分类号：TN492　　学校代码：10363
密　级：公开　　学　　号：2210330156

安徽工程大学
Anhui Polytechnic University

# 硕士学位论文

**题目**　MJPEG视频编解码的IP核优化设计方法

论文作者　张星
指导教师　张肖强
学科（专业）　电子信息
研究方向　系统集成技术及应用

论文提交日期：2024 年 5 月 31 日

分类号： TN492 单位代码：10363

密　级：公开 学　号：2210330156

# 题　目 MJPEG 视频编解码的 IP 核优化设计方法

# 题　目 Optimization Design Method for Video Encoding and Decoding IP Core Based on MJPEG

学生姓名：_____张星_____

指导教师：_____张肖强_____

专　　业：_____电子信息_____

研究方向：_____系统集成技术及应用_____

论文答辩日期：2024 年 5 月 22 日

# 安徽工程大学学位论文原创性声明

本人郑重声明：我恪守学术道德，崇尚严谨学风。所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已明确注明和引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品及成果的内容。论文为本人亲自撰写，我对所写的内容负责，并完全意识到本声明的法律后果由本人承担。

学位论文作者签名： 

日期：2024 年 5 月 31 日

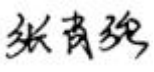
# 安徽工程大学学位论文版权使用授权书

学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅或借阅。本人授权安徽工程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，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密 □，在 __ 年解密后适用本版权书。

本学位论文属于

不保密 ☑。

学位论文作者签名：  　　指导教师签名： 

日期： 2024 年 5 月 31 日　　日期： 2024 年 5 月 31 日

# MJPEG 视频编解码的 IP 核优化设计方法

## 摘 要

随着数字信息技术的不断发展，图像信息在人们日常信息交流之中发挥着越来越重要的作用，由于内存和带宽有限，视频图像的压缩也显得尤为关键。M-JPEG（Motion JPEG）作为一种视频压缩格式，以每帧图像作为独立的JPEG图像来压缩视频，因其简单的编解码方式、快速的处理速度和高压缩比而备受青睐。通过M-JPEG压缩可以大幅度减少存储空间和网络传输的带宽资源。

本文以FPGA为开发平台，对基于M-JPEG视频编解码的IP核做了相关优化改进，主要的研究内容如下：

（1）对常规的二维离散余弦变换（Two-dimensional Discrete Cosine Transform, 2D-DCT）进行了改进，在有符号离散余弦变换（Signed Discrete Cosine Transform, SDCT）的基础上进行部分取零操作，将其进行正交化调整得到新的DCT变换矩阵，并在2D-DCT变换模块的改进设计的基础上采用了乒乓操作、流水线、chen氏算法等技术，新的DCT变换矩阵在1D-DCT阶段减少了1/3的乘法器的调用，从而进一步降低了算术复杂度，在提高运算速度的同时减少了资源的消耗。

（2）在量化阶段对量化表做了近似的处理，并且将量化模块和Z扫描模块集成到一起，减少对缓存器的访问次数，节省了时间。通过移位操作完成量化的步骤，避免在量化阶段对乘法器进行调用，进一步减少存储器的占用、提升了处理速度。

（3）在Vivado中利用测试平台（Testbench）对M-JPEG视频编

解码的IP核进行仿真，在74.25M的系统时钟下对分辨率为1280×720的彩色图像数据文件进行仿真处理，其编码帧率可以达到66.17帧每秒，解码帧率达到62.76帧每秒。最后基于Xilinx（xc7a100tfgg484）FPGA平台对视频编解码系统进行了实际电路的性能测试。最终的实验表明M-JPEG视频编解码IP核能够实时处理1280×720的视频图像，且在与其他文献的对比中有着良好的编解码性能，在速度和资源消耗的平衡上有着更好的表现。

本文在FPGA平台上实现了基于M-JPEG算法的视频编解码，其IP核具有复杂度低、资源消耗少、性能稳定等优点。在降低成本、易于实现方面有着现实意义，为其他图像编解码算法的硬件实现提供了参考借鉴。

**关键词：**M-JPEG，视频编解码，FPGA，DCT，量化

# Optimization Design Method for IP Core of MJPEG Video Encoding and Decoding

## ABSTRACT

With the continuous development of digital information technology, image information plays an increasingly important role in people's daily information communication. Due to limited memory and bandwidth, video image compression is also particularly crucial. M-JPEG (Motion JPEG) is a video compression format that compresses videos using each frame of image as an independent JPEG image. It is highly favored due to its simple encoding and decoding method, fast processing speed, and high compression ratio. M-JPEG compression can significantly reduce storage space and network bandwidth resources.

This article uses FPGA as the development platform to optimize and improve the IP core based on M-JPEG video encoding and decoding. The main research content is as follows:

(1) The conventional Two dimensional Discrete Cosine Transform (2D-DCT) has been improved by performing partial zeroing operations on the basis of the Signed Discrete Cosine Transform (SDCT), orthogonalizing it to obtain a new DCT transformation matrix. Based on the improved design of the 2D-DCT transformation module, techniques such as ping pong operation, pipeline, and Chen's algorithm have been adopted. The new DCT transformation matrix reduces the number of multiplier calls by one-third in the 1D-DCT stage, further reducing arithmetic complexity and reducing resource consumption while increasing speed.

