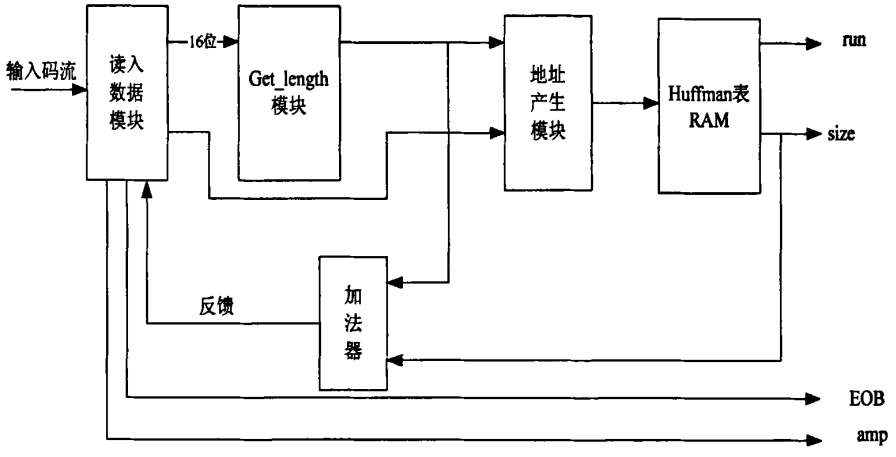

图4.5 Huffman解码实现电路结构图

由于JPEG的Huffman码表的码字的最大码长为16，为了能够一次性解码出一个码字，需要截取码流中16位的数据进行解码。另外由于解码的数据流是由编码和二进制数值大小构成，而在本文所用的码表中，二进制数值最大为11位。所以完整地解码一次码流所需的最小位数为27位，由于读文件模块输出的压缩数据为8bit，本文设置一个32位的变量进行解码。当一次解码完成后，必须将已解码过的数据从变量中移出，重新读入数据进行下一次解码。读入数据通过Get_Length模块则可确定此段数据流的Huffman码长。码长判断后，在地址产生模块中计算码字的存储地址。对于JPEG标准而言，Huffman码表包括亮度DC表、亮度AC表、色度DC表和色度AC表，所以必须设置四组Get_Length模块和相应的地址产生模块，以进行各个码表的解码。在随后取出的数据中，依据解码结果所表达的原编码数据的位数（对于DC分量是整个的值，对于AC位数是后四位值），从码流中取出数值。最后用码长与原编码数据的位数之和计算出变量中应该移出的数据位数，传递给移位寄存器控制模块，移出已解码的Huffman位数以及原编码的二进

制数amp。

4.3.3 关键模块的实现

4.3.3.1 读入数据模块

读入数据模块的主要完成接收固定长度（32 比特）的数据构成的位流，输出待解码的比特部分。其实现原理如图 4.6 所示。

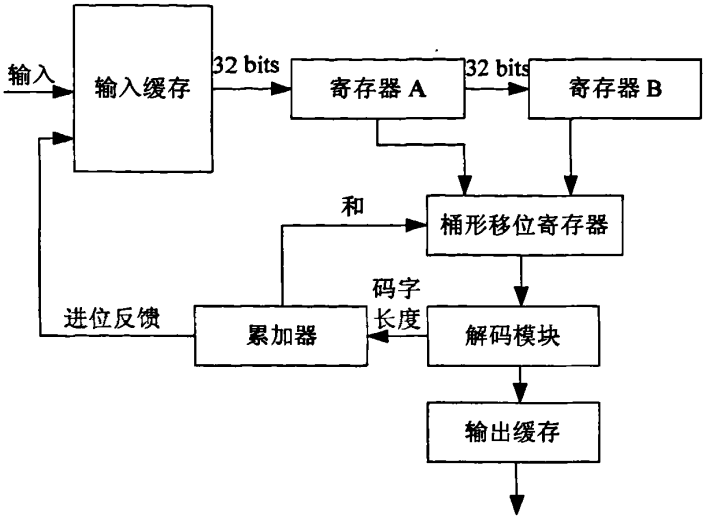

图4.6 读入数据模块实现原理图

在解码完一个码字进行下一个码字的解码时，必须对数据进行移位，所以解码器中肯定会涉及到移位操作。由于一般的移位寄存器在一个时钟周期内能够移位输出的数据宽度是固定的，而可变长码字是变长的，因此在移位时必须要用到桶形移位寄存器（Barrel shifter register）。它在一个时钟周期内可以输出不超过其数据宽度的任意宽度的数据。

在初始状态下，寄存器 A 和寄存器 B 的值为 0。解码开始时，首先从输入缓存中读出一个数据送到寄存器 A，原寄存器 A 中的值送到寄存器 B 中。累加器的输出是{寄存器 B，寄存器 A}存储的数据中已处理的比特数。通过桶形移位寄存器后，其输出的即是待处理的数据。

通过解码模块后将解码得到的码字码长送到累加器，修改累加器的值，以便进行下一轮的控制。同时，当累加器中的值大于 32 比特时，向输入缓冲发出读请求，寄存器 A 中读入新的值，寄存器 B 中的值也同时更新。这样反复，达到连续解码的功能要求。

4.3.3.2 Get_Length模块

之前我们已经介绍过可以逐个比特读入数据与表 4.3 中的值依次比较，得到码字的长度。但是这种方法数据率较低，在此提出了一种新的解码电路，能在一个时钟内判断出 Huffman 编码长度，而不用逐位操作判断，节约了大量的

判断时间；并使用 pipeline 结构，加速硬件解码速度。

结合编码表特性，如果将码字后面补0扩展到m（m为码表中的最大码长位），则码长小的码字其扩展后的值一定小于码长大的码字扩展后的值。例如：若某个Huffman码表中，码长为6的最后一个码字为111001，码长为7的第一个码字1110100，将两个码都扩展到16位，两者一比较得证。

JPEG 标准中 DC 的 Huffman 编码长度从 2 位到 11 位，AC 的 Huffman 编码长度从 2 位到 16 位。为了让硬件既可以解 AC 编码，又可以解 DC 编码，构建了如图 4.7 所示的 Get_Length 电路框图。因为每张 JPEG 图像中，DC，AC 的 Huffman 编码都存储在 JPEG 头中 DHT 中，其中一般有 4 个 Huffman 表，Y 分量 DC，AC 的 Huffman 编码表，Cb 和 Cr 分量共用一个 DC，AC 的 Huffman 编码表。根据 DHT 表中关于 DC，AC 的 Huffman 表的长度和编码规律，就可以计算不同码长时对应的最小值了，将最小值扩充到 16 位，不足位末尾添“0”，用 mincode[n]表示，其中 n 代表码长。在读取 JPEG 头信息时，就可以计算出这个值。将编码好的 mincode[n]存储到 reg_mc0 到 reg_mcl4 共 15 个寄存器中，然后取出 JPEG 数据流的 16 位数据存入寄存器 reg_code 中，利用 15 个 16 位的比较器，同时比较 reg_mc0 到 reg_mcl4 寄存器数值与 reg_code 寄存器数值，比较输出值编码为 reg_HF[14：0]，如图 4.6 所示。根据 reg_HF[14：0]的值，经过 Judge_length 模块，就可以得到 reg_code 中有效的 Huffman 编码长度，然后在得到 Huffman 编码。