(2) In the quantization stage, the quantization table was approximated and the quantization module and Z-scan module were integrated together

to reduce the number of accesses to the buffer and save time. By performing a shift operation to complete the quantization step, it avoids calling the multiplier during the quantization phase, further reducing memory usage and improving processing speed.

(3) Simulate the IP core of M-JPEG video encoding and decoding using the Testbench platform in Vivado. Simulate and process color image data files with a resolution of 1280×720 under a 74.25M system clock. The encoding frame rate can reach 66.17 frames per second and the decoding frame rate can reach 62.76 frames per second. Finally, actual circuit performance tests were conducted on the video encoding and decoding system based on Xilinx (xc7a100tfgg484) FPGA platform. The final experiment showed that the M-JPEG video encoding and decoding IP core can process 1280×720 video images in real-time, and has good encoding and decoding performance compared to other literature, with better performance in balancing speed and resource consumption.

This article implements video encoding and decoding based on M-JPEG algorithm on FPGA platform. Its IP core has advantages such as low complexity, low resource consumption, and stable performance. It has practical significance in reducing costs and facilitating implementation, providing reference for the hardware implementation of other image encoding and decoding algorithms.

**KEY WORDS:** M-JPEG, video encoding and decoding, FPGA, DCT, quantization

# 目录

**摘 要** ..... **I**

**ABSTRACT** ..... **III**

**第 1 章 绪论** ..... **1**

1.1 研究背景及意义 ..... 1

1.2 JPEG 编解码器国内外研究现状 ..... 2

1.3 研究内容 ..... 4

1.4 论文结构安排 ..... 5

**第 2 章 M-JPEG 压缩原理** ..... **6**

2.1 JPEG 压缩基本原理和流程 ..... 6

2.2 颜色空间转换 ..... 7

2.3 降采样与升采样 ..... 7

2.4 二维离散余弦变换及反变换 ..... 8

2.5 量化及反量化 ..... 9

2.6 Z 扫描和反 Z 字扫描 ..... 10

2.7 熵编码与熵解码 ..... 11

2.7.1 DC 系数的中间格式 ..... 11

2.7.2 AC 系数的行程编码 ..... 12

2.7.3 AC 系数的中间格式 ..... 12

2.7.4 Huffman 编码及解码 ..... 13

2.8 JPEG 文件格式 ..... 14

2.9 本章小结 ..... 15

**第 3 章 图像编解码模块的研究与设计** ..... **16**

3.1 JPEG 编码的基本架构 ..... 16

3.2 图像预处理模块 ..... 16

3.2.1 色彩空间转换模块 ..... 17

3.2.2 下采样模块 ..... 18

3.3 JPEG 编码各模块设计 ..... 19
3.3.1 二维离散余弦变换模块 ..... 19
3.3.2 量化和 Z 扫描模块 ..... 25
3.3.3 熵编码模块 ..... 27
3.3.4 码流组装模块 ..... 31
3.4 JPEG 解码的基本架构 ..... 31
3.5 JPEG 解码各模块设计 ..... 32
3.5.1 Huffman 解码模块 ..... 32
3.5.2 2D-IDCT 模块 ..... 33
3.5.3 其他模块 ..... 34
3.6 本章小结 ..... 34
**第 4 章 仿真与系统验证 ..... 35**
4.1 M-JPEG 编解码 IP 核的功能仿真 ..... 35
4.1.1 搭建仿真测试平台 ..... 35
4.1.2 仿真测试结果 ..... 36
4.2 基于 FPGA 的验证平台设计 ..... 38
4.3 验证所需模块的设计 ..... 38
4.3.1 视频采集模块设计 ..... 39
4.3.2 DDR3 存储模块 ..... 42
4.3.3 以太网模块 ..... 45
4.3.4 HDMI 接口模块 ..... 48
4.4 静态及动态图像的板载验证 ..... 50
4.4.1 实验结果 ..... 51
4.5 本章小结 ..... 52
**第 5 章 总结与展望 ..... 53**
5.1 全文总结 ..... 53
5.2 未来工作展望 ..... 53
**参考文献 ..... 55**
**攻读硕士期间取得学术成果 ..... 59**

# 第 1 章 绪论

## 1.1 研究背景及意义

随着计算机和信息科学技术的不断发展，视频图像在生活中得到了越来越多的应用[1][2]，视频应用不再局限于娱乐场景，而是向工业制造、安防领域、生产办公等场景拓展。在办公场景中，远程会议作为新型视频应用是远程办公的重要模式；在工业、安防等领域，机器视觉和视频监控对视频形式提出了新的要求。未来随着产业整合泛化和元宇宙布局趋势加强，人们需求和视频类型将会更趋多元[3]。而这些应用的基础都需要对图像进行网络传输[4]。如果一张图片不经任何处理直接传输或者存储，本身的数据信息量是比较大的[5]。以一张分辨率大小为$1280\times720$的真彩图片为例，它的图片像素的数据大小达到了$1280\times720\times24\approx2.76MByte$，如果以60帧率的视频显示，那么一秒的数据量为$2.76\times60=165.6MByte$。保存一段视频就需要更大的存储空间，同时在传输这些视频图像数据的时候还会占据非常多的带宽资源，影响传输速度[6]。只靠扩大内存和带宽是不现实的，也不能从根本上解决问题，因此，对视频图像的数据压缩显得尤为重要[7]。