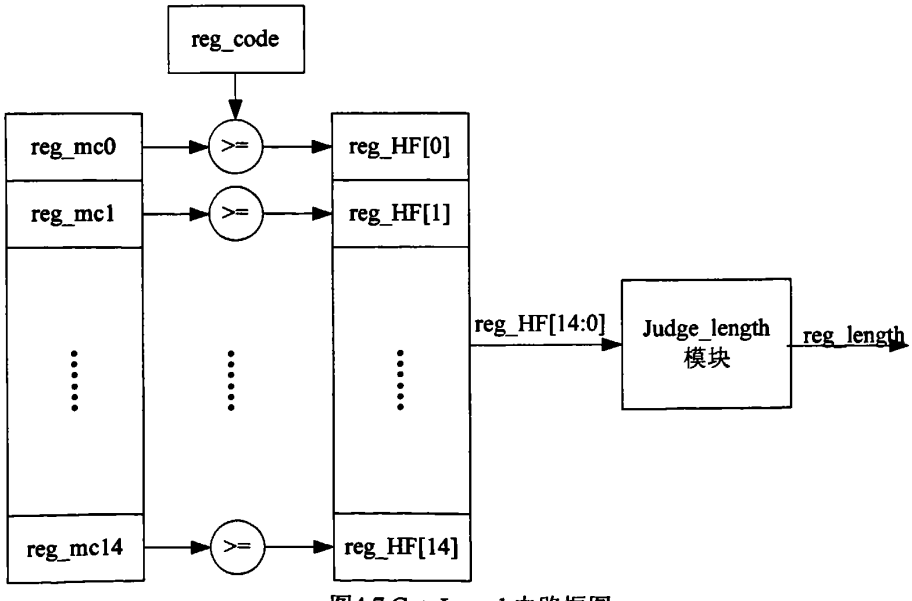

图4.7 Get_Length电路框图

Judge_length代码表示如下：

reg[4：0] reg_length;

```
always @（posedge in_clk or in_rst_n）
   if （!in_rst_n）
      begin
     reg_length<=5'00000;
      end
      else
        begin
          case（reg_HF[14：0]）
              15'b000_0000_0000_0001：reg_lenth<=4'd0;
              15'b000_0000_0000_0011：reg_lenth<=4'd1;
              15'b000_0000_0000_0111：reg_lenth<=4'd2;
              ……
              15'b011_1111_1111_111：reg_lenth<=4'd15;
              default：reg_lenth<=4'd0;
            endease
        end
```

4.3.3.3 DPCM_RLD模块的设计

DPCM_RLD 的功能是在 Huffman 解码完成之后的后续处理，该模块负责从 FIFO 中取出数据，进行 DPCM 和 RLD 解码。对 DC 子带的数据进行 DPCM 解码，对 AC 子带的数据进行 RLD 解码。解码出来的串行数据经过一个计数模块，计数模块对有效数据进行计数，并判断数据属于哪个子带，由此给下级模块提供子带号信息。DPCM_RLD 模块运行时钟频率较高，与下级模块的使用同一个时钟。解码模块实现框图如图 4.8 所示。

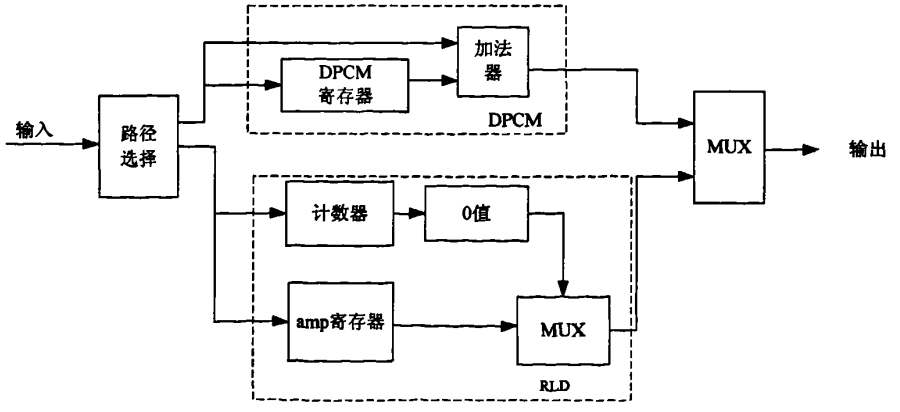

图4.8 DPCM_RLD模块实现电路图

当输入为 DC 子带数据时，DPCM 解码模块被激活，将输入与 DPCM 寄存器中存储的上一个 DC 值相加，其解码得到的数据送到多路选择器。当输入数

据为 AC 子带数据时，RLD 模块被激活，它将数对（run，amp）转化为连续的零串和紧随其后的非零值，具体来说，根据 run 值设定计数器，输出 run 个 0，送入选择器；而对于 amp 值进行 VLI 解码，也就是当 amp 最高位为 0 时，表明其为正数，最高位为 1 时，各位取反，送入选择器，选择输出 AC 子带解码结果。DPCM 和 RLD 解码以后的输出经过一个多路器输出。

4.4 反量化与反Zig-Zag模块的实现

4.4.1 反量化与反 Zig-Zag 设计思想

JPEG解码器要利用文件头信息中的量化表对量化值进行译码。JPEG文件里一般含有两个量化表：一个亮度分量的量化表，一个色度分量的量化表。反量化就是对Huffman解码出来的系数矩阵乘上相应的量化矩阵：

$$F_{uv} = C_{uv} \times Q_{uv}$$

其中：C_{uv}代表Huffman解码输出，Q_{uv}代表相应的量化矩阵。

为了进行后面的IDCT，还需要将完成反量化运算的数组进行反Z字型扫描，反Z扫描的顺序如表4.7。这样，一个MCU经过反量化和Z字型变换后，就得到了8×8的亮度数组和色度数组。

表4.7 反Zig-Zag编码表

0	1	8	16	9	2	3	10
17	24	32	25	18	11	4	5
12	19	26	33	40	48	41	34
27	20	13	6	7	14	21	28
35	42	49	56	57	50	43	36
29	22	15	23	30	37	44	51
58	59	52	45	38	31	39	46
53	60	61	54	47	55	62	63

4.4.2 反量化与反 Zig-Zag 模块的设计

通过前面对反量化部分内容的分析可以得出：反量化的目的就是对解码出的数据通过乘法而产生重构的IDCT系数。这个过程的实质是完成量化表中的系数与解码出的数据的乘法运算。反量化反Zig-Zag模块的实现如图4.9所示，它由一个存储反量化表的存储器、一个乘法器以及存储反量化后系数的存储器和按照反Zig-Zag顺序的地址发生模块构成。

反Zig-Zag模块是根据表中提供的矩阵数据位置的转换顺序来对量化后的矩阵数据重新进行排序。它需要一对深度为8×8的RAM来对矩阵数据进行存储，同

时外加一个驱动RAM工作的地址产生电路。之所以需要一对RAM，是为了加快解码的效率。因为在JPEG解码电路的各个流程采用流水线的方式进行通讯，那么在反"Z"扫描的模块中，不仅需要一个RAM来接收量化后系数，还需要另一个RAM输出数据供IDCT模块使用的数据。

在一个寄存器阵列中共有64个16位的寄存器，而一个8×8数据单元经过反量化后非零的数据大约只有整个数据的10%，因此在对寄存器阵列赋值前，将所有的寄存器初始化为零，然后在具体解码时，将非零数据赋给地址正确的寄存器，其它位置的寄存器数据可保持零。这样可以避免对零数据的存储，加快此环节的处理速度，使得对于一个8×8数据单元的反Zig-Zag的扫描操作可在对同样单元进行Huffman解码所需要的周期数内同步地完成，提高了流水的效率。

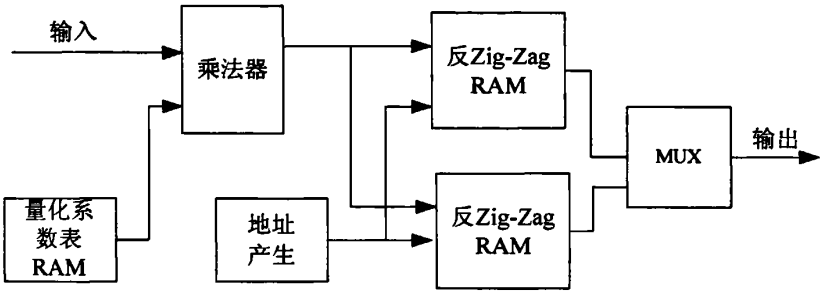

图4.9 反量化反Zig-Zag扫描模块实现框图

表4.8 反量化反Zig-Zag扫描模块接口说明

名称	宽度	功能
in_clk	1	系统时钟
in_rst_n	1	系统同步复位
in_data	11	量化后 DCT 系数
in_en	1	使能信号
in_Y	1	亮度/色度指示
out_data	18	反量化反 Z 扫描后系数
out_rdy	1	输出数据有效

4.5 IDCT模块的设计

4.5.1 IDCT 算法设计

当要恢复原始图像信息时，就需要将压缩编码后的信息进行离散余弦逆变换（IDCT）。对于8×8矩阵数据的IDCT变换如式（4.3）所示，$x(i,j)$是原始图像像素值，$Z(u,v)$是各频率分量的大小，$i,j,u,v=0,1,2...,7$。IDCT是图像解码过程中计算量最大的部分，采用一种快速有效的IDCT算法对图像的解码尤为重要。

$$x(i,j)=\frac{1}{4}C(u)C(v)[\sum_{u=0}^{7}\sum_{v=0}^{7}Z(u,v)\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}] \tag{4.3}$$

$$\begin{cases}u,v=0\text{时}, & C(u),C(v)=1/\sqrt{2}\\ u,v\neq 0\text{时}, & C(u),C(v)=1\end{cases}$$

二维 DCT/IDCT 快速变换的实现方法主要有两种：第一种为行列分解法，即利用二维 DCT/IDCT 变换的可分解特性，首先对 DCT/IDCT 矩阵的每一行进行一维 DCT/IDCT 变换，然后对变换结果的每一列进行一维 DCT/IDCT 变换，从而得到二维 DCT/IDCT 变换的结果。这种算法的关键是提高一维 DCT/IDCT 变换的效率以适应不同的软硬件环境。对于 8×8 字块，需要进行 16 次的一维 8 点 DCT/IDCT 变换。

第二种是采用直接法实现二维 DCT/IDCT 变换。直接法是通过多项式转换或者三角分解法把要进行二维 DCT/IDCT 变换的 8×8 字块分成 8 组进行一维 DCT/IDCT 变换，然后把一维 DCT/IDCT 变换的结果进行加减组合，得到二维 DCT/IDCT 变换结果。直接法只需 8 次一维变换和附加组合运算。

两种方法相比，第二种方法实现二维 DCT/IDCT 变换只需要 8 个一维 DCT/IDCT 减少了资源消耗，但是时序控制复杂。第一种方法的算法规律性强，实现结构直观，时序控制简单，论文中采用了行列分解算法结构。

二维DCT/IDCT的矩阵形式如公式（4.4）、公式（4.5）所示：

$$Z = AXA^T \tag{4.4}$$

$$X = A^T ZA \tag{4.5}$$

由于二维 DCT/IDCT 变换是可分离的，它可以分解成串联的二次一维变换，首先我们来看 DCT，可以把二维 DCT 分解成两个一维的 DCT，因此式(4.4)可以写成：

$$Z = AY \tag{4.6}$$

$$Y = XA^T \tag{4.7}$$

这里，A 为带余弦基本函数的变换系数矩阵，A^T 为 A 的转置。A 是一个正交矩阵，即 $AA^T = I_N$。

$$A=\begin{bmatrix} a & a & a & a & a & a & a & a \\ b & d & e & g & -g & -e & -d & -b \\ c & f & -f & -c & -c & -f & f & c \\ d & -g & -b & -e & e & b & g & -d \\ a & -a & -a & a & a & -a & -a & a \\ e & -b & g & d & -d & -g & b & -e \\ f & -c & c & -f & -f & c & -c & f \\ g & -e & d & -b & b & -d & e & -g \end{bmatrix}$$ 其中，$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{bmatrix}=\frac{1}{2}\begin{bmatrix} \cos\frac{\pi}{4} \\ \cos\frac{\pi}{16} \\ \cos\frac{\pi}{8} \\ \cos\frac{3\pi}{16} \\ \cos\frac{5\pi}{16} \\ \cos\frac{3\pi}{8} \\ \cos\frac{7\pi}{16} \end{bmatrix}$$