JPEG（Joint Photographic Experts Group）由联合图像专家组与国际标准化组织共同制定的静态图像编码和压缩标准，以离散余弦变换（DCT）为核心编码技术，JPEG为静态图像的压缩和解压缩流程及数据存储提供了详细的规定[8]。它以其卓越的压缩比而著称，同时保持了接近原始的图像质量，展现了其在高质量数字图像管理方面的高效性和有效性[9]，现在广泛应用在互联网、数字摄影、媒体传输、打印扫描等多方面领域，JPEG是目前最常用的静态图像压缩存储方式，网页上绝大多数图片都采用的JPEG压缩标准[10]，虽然在JPEG的基础上进一步发展出了如JPEG2000、JPEG-LS等新一代的无损压缩标准，它们有着无损的压缩模式，更清晰的图像质量，甚至有着更高的压缩比，在一些低带宽环境下，或者医学、卫星图像等要求图像质量非常高的特殊场合中发挥着重要作[12]。但是在绝大多数普通场景中JPEG已经能满足人们的需求，而且JPEG压缩算法相对简单、低成本、易于实现和处理的优点也是别的算法无法替代的，

因此JPEG在静态图像压缩中仍然有着主导地位，不仅如此，许多视频编解码协议也会使用JPEG格式来压缩视频帧中的关键帧，如MPEG[13]，H.26X[14]系列等，但是MPEG由于专利问题使得一些开源或低成本项目难以采用，H.26X系列在编解码过程中需要更多的计算资源，因此需要更强大的处理器。而JPEG有着自己的视频标准M-JPEG（Motion-JPEG），它是一种基于JPEG的视频压缩技术，它通过将视频分割成一系列独立的帧，并对每一帧进行单独的JPEG压缩，从而实现视频压缩[15]，特别适用于视频监控以及视频编辑等应用场景。其每帧独立的特性使得M-JPEG非常适合于那些需要高质量图像和快速、随机访问帧的场合。此外，M-JPEG还具有系统实现简单方便，压缩后图像恢复质量好，且没有码率上限等优点[16]。

基于M-JPEG实现编解码的方法主要有两种，一种是通过硬件平台的软件上实现功能，如今比较常见的是在多核平台上设计并行框架对编解码进行处理，结合了多个CPU（Central Processing Unit）和GPU（Graphics Processing Unit）加强了对数据的处理能力[17]。另一种就是在集成电路上实现，ASIC（Application-Specific Integrated Circuit）为特定应用或任务量身定制的集成电路[18]。在其专门的应用领域内通常能提供最优的性能和能效，但缺乏灵活性，一旦制造出来，其功能就固定了，不能更改或升级。FPGA（Field-Programmable Gate Array）是一种可以在现场配置和重新配置的集成电路。FPGA提供了比ASIC更高的灵活性，允许快速原型设计和测试，避免了高昂的ASIC设计和制造成本，凭借自身并行处理架构和丰富的资源在通信、视频图像、数字信号等领域都有着广泛的运用[19]。M-JPEG编解码硬件上的实现方式相对软件实现方式相比，它可以通过硬件并行计算提供高效的数据处理和计算能力，在功耗和处理速度中有着出色表现[20]。

因此，基于FPGA对M-JPEG视频编解码IP核进行研究设计，对视频图像处理的其他压缩算法在FPGA实现以及IP核的设计有着一定的参考借鉴意义，在视频图像存储和传输方面具有重要的研究意义和应用意义。

## 1.2 JPEG 编解码器国内外研究现状

数字图像压缩技术已经取得很大成绩，广泛应用于医学、数码相机等数字图像存储和通讯领域。随着FPGA硬件平台的不断发展和算法优化的不断深入，

国内外的研究者们基于FPGA硬件平台设计了高效的JPEG编解码器，以满足不同应用场景下的需求。这些研究有的着重于优化算法、有的从架构上进行改进，以此来提高编解码速度、降低功耗，实现硬件资源的高效利用。

在对JPEG编解码的硬件研究中，有研究者对编解码中局部算法进行改进优化，如江西理工大学的研究者为了解决图像边界效应采用了改进的离散余弦变换(MDCT)，结果表明MDCT算法替代JPEG标准中的DCT算法后，图像的分辨率越高，算法的时间成本提升越高，且细节越多的图像得到的压缩重建效果提升更高[21]；

还有武汉大学的研究者是针对量化阶段进行了优化设计，研究一种基于改进量化表的JPEG图像压缩算法(JPEG-HVS)，利用人眼亮度对比度敏感函数(CSF)生成一种新的量化表来代替传统JPEG标准推荐的亮度量化表，结果显示JPEG-HVS实现的压缩比要比JPEG实现的压缩比平均高出53.56%，比JPEG区域法平均高出18.75%，且编解码所需时间要明显少于JPEG[22]。

YW.Chang等人提出了一种新的JPEG霍夫曼编解码算法，将交流霍夫曼表划分为四个模块。根据这个划分，开发了一种直接映射技术来解码霍夫曼码字而不需要任何搜索或匹配操作，该技术的算法可以节省几个控制电路，并且硬件实现简单[23]。