我们用第一级DCT变换来实现$Y=XA^T$，由于A^T的偶数列为偶对称，奇数列为奇对称，公式（4.7）可分解成两个4×4的矩阵乘的形式，从而使运算量减半，如公式（4.8），公式（4.9）所示。

$$\begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix}=\begin{bmatrix} a & a & a & a \\ c & f & -f & -c \\ a & -a & -a & a \\ f & -c & c & -f \end{bmatrix}\begin{bmatrix} X(0)+X(7) \\ X(2)+X(5) \\ X(4)+X(3) \\ X(6)+X(1) \end{bmatrix} \tag{4.8}$$

$$\begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix}=\begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix}\begin{bmatrix} X(0)-X(7) \\ X(2)-X(5) \\ X(4)-X(3) \\ X(6)-X(1) \end{bmatrix} \tag{4.9}$$

同样地，我们也可以将式（4.5）分解成两个一维的IDCT运算，利用矩阵A的对称性，将一维IDCT分解得到：

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix}=\begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix}\begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix}+\begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix}\begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix} \tag{4.10}$$

$$\begin{bmatrix} X(7) \\ X(6) \\ X(5) \\ X(4) \end{bmatrix}=\begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix}\begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix}-\begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix}\begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix} \tag{4.11}$$

由于系数矩阵全是浮点数，为了能用定点乘法器运算，将它们扩大2^{16}倍后再取整参与运算。

注意：由于编码时，DCT 变换的输入系数范围是-128~127，所以在 DCT 变换前将数据都减去了 128，相应地对 IDCT 的输出系数应加上 128，然后再进行内插和色彩空间的转换。

4.5.2 IDCT 模块设计

4.5.2.1 总体结构

本课题采用基于行列分解算法和查找表的结构，主要分为三个单元，分别是一级一维 IDCT 单元、转置存储器单元、二级一维 IDCT 单元。总体的模块结构设计如图 4.10 所示，模块接口信息如表 4.9 所示：

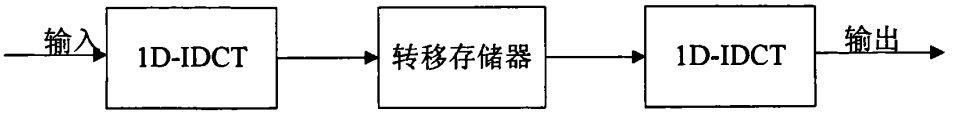

图 4.10 模块结构设计

对 8×8 矩阵先逐行进行一维 IDCT 变换，所得数据存储到转置存储器中，经过转置存储器进行矩阵转置并输出，即将一维行变换的结果按列输出，再将数据送入一维列 IDCT 变换模块，经运算所得输出即为二维 IDCT 的结果。

表 4.9 IDCT 模块接口说明

名称	宽度	功能
in_clk	1	系统时钟
in_rst_n	1	系统同步复位
in_din	18	DCT 系数
in_en	1	使能信号
out_rdy	1	输出数据有效
out_data	21	灰度数据输出

4.5.2.2 第一级 IDCT 单元

图 4.11 是实现一维 IDCT 变换的电路结构，它主要由串行到并行转换电路、系数矩阵循环移位寄存器、加法器、乘法器等电路组成。串行到并行的转换目的是对串行输入的数据进行并行化处理，以便于后面的 1D-IDCT 单元进行运算。

对于一个 8×8 图像块来说，共有 64 个输人数据，它们以串行的方式输入到电路中来，每行数据（共 8 个）为一组，分别为 Y0~Y7。当第 1 组数据输入进来后，经过串/并转换器后在寄存器中保持 8 个时钟，在具体实现过程中，需定义一个八进制计数器，对串行输入的数据进行计数。当计数器计数到 8 的时候，输出一次并行数据。

将 64 个系数分别存储在 8 个循环移位寄存器中，在每个时钟周期寄存器循

环右移一次，并行输出的 8 个系数分别与输入的 8 个数送入乘法器相乘，最后进行 8 个乘积相加及去低 16 位处理，去除低 16 位是为了抵消前面对 IDCT 系数所做的2^{16}倍放大。至此，完成了一行数据的一维 IDCT 运算。随着 64 个输入数据源源不断地送人，在第 10～73 个时钟期间，所有 64 个一维 IDCT 变换结果将一一输出，所得结果输入转置存储器单元。

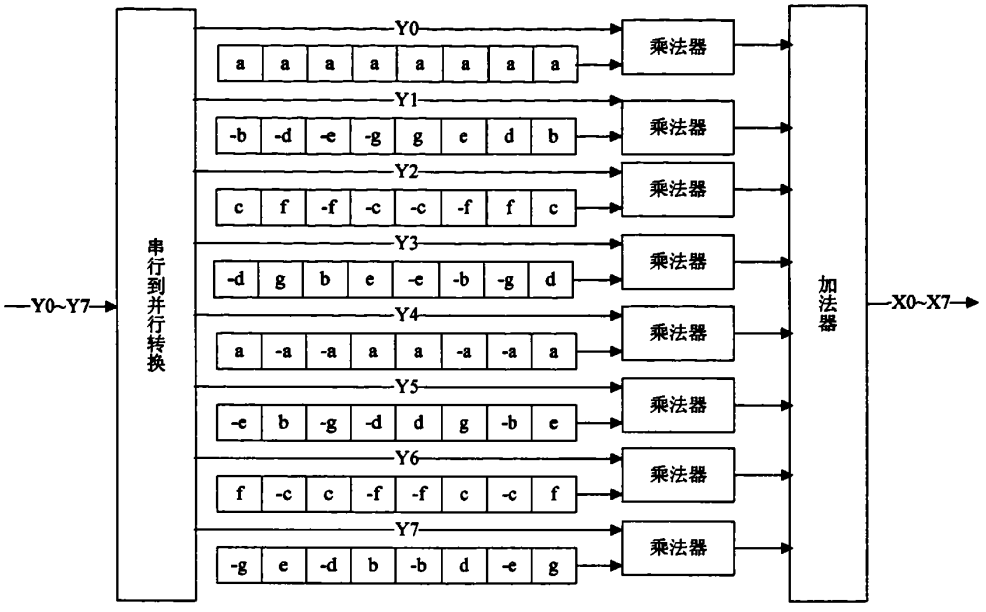

图 4.11 1D-IDCT 模块电路结构

4.5.2.3 转置存储器

二维 IDCT 可分解为两个一维 IDCT 实现，转置 RAM 是对前面的一维 IDCT 结果做行列转置，即将逐行输入的 8×8 个数据转换为逐列输出。从第 1 个数据输入到第 1 个数据输出，中间需要 64 个时钟间隔。对一个深度为 64 的 RAM 来说，要先用 64 个时钟完成数据写操作，再用 64 个时钟完成数据读操作。考虑到时序的合理性和连贯性，即前后两级 1D-IDCT 都在连续工作，转置存储器必须能够同时接受第一级 1D-DCT 的数据输出并为第二级 IDCT 提供输入数据源。因此，在设计时采用了双 RAM 结构，并让这两个 RAM 工作在交替读/写模式。

具体的工作模式如下：假设前 64 个周期写满第一个 RAM，第 65 个时钟开始写第二个 RAM，同时开始从第一个 RAM 里读取数据，读取顺序是按列进行的。这样，在花费 64 个周期后，读完第一个 RAM 里的数据，此时第二个 RAM 也已经写满。然后开始读第二个 RAM 的数据，同时写第一个 RAM……依此进行下去，有效提高了数据吞吐率。图 4.12 给出了转置存储器的电路结构。

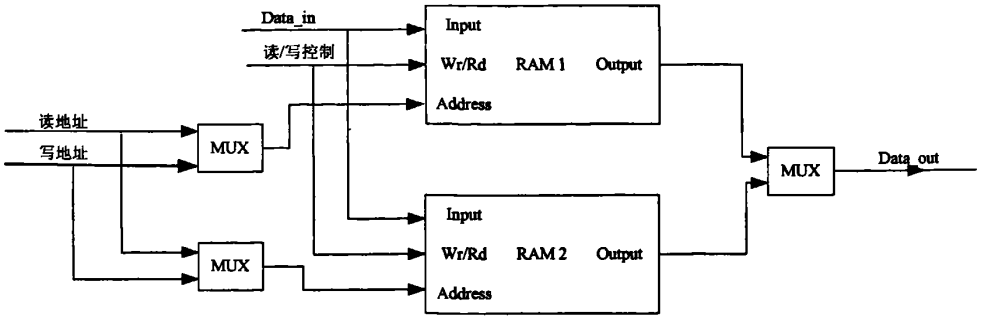

图 4.12 转置存储器模块电路结构

4.5.2.4 第二级 1D-IDCT 单元

第二级 1D-IDCT 逐列完成变换，和第一级 1D-IDCT 采用相同的快速算法，因此这一部分的电路结构与第一级 ID-DCT 基本相同，不同之处仅在于所处理的数据长度不同（比第一级 ID-IDCT 略长），因此，在电路实现上耗费的片内资源会略多些。从时序上来看，第 75 个时钟开始接受数据输入；而输出数据则从第 84 个时钟开始。

5 JPEG 解码器的验证与分析

5.1 测试平台

测试平台是为了向被测设计对象施加输入模式（即激励）而建立出来的那一层代码，它包括测试激励的生成、输出响应的比较两部分。如图 5.1 所示，所建立的测试平台向被测设计对象施加输入激励，对其输出进行采样，并将其输出与期望结果进行比较。如果采样出来的输出与期望结果不一致，那么测试平台将生成错误报告。如果报告中标明输入、所得的输出、期望结果等状态信息及错误信息，将有助于分析被测设计对象的功能细节，也可以将错误的成因隔离出来。

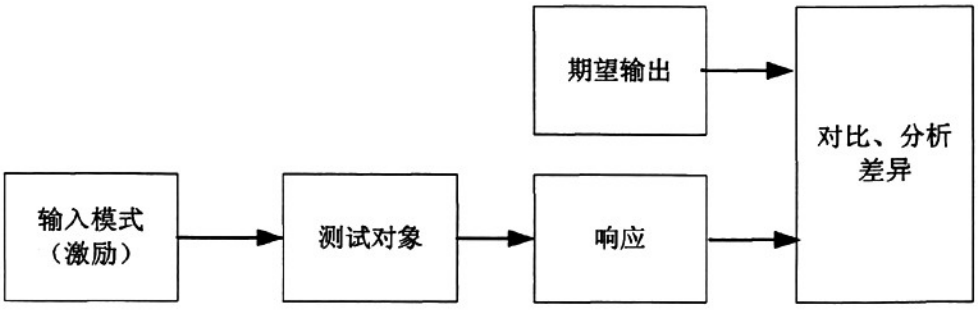

图 5.1 测试平台

本文中用到的测试平台是 ISE 9.2 自带的仿真工具，接下来给出各个模块的测试结果。

5.2 读入码流模块的仿真与验证

使用 Ultra edit 软件打开.jpg 文件，由图 5.2 可以很清楚地看到压缩后的数据。由于读入模块输出端口较多，不一一证明，举例验证当读取至 DQT 量化标志段，即“FFDB”，输出是否正确。

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00000010h:	00	64	00	00	FF	DB	00	43	00	08	06	06	07	06	05	08
00000020h:	07	07	07	09	09	08	0A	0C	14	0D	0C	0B	0B	0C	19	12
00000030h:	13	0F	14	1D	1A	1F	1E	1D	1A	1C	1C	20	24	2E	27	20
00000040h:	22	2C	23	1C	1C	28	37	29	2C	30	31	34	34	34	1F	27
00000050h:	39	3D	38	32	3C	2E	33	34	32	FF	DB	00	43	01	09	09
00000060h:	09	0C	0B	0C	18	0D	0D	18	32	21	1C	21	32	32	32	32
00000070h:	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
00000080h:	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
00000090h:	32	32	32	32	32	32	32	32	32	32	32	32	32	32	FF	C0

图 5.2 压缩文件数据

ISE 仿真读入模块后，输出仿真波形如图 5.3 所示：

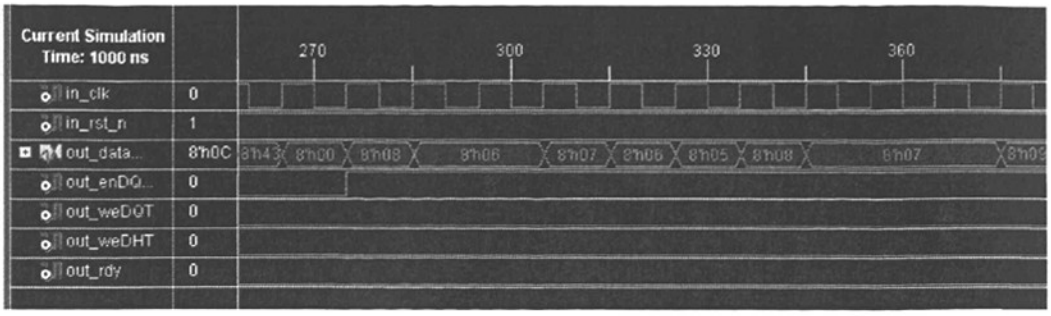

图 5.3 读入码流模块仿真波形

“FFDB”接下来的数据从“08”开始才是真正的量化表系数，由图 5.3 可知，此时数据输出为“08 06 07……”，连接 DQT 存储器使能端产生上升沿，其余的输出管脚保持低电平，因此得到了需要的结果，模块结果经验证正确。