研究者们不仅在编解码的局部算法中有着改进，还有的是在架构上进行创新设计。南京航空航天大学的研究者提出并设计了一种基于颜色分量的多流水线架构，解决了常见多流水线架构中DC系数无法在单条流水线中编码的问题；并对Huffman编码过程设计了流水级拓展的快速编码方式，解决了在复杂图像的编码过程中Huffman编码周期过长的问题[24]。

韦洛尔科技大学学者研究介绍了一个完全流水线化的无乘法器架构的8×8二维DCT/IDCT的VLSI实现，通过引入流水线和基于Loeffler因子分解的BinDCT无乘法器架构来说明对现有JPEG有损压缩设计的改进，使其硬件设计变得简单[25]。

俞世超等人提出了一种可扩展的多流水线JPEG编码体系结构[26]。RTL级别描述的编码器可以在性能、占用的资源和功耗方面灵活定制。根据仿真和FPGA验证，当扩展到2条流水线时，编码器能够以每秒47.51帧的速度编码1920×1080大小的RGB图像。

JPEG编解码器的研究不仅局限于学术研究，也有国内外厂商对其进行研究

设计，国内珠海一家公司推出的MJPEG编解码IP核在Xilinx Zynq7020上在133MHz的时钟频率条件下单核就能实现1080p@30fps的速率，双核可以实现1080p@60fps的应用场景，并且在Zynq7020上只占用了25%的逻辑资源；国外比利时的intopix公司的JPEG2000编解码器的高清IP核，对1920×1280的图像它的帧率最高能达到120，速度最多为1Gbps/s。

综上所述，在JPEG编解码的硬件实现上国内外有了比较多的研究与应用，他们在算法优化、硬件架构设计等方面取得了一系列成果，实现了更快的编解码速度和更高的图像质量。

## 1.3 研究内容

本文的主要研究内容是以FPGA为平台，基于M-JPEG算法利用Verilog HDL语言设计视频编解码的IP核。在JPEG算法的基础上进行研究分析，总结其中的不足之处，并对编解码的局部电路进行优化设计，采用Verilog硬件描述语言按照Top-Down（自顶向下）的方法对M-JPEG基本模式硬件编解码器的各主要模块进行设计仿真和实现。最终搭建M-JPEG视频编解码系统的硬件平台并对其进行实际验证，其主要的研究内容如图1-1所示。

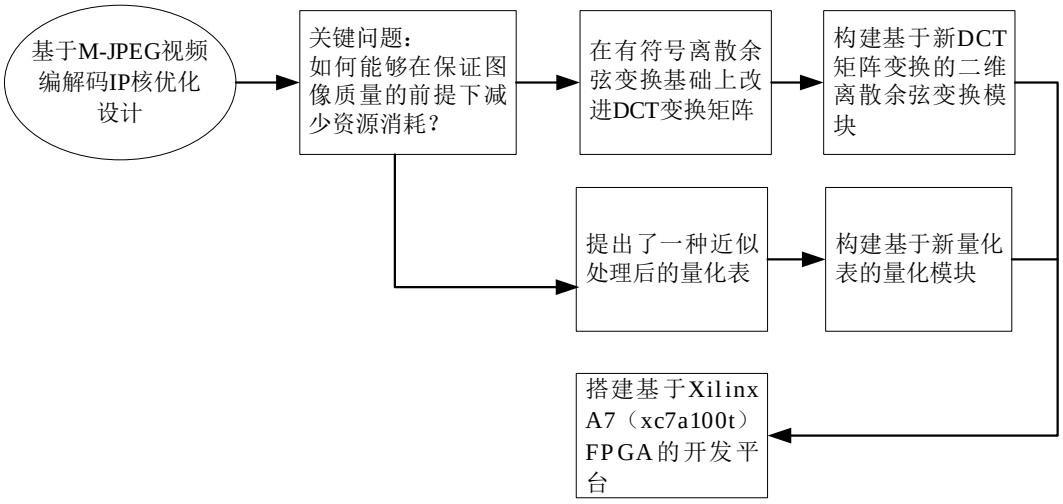



图1-1 研究内容框架

（1）在保证图像质量的前提下加快编码的速度，减少资源消耗，在SDCT的基础上进行部分取零的操作，并通过行列分解以及对称性来化简公式，降低2D-DCT的运算复杂度，使用新的变换矩阵可以在1D-DCT阶段减少了1/3乘法器和加法器的调用，减少了逻辑门的数量，节省了资源。

（2）将量化表和Z字形扫描模块集成在一起减少对缓存器的访问次数，并

对量化阶段的量化表进行特殊的近似处理从而得到新的量化表，新的量化表中的步长可以通过移位操作完成量化操作，避免了在量化阶段对乘法器的调用，减少了资源的消耗。

（3）对编解码的各模块进行仿真测试，验证了M-JPEG的编码和解码IP核的正确性，测试其编解码的速度，在74.25M的系统时钟下对于1280×720的彩色图像其编码帧率可以达到66.17帧每秒，解码帧率达到62.76帧每秒，最终在FPGA平台上搭建M-JPEG视频编解码系统并进行了实际的板载验证并将实验结果与其他文献做出对比分析。