5.3 DPCM_RLD模块的仿真与验证

模块主要输入为 in_dc（指示数据是否属于 DC 子带），in_run（连零个数），in_size（VLI 编码位数），in_amp（VLI 码被扩充到 11 位，高位补 0）；输出为 out_data（输出数据），out_rdy（输出数据有效性）。

表 5.1 主要输入端数据

in_dc	1	0	0	0	0	0	0	0	0	0	0	0	0	0
in_run	0	0	0	1	0	0	0	0	1	2	1	2	0	0
in_size	2	5	2	2	1	3	1	3	2	3	2	1	1	0
in_amp	11	00101	01	00	0	001	1	011	00	100	10	0	1	0

模块仿真波形如图 5.4 所示，综合 out_data 以及 out_rdy 确定模块有效数据输出为：

（3，-26，-2，0，-3，-1，-6，1，-4，0，-3，0，0，4，0，2，0，0，-1，1，45 个 0）。与手动计算结果相同，模块结果验证正确。

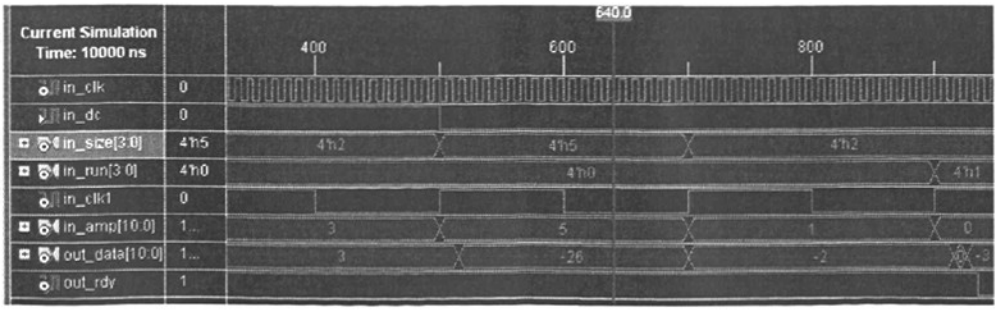

图 5.4 DPCM_RLD 模块仿真图

5.4 反量化反Zig-Zag模块的仿真与验证

利用下面的矩阵对反量化反 Zig-Zag 模块进行测试，输入的测试矩阵如表 5.2，量化表采用的是图 5.2 所示的第一个反量化表（亮度反量化表），经 ISE 9.2 仿真得到反量化排序后的矩阵如表 5.4 所示。表 5.3 为手动计算的实际值，通过比较可以发现结果是一致的。

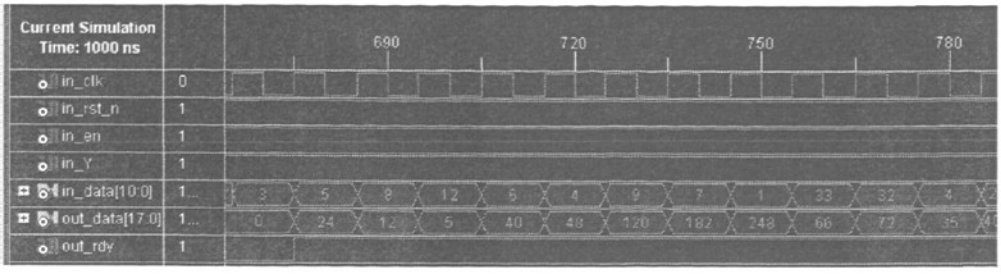

图 5.5 反量化反 Zig-Zag 模块的仿真波形图

表 5.2 输入矩阵

3	2	1	5	4	6	7	8
11	12	5	4	3	1	2	4
1	2	7	3	5	0	9	5
6	9	11	3	6	13	9	3
3	5	8	7	2	11	21	2
5	3	6	7	8	0	9	1
2	1	4	5	9	5	3	2
3	4	6	8	9	2	3	4

表 5.3 手动计算输出矩阵

24	12	5	40	48	182	248
66	72	35	40	39	60	112
7	14	56	36	100	315	140
42	81	121	45	156	360	93
27	55	152	196	68	1092	78
60	54	168	224	328	513	46
50	32	156	220	468	180	102
108	184	288	392	504	156	200

表 5.4 ISE 仿真输出矩阵

24	12	5	40	48	120	182	248
66	72	35	40	39	29	60	112
7	14	56	36	100	0	315	140
42	81	121	45	156	572	360	93
27	55	152	196	68	605	1092	78
60	54	168	224	328	0	513	46
50	32	156	220	468	305	180	102
108	184	288	392	504	100	156	200

5.5 IDCT模块的仿真与验证

对 IDCT 模块给予测试，仿真得到仿真波形图如图 5.6 a)、5.6 b）所示。然后用 matlab 对编写 IDCT 程序，结果取整后与仿真结果比较。

设置 IDCT 模块的输入矩阵：

235	-1	-12	-5	2	-1	-2	1
-22	-17	-6	-3	-2	0	0	-1
-10	-9	-1	1	0	0	0	0
-7	-1	0	1	0	0	0	0
0	0	1	1	0	0	0	1
1	0	1	0	0	1	1	-1
-1	0	0	1	0	1	1	0
-2	1	-3	-1	1	1	0	0

该矩阵经 matlab 仿真取整得输出结果为：

12	16	22	26	27	27	27	27
16	22	25	28	30	27	28	28
23	27	31	34	31	29	28	28
30	32	33	33	32	30	30	30
31	33	34	34	34	28	28	27
30	30	32	32	32	29	28	28
33	34	34	35	33	30	29	29
34	33	33	33	34	30	30	30

通过 ISE 仿真计算后输出为：

11	16	23	30	27	27	28	29
15	22	26	30	33	27	27	28
20	27	33	33	31	29	28	28
24	32	33	33	31	30	31	31
27	33	35	35	35	29	28	27
32	33	32	32	32	29	28	28
32	34	34	35	33	30	29	29
33	33	33	33	34	31	31	31

经过比较可以看到输出结果存在误差，最大误差为±4，分析导致的原因是：硬件截位和有效字长。

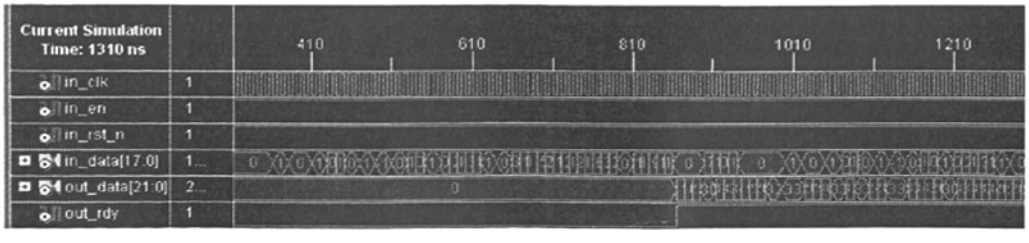

图 5.6 a） IDCT 仿真波形图

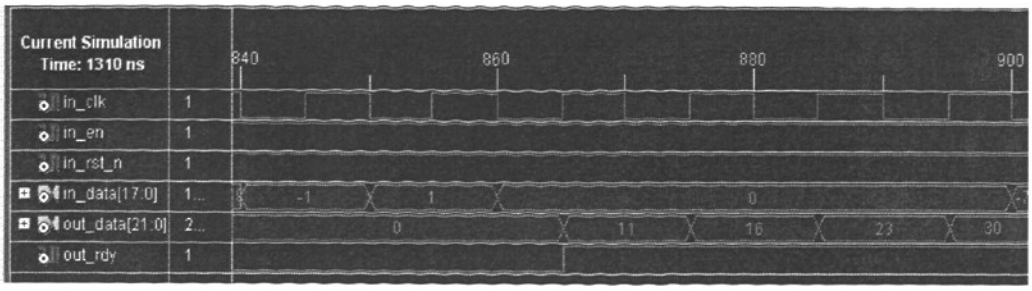

图 5.6 b） IDCT 仿真波形图

6 总结与展望

6.1 总结

本文完成的主要工作包括以下几个方面：

（1）深入调查研究了国内外图像压缩技术的应用研究发展状况，介绍了JPEG静止图像压缩编码系统，主要算法的基本原理及其压缩文件格式。并针对本系统，对JPEG压缩解码算法的主要步骤也进行了比较详细的阐述。

（2）对FPGA技术、FPGA的结构特点及其开发流程和开发工具以及开发语言等进行了介绍，为本文的JPEG图像解压缩算法的设计实现提供了技术支撑。

（3）提出JPEG压缩解码器的总体设计方案，设计了整个系统的各个模块，并给出设计思路，设计步骤以及仿真结果。其中，对影响解码芯片速度和面积的关键技术及难点：IDCT变换和熵解码算法，本文进行了改进和优化，整个设计结构满足更小、更快的设计思想。

（4）使用ISE自带的仿真开发工具，对整个设计系统中的主要模块进行了系统级仿真验证。并对系统模块测试结果进行分析与调试，分析结果充分表明：该系统设计基本符合要求，具有一定的可行性

综上所述，本文针对JPEG标准的基本系统，应用Verilog HDL硬件描述语言，设计并实现了基于FPGA的JPEG图像解码主要模块。设计充分利用了自顶向下的设计方法，使整个系统设计思路清晰、电路简单、功能实用、抗干扰能力强、工作稳定；采用流水线优化算法有效地解决时间并行性问题。

6.2 展望

图像压缩解压缩算法是一个很有前景的研究领域，特别是随着图像压缩标准的建立以及集成电路技术的发展，图像视频的压缩及解压芯片已成为多媒体技术的核心，多媒体芯片研究已成为信息产业的新热点，在这方面投入一定的资金和人力进行深入地研究，将有助于提高我国在高科技领域方面的竞争力。

在本课题的研究中，通过对JPEG图像压缩理论和基于FPGA的开发技术的研究，对JPEG解码系统的结构设计和模块化设计等方面作了深入学习和研究。但是由于工作量较大，时间有限，在未来的工作中还有许多可以改善和补充的方面，主要包括以下几点：

（1）继续研究熵解码模块的设计与实现，总结和纠正现有设计中的错误，实现完整JPEG系统的设计。

（2）优化设计方案，提升系统可运行的频率。

（3）进一步简化系统硬件结构，对整个图像压缩系统设计硬件电路并下载到FPGA芯片中运行，达到系统可实现的目标。

致 谢

在学业即将完成之际，我要衷心感谢我的导师钱玲副教授。在两年的研究生学业中，无论是在学术上还是在生活中，钱老师对我的关怀和照顾都是无微不至的。特别在课题研究中，钱老师从理论和实践两方面，都给予我大量的、极其有益的建议和指导。