## 1.4 论文结构安排

本文以视频图像压缩为研究对象，以FPGA为硬件平台，基于M-JPEG算法设计、实现、优化视频编解码IP核，在组织结构上分五个章节展开，内容如下：

（1）第一章为绪论，介绍了视频编解码的研究背景和意义，总结分析了国内外对JPEG编解码在硬件上实现的研究现状，最后阐述了论文的研究内容和结构安排。

（2）第二章介绍了M-JPEG的整个流程和基本原理，对编码各个模块的功能原理进行详细阐述，并简略介绍了相对应的逆过程，为后续的设计实现提供了理论基础。

（3）第三章对编解码的基本框架进行介绍，对编码各模块的硬件进行设计，包括色彩空间转换、下采样、2D-DCT、量化和扫描、熵编码和最后的码流组装模块，并在2D-DCT、量化模块上进行了相关改进和优化。解码由于是编码的一个逆过程，原理框架基本一致，所以只对比较重要的二维逆离散余弦变换（Two-dimensional Inverse Discrete Cosine Transform, 2D-IDCT）和熵解码进行设计，而其他的模块只做了相关文字介绍。

（4）第四章先是对M-JPEG的编解码IP核搭建仿真测试平台，初步证明了IP核的功能正确性，接着构建了验证视频编解码IP的整体硬件框架，介绍了平台所需要的其他模块的作用及设计，最后利用搭建的FPGA平台对M-JPEG编解码IP核进行静态图像和动态视频的验证，并将IP的性能与其他文献的编解码器进行对比。

（5）第五章是全文的总结以及对未来工作的展望。

# 第 2 章 M-JPEG 压缩原理

MJPEG（Motion JPEG）是一种视频压缩格式，它使用JPEG压缩算法单独压缩。相比于其他视频压缩格式，MJPEG没有利用帧间关系，而是把视频分解为一系列的JPEG图像对每一帧单独编码。因此本章着重介绍了JPEG的压缩相关模块和流程。

## 2.1 JPEG 压缩基本原理和流程

JPEG是一种广泛应用于图像压缩的标准，其核心技术采用了离散余弦变换（DCT）的压缩方法。JPEG压缩丢失了部分原始图像的数据信息，但这个丢去的部分是人视觉中不容易察觉到的高频信息部分[27]，这样一来需要处理的数据信息就大大减少了。图2-1是JPEG编解码的基本流程。

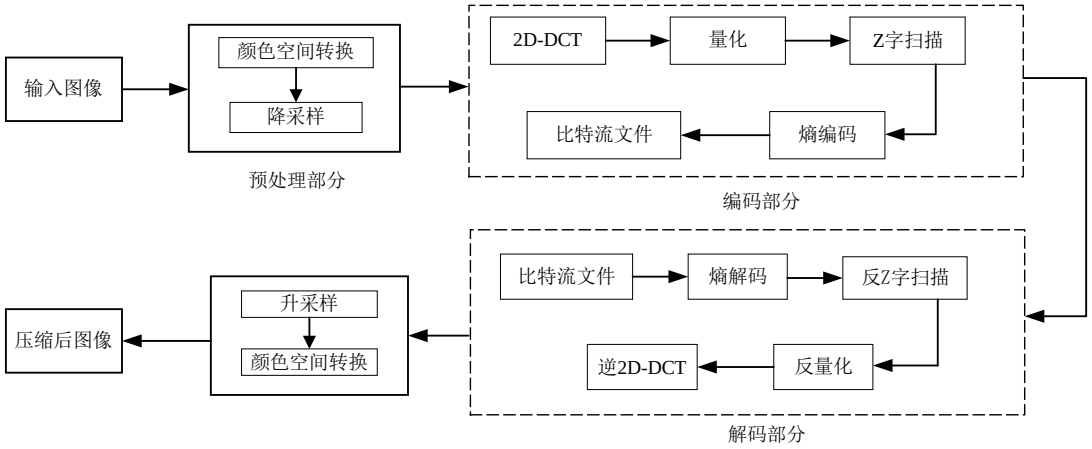



图2-1 JPEG编解码流程图

在JPEG编码中，首先对图像进行颜色空间转换并通过降采样来减少图像的数据大小；之后将整个图片切割成图像块；对这些图像块进行二维离散余弦变换（2D-DCT），使图像从空间域变换到频率域[28]，这样能够更好的表示图像的频谱特性；下一步的量化是针对DCT系数进行的数据又一次有损压缩，Z扫描则是对系数进行排序重组；最后应用熵编码来减少数据冗余。

而在JPEG解码中，只需要将编码步骤反过来执行一遍便得到最终还原出来的图像。在此就不再做过多赘述。

## 2.2 颜色空间转换

颜色空间转换是将图像从一个颜色表示方式转换为另一种的过程，在日常生活中人们见的最多的是RGB色彩模型。而在JPEG压缩中，为了提高压缩效率从而选择YCbCr色彩空间模型，可以通过转换公式将RGB的三个通道转换为YCbCr的三个通道[29]，具体矩阵变换如公式(2-1)所示。

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} 0.299 & 0.578 & 0.114 \\ 0.500 & -0.4187 & -0.0813 \\ -0.1687 & -0.3313 & 0.500 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{2-1}$$

在JPEG压缩中，通常会将上述结果进行一个128的偏移处理，将YCbCr的数值限定在[-128，127]这个区间，这个处理的目的是将原始数据的中心平移到零点，以便更好的适应DCT的性质。同样的，在解码操作时需要将这个偏移还原，以得到原始的图像数据。

图2-2为RGB原图和它的Y、Cb、Cr分量所表示的图像，从图中可以当RGB被转换成YCbCr色彩空间模型之后，其中Y分量包含了大多数的图像信息，决定了图像的明暗程度，而Cb和Cr分量表示色度信息，但是对人眼的影响程度相对较小。

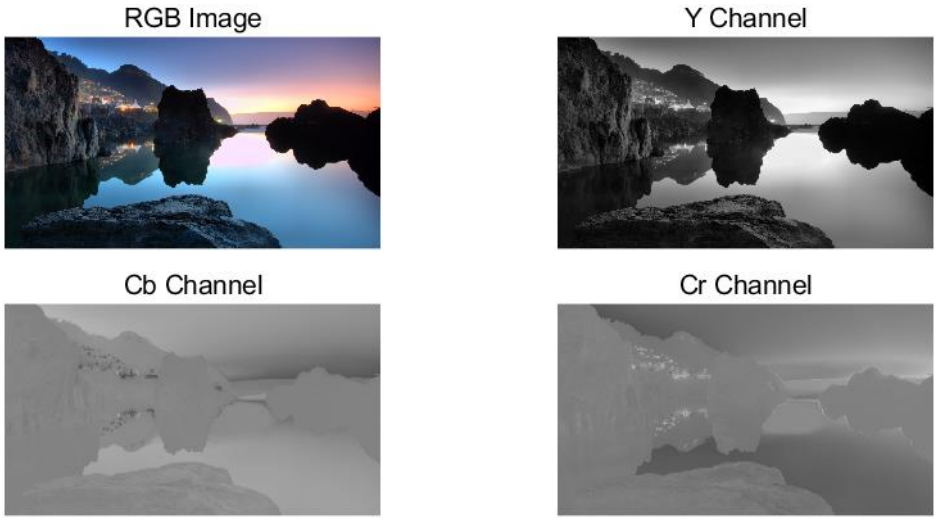



图2-2 Y、Cb、Cr分量的图像

## 2.3 降采样与升采样

下采样是JPEG中一种常用的压缩技术，由于JPEG图像通常使用YCbCr颜色空间模型，其中包含亮度（Y）分量和两个颜色分量（Cb和Cr），下采样利用人

眼对亮度和色度不同敏感这一特点，通过减少色度分量的采样率减少图像文件大小。

JFIF标准下采样模式一般包括4:4:4，4:2:2，4:2:0三种，分别对应a、b、c，具体如图2-3所示。

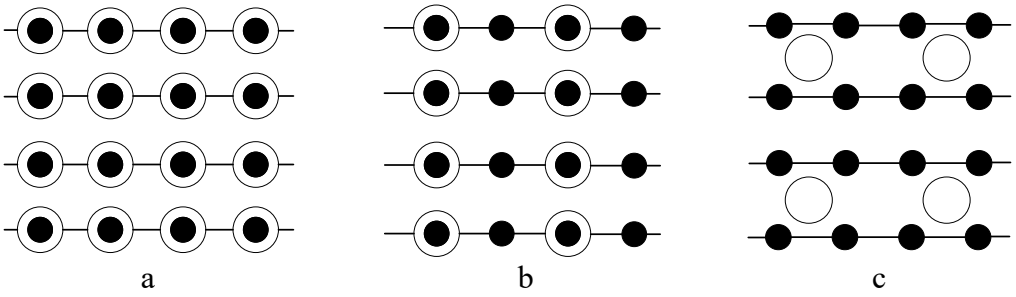



图 2-3 三种常见下采样示意图

黑点表示该像素点的Y，空心圆表示该像素点的UV。a图为4:4:4采样模式，即无下采样，Y、Cb、Cr分量都以全分辨率表示，没有下采样，这是最高质量的模式，但文件大小相对较大。b图为4:2:2采样模式，Y分量的分辨率保持不变，而Cb和Cr分量的水平方向的分辨率减半。即水平方向上每两个相邻的色度样本只保留一个。c图为4:2:0采样模式，Y分量分辨率不变，而Cb和Cr分量的水平方向和垂直方向的分辨率均减半，也就是说4个Y分量共用一组UV。

在解码的升采样过程中，需要把YCbCr模型转换回RGB模型，为了提供降采样时舍去的Cb、Cr分量，直接把剩余的Cr00、Cb00等同于舍去的Cb和Cr分量。

## 2.4 二维离散余弦变换及反变换

二维离散余弦变换（2D-DCT）是在图像和信号处理中常用的技术[30]，特别是在图像和视频压缩领域中广泛应用，它是傅里叶变换相关的一种变换，能够将一个二维的空间域图像转换为一个二维的频域表示，从而实现信号压缩和信息隐藏等应用。

2D-DCT是从一维DCT推广而来的，其基本思想是将输入的二维数据块分别沿行和列进行一维DCT变换，然后得到每个数据块在频域上的表示。在二维离散余弦变换中，对于一个N×N的输入，输出也是N×N的系数矩阵。其基本思想是将图像分解成一系列基函数，这些基函数是余弦函数的二维变体。通过对图像进行这种变换，信号的能量主要集中在少量的低频系数上。二维离散余弦变换的公式如(2-2)所示。

$$F(u,v)=\frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad (2\text{-}2)$$