她严谨的治学态度和对人生的孜孜不倦的追求，深深的感染着我，让我在以后的道路中，以一个积极的精神面貌去面对我的学业和人生。在此，我谨向她表达我深深的谢意！

感谢李彧晟讲师、李洪涛博士、顾陈博士、张劲东博士、王克让博士、柏磊博士在课题研究期间给予我的帮助，他们严谨缜密的治学态度、扎实的理论基础给我留下了深刻的印象。

感谢严璐、贺亚鹏、庄姗娜、郭俊芳、孟杲杲、郑重智、陶耕对我的热心帮助，是他们的帮助加速了课题研究的进程。

此外，感谢学院提供本人良好的学习环境与科研条件。感谢同窗好友们以及室友的陪伴、关心与帮助。感谢家人对我长久以来的支持和鼓励。

参考文献

[1] 张太怡等. 基于JPEG国际标准的图像压缩方法的研究[[J]. 重庆大学学报. 1994（9）：48~53

[2] 叶轻舟，林挺钊. 基于FPGA的JPEG静态图像压缩实现[J]. 福建工程学院学报. 2005 （3）：21~25

[3] 周海斌. 静态时序分析在高速FPGA设计中的应用.电子工程师. 2005（31）：11

[4] 景峰，崔春姬. 静态图像压缩一从JPEG到JPEG-2000.长春理工大学学报. 2002（25）：13

[5] 汪宇飞. JPEG高速编码芯片的设计及其性能优化. 西北工业大学硕士论文. 2006

[6] Bob Zeidman 著，赵宏图译. 基于FPGA&CPLD的数字IC设计方法. 北京：北京航空航天大学出版社，2004

[7] J. Bhasker 著，徐振林译. VERILOG HDL 硬件描述语言. 北京：机械工业出版社，2000

[8] 刘秋云. Verilog HDL 设计实践与指导. 北京：机械工业出版社，2005

[9] 袁俊泉，孙敏琪，曹瑞. Verilog HDL 数字系统设计及其应用. 西安： 西安电子科技大学出版社，2002

[10] 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，2003:1~2

[11] 刘峰. 视频图像编码技术及国际标准.. 北京：北京邮电大学出版社，2005

[12] 吴继华. 设计与验证-Verilog HDL. 北京：人民邮电出版社，2006

[13] 孙即祥. 图像压缩与投影重建. 北京：科学出版社，2005: 131~160

[14] 姚庆栋，毕厚杰. 图像编码基础. 北京：清华大学出版社，2004: 16~17

[15] 小野定康，铃木纯司. JPEG/MPEG2.. 北京：科学出版社，2004: 62~65

[16] Uwe Meyer-Baese 著，刘凌翻译. 数字信号处理的FPGA实现. 北京：清华大学出版社，2006

[17] 求是科技. CPLD/FPGA应用开发技术与工程实践. 人民邮电出版社，2005:1~2

[18] [美]Michael D. Ciletti，张雅绮，李锵等译. VerilogHDL 高级数字设计. 电子工业出版社，2007

[19] 简弘伦. 精通 Verilog HDL：IC设计核心技术实例详解. 电子工业出版社，2007

[20] 刘东. 用 VHDL 设计实现 JPEG（基本系统）硬件编码器. 西南交通大学硕士学位论文. 2003

[21] 王金明. VerilogHDL 程序设计教程. 第 1 版. 人民邮电出版社，2004

[22] [美]Michael D. Ciletti 著，张雅绮，李锵等译. Verilog HDL 高级数字设计. 北京：电了工业出版社，2005

[23] 尹伟. 基于 FPGA 的 JPEG 编解码芯片设计. 大连理工大学硕士论文. 2004

[24] 胡栋 静止图像编码的基本方法与国际标准. 北京邮电大学出版社，2003:36~40

[25] 褚振勇，翁木云. FPGA 设计与应用. 西安电子科技大学出版社，2002:12~18

[26] 巴斯克尔，孙海平. Verilog HDL 综合实用教程. 清华大学出版社，2004

[27] 洪志良. JPEG 静止图像压缩解压缩标准的硬件实现及其改进算法的研究. 复旦大学博士论文，2002

[28] J.Bhasker（美国）. Verilog HDL 综合实用教程[M]. 清华大学出版社，2004

[29] 夏宇闻. 复杂数字电路与系统的 Verilog HDL 设计技术. 北京：北京航空航天大学出版社，1998

[30] CCITT Rec.T.81（1992 E）104~105.

[31] Agostini L. V , Silva I. S., Bampi S. Pipelined fast 2D DCT architecture for JPEG image compression. Integrated Circuits and Systems Design. 2001（14）：226~231

[32] JPEG Committee Draft of the ISO/IEC JTCL COM VIII-R.24

[33] W.D.Pennebaker，J.Lmitchell. JPEG Still image Compression Standard.New York：Van Nostrand Reinhold，1992

[34] The IEEE Verilog 1364-2000 Standard，2000

[35] Mohamed El-Sharkawy，Waleed Eshma. A Fast 8 X 8 Pruned DCT Algorithm. Digital Signal Processing. 1996（6）：145-154.

[36] The Verilog Language Reference Manual，1995

[37] LEGER （A.），OMACHI （T.），WALLACE （G.K.）. JPEG Still Picture Compression Algorithm. Optical Engineering. 1991（7）：947~954

[38] G. K. Wallace. The JPEG still picture compression standard. Commun. ACM. 1991（34）：30-44

[39] Agostini L. V Silva I. S.，Bampi S. Pipelined Entropy Coders for JPEG image compression. Integrated Circuit sand System Design，2001

[40] Kachouri，R.，Abid，M.，Amar，C.B. Design of DCT/spl Lbar/2D toward FPGA[J]. Control，Communications and Signal Processing. 2004：713~716

[41] O. Cadenas, Cz Megson, T. Plaks. Accelerating JPEG compression with a dynamically reconfigurable systolic array [J]. International Conference on Parallel and Distributed Processing Techniques and Applications. 2006（4）：3023~3026

[42] The IEEE Verilog 1364-2000 Standard. 2000

[43] Pennebaker and Mitchell. JPEG Stidd Image Data Compression Standard. Copyright by Van Nostrand Reinhold，1993

[44] A. Skodras，T. Ebrahimi. The JPEG 2000 Still Image Coding： An Overview. Computer and Education. 2000（2）：20~22

[45] D. H. Ballard. Benchmarking and hardware implementation of . JPEG-LS. Artificial Intelligence. 1984（3）：235~276

[46] 王金明. 数字系统设计与 Verilog HDL. 第 2 版. 北京：电子工业出版社，2005

[47] 吴继华. 设计与验证-Verilog HDL. 北京：人民邮电出版社，2006

[48] 任爱锋，初秀琴，等. 基于 FPGA 的嵌入式系统设计. 西安：西安电子科技大学出版社，2004

[49] 薛小刚，葛毅敏. Xilinx ISE 9.X FPGA/CPLD 设计指南. 北京：人民邮电出版社，2007

[50] 卓兴旺. 基于 Verilog HDL 的数字系统应用设计. 第 2 版. 北京：国防工业出版社，2007

基于FPGA的JPEG解码算法的研究与实现

作者：张艳
学位授予单位：南京理工大学

相似文献(1条)

1.学位论文 陈科研 基于Nios Ⅱ的双核处理器系统研发 2009

可编程片上系统（SOPC）是基于可编程逻辑器件的可重构片上系统，作为片上系统（SOC）和现场可编程门阵列（FPGA）相结合的一项综合技术，它利用软核处理器在FPGA开发板上方便地实现针对具体应用的系统设计方案，能够获得系统设计的可编程性和低风险性，应用的灵活性及价格优势，被称为“半导体产业的未来”。

本文基于系统可重配置的思想，及面向用户，面向应用的SOPC技术设计思想，研究了NiosⅡ软核处理器的结构、性能和总线规范；构建了单核NiosⅡ处理器系统，并在FPGA芯片上实现，通过运算密集型的JPEG解码任务验证了系统的具体应用和正确性；提出基于NiosⅡ软核处理器的嵌入式多核系统设计思想，设计了双核NiosⅡ处理器系统，实现了双核系统对JPEG解码的并行处理；开发了JPEG解码中YUV到RGB转换的自定义功能模块，并成功将此模块集成入NiosⅡ处理器系统中，实现了浮点运算和乘法运算硬件加速处理，提高了系统的整体性能。对系统功耗和性能进行了优化，利用逻辑映射重构，存储器优化和存储器平衡等系统优化手段，使双核处理器系统功耗降低55mW，减少幅度达到总体功耗的7．03%。通过将自定义功能模块集成入系统中，减小处理器运算开销，增大系统处理并行度，单核处理器系统和双核处理器系统的性能分别被提高了约17%和15%。本文应用与NiosⅡ相关的集成开发平台和FPGA开发板，加快了SOPC系统的设计与验证环节的开发速度，对于嵌入式系统，特别是多处理器系统的开发和应用，具有广泛的价值和积极的意义。