其中，$F(u,v)$是对$f(x,y)$变换后的频域表示；$u$、$v$分别表示频域的水平和垂直位置；$f(x,y)$是输入图像二维数据块；$x$、$y$分别表示数据块的水平垂直位置；$C(u)$和$C(v)$是权重系数，定义为：

$$C(i)=\begin{cases}\sqrt{\frac{2}{N}}, i\neq 0\\ \sqrt{\frac{1}{N}}, i=0\end{cases} \quad (2\text{-}3)$$

逆二维离散余弦变换（IDCT）是将二维离散余弦变换（2D DCT）的系数转换回原始信号的过程。其公式和二维离散余弦变换基本一样，只是其中的系数有所不同，在此不再单独列出公式。

## 2.5 量化及反量化

经过离散余弦变换之后图像信息被分成了两部分：左上角低频部分（整体结构）和右下角高频部分（细节纹理）。一般来说，人眼对图像的高频信息不敏感,对低频信息比较敏感[31]。因此在量化的过程中舍弃了一些高频细节部分，减少了图像数据大小，从而实现了压缩了图片。

在JPEG编码中，对分成$8\times 8$的图像块进行二维离散余弦变换，得到的系数称为DCT系数。随后这些DCT系数进入量化阶段，即通过除以一个固定的量化矩阵中的相应元素，再将系数四舍五入近似为整数值，而反量化就是这个步骤的逆过程，把熵解码出的数据乘上量化矩阵中对应的元素，这个量化矩阵就是量化表。

JPEG标准使用了两个独立的量化表，一个用于Y分量，另一个用于Cb和Cr分量。这是因为在彩色图像中，亮度和色度有着各自对应的量化表，这样可以更好地适应人眼对不同颜色通道的感知特性。亮度和色度的量化表内容分别如公式(2-4)、(2-5)所示。

量化表中的值越大，表示对应的DCT系数经过量化后越不精准，所以越容易被舍弃。通过量化图像的数据被显著压缩，但同时也会引入失真。

$$Q_L = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{2-4}$$

$$Q_C = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix} \tag{2-5}$$

## 2.6 Z 扫描和反 Z 字扫描

为了提高熵编码的效率，需要对量化之后的64个系数的位置进行新一轮的排序，即Z字形扫描。它是以Z形的走势对64个矩阵系数排列成一个一维序列。由于量化表的数据分布特点，经过量化后的右下角数据有较多的0，而Z扫描可以有效地将这些0值集中在一起，这样可以使得接下来的熵编码更适合压缩图像数据，其扫描顺序如图2-4所示。

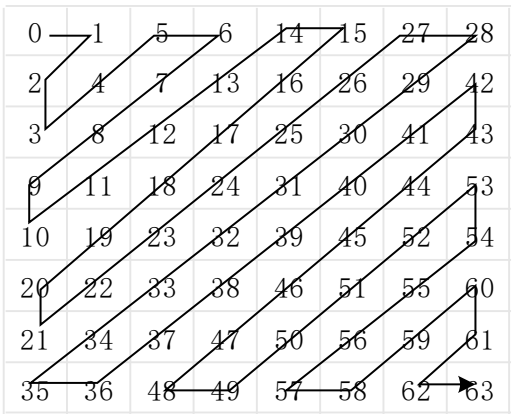



图 2-4 Z 扫描示意图

反Z字扫描则是将一维序列还原成8×8 的矩阵，使得原始的DCT系数在二维空间中的排列得以恢复。

## 2.7 熵编码与熵解码

熵编码是用来对数据进行压缩的一种方法，它能够在将数据表示为更紧凑的形式的同时保持数据的原始信息，利用输入数据的统计特性来减少数据的表示长度。熵编码的核心概念是信息熵，即表示随机变量的不确定性或信息量的度量。从而达到最佳的压缩效果。

在JPEG图像压缩中，用DC系数（直流系数）和AC系数（交流系数）两个概念来表示图像的频域信息。其中DC系数代表了DCT变换后信号的低频成分，即整体的亮度或能量。在图像中，它对应于图像的平均亮度或灰度偏移。DC系数一般用图像块进行DCT变换后的第一个系数来表示；对于DC系数采用差分脉冲编码，即用当前块的DC系数减去前一个同类型块的DC系数，这样可以减小表示的数值范围，提高编码效率；AC系数表示DCT变换后信号的高频成分，在图像中，AC系数通常对应于图像中的纹理、边缘和其他细微结构，通常用图像块进行DCT变换后的一系列系数来表示，由于量化后的AC系数存在大量连续的零，通常采用行程编码对其进行压缩。

在本文中，使用的是霍夫曼编码作为熵编码方式，而熵解码则是对已压缩数据进行解码的过程，其中包括了霍夫曼解码、行程解码和变长解码几部分。其目的是将压缩后的数据重新还原为之前的数据，以方便进行后续的处理或传输。

### 2.7.1 DC 系数的中间格式

在经过离散余弦变换（DCT）后，图像块中的第一个系数被称为DC系数，其余系数称AC系数。JPEG图像压缩中，DC系数通常采用差分脉冲编码来表示相邻图像块之间DC系数的变化。计算公式如下所示。

$$DC_{diff} = DC_{cur} - DC_{pre} \tag{2-6}$$

其中 $DC_{diff}$ 表示差分编码后的DC系数， $DC_{cur}$ 表示当前图像块的DC系数。$DC_{pre}$ 表示前一个图像块的DC系数。

为了进一步压缩数据大小，JPEG将数据分为多组并保存在表格中，也就是变长整数编码（VLI）。如表2-1所示。

表2-1 VLI表

| 数值 | 组 | 实际保存数值 |
|---|---|---|
| 0 | 0 | 0/无 |
| -1，1 | 1 | 0，1 |
| -3，-2，2，3 | 2 | 00，01，10，11 |
| -7，-6，-5，-4，4，5，6，7 | 3 | 000，…，111 |
| -15,…,-8,8,…,15 | 4 | 0000，…，1111 |
| -31,…,-16,16,…,31 | 5 | 00000，…，11111 |
| -63,…,-32,32,…63 | 6 | 000000，…，111111 |
| -127,…,-64,64,…127 | 7 | 0000000，…，1111111 |
| -255,…,-128,128,…255 | 8 | 00000000，…，11111111 |
| -511,…,-256,256,…511 | 9 | 000000000，…，111111111 |
| -1023，…，-512，512，…1023 | 10 | 0000000000，…，1111111111 |
| -2047，…，-1024，2014，…2047 | 11 | 00000000000，…，11111111111 |

比如 $DC_{diff}$ 等于7，经过查上表可知该数据在第3组，所以可以表示为（3）（7），该形式就称为DC系数的中间格式。

### 2.7.2 AC 系数的行程编码

行程编码（Run-Length Coding，RLC）是一种简单而有效的无损数据压缩技术。它利用连续出现的相同符号或数据值，用一个符号和一个计数值来表示，从而减少重复数据的存储空间。例如一组序列为：AAAABBCCCDAA，通过行程编码可以表示为：4A2B3C1D2A。

对于一些随机性较强的数据，行程编码可能并不会带来很大的压缩效果，但是在对一些连续且重复的数据非常有效，而经过量化以及Z扫描的数据序列含有大量的0，因此可以采用RLC来更进一步降低数据的传输量。假设RLC编码之后得到了(X，Y)，则X表示两个非零系数间零的个数，Y表示下一个非零系数的值。当从某一个系数开始剩余的数全部为零，则用（0，0）表示结束。例如现有一组字符串：8，6，0，0，0，0，16，0，-31，-7，0，0，1，0，…，0。经过RLC之后得到(0，8)；(0，6)；(4，16)；(1，-31)；(0，-7)；(2，1)；(0，0)。

### 2.7.3 AC 系数的中间格式

与DC系数中间格式一样，AC系数的中间格式也是通过前面提到的VLI表来进行转换的。如上文的(0，8)；(0，6)；(4，16)；(1，-31)；(0，-7)；(2，1)；(0，0)；对其只处理右边的数据，查找VLI表格，8在第四组，所以可以写成

RUN/SIZE/LEVEL的形式，即（0，4），8的形式，这种格式称为AC系数的中间格式。

### 2.7.4 Huffman 编码及解码

在对AC、DC系数进行转换之后，为了去除图像数据的冗余部分，需要对其再进行Huffman编码。在JPEG标准中有四张不同的Huffman编码表，由于篇幅原因只列出部分，如表2-2至2-5所示。

表2-2 亮度DC系数Huffman表

| 尺寸大小 | 码长 | 码字 |
|---|---|---|
| 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2-3 亮度AC系数Huffman表

| 行程/尺寸 | 码长 | 码字 |
|---|---|---|
| 0/0（EOB） | 4 | 1010 |
| 0/1 | 2 | 00 |
| 0/2 | 2 | 01 |
| … | … | … |
| 0/A | 16 | 1111111110000011 |
| 1/1 | 4 | 1100 |
| 1/2 | 5 | 11011 |
| 1/3 | 7 | 1111001 |
| … | … | … |
| F/9 | 16 | 1111111111111101 |
| F/10 | 16 | 1111111111111110 |

表2-4 色度DC系数Huffman表

| 尺寸大小 | 码长 | 码字 |
|---|---|---|
| 0 | 2 | 00 |
| 1 | 2 | 01 |
| 2 | 2 | 10 |
| 3 | 3 | 110 |

续表2-4 色度DC系数Huffman表

| 尺寸大小 | 码长 | 码字 |
|---|---|---|
| 4 | 4 | 1110 |
| 5 | 5 | 11110 |
| 6 | 6 | 111110 |
| 7 | 7 | 1111110 |
| 8 | 8 | 11111110 |
| 9 | 9 | 111111110 |
| 10 | 10 | 1111111110 |
| 11 | 11 | 11111111110 |

表2-5 色度AC系数Huffman表

| 行程/尺寸 | 码长 | 码字 |
|---|---|---|
| 0/0（EOB） | 2 | 00 |
| 0/1 | 2 | 01 |
| … | … | … |
| 0/A | 12 | 111111110100 |
| 1/1 | 4 | 1011 |
| 1/2 | 6 | 111001 |
| 1/3 | 8 | 11110110 |
| … | … | … |
| F/9 | 16 | 1111111111111101 |
| F/A | 16 | 1111111111111110 |

把中间格式按照以上的四个Huffman表格进行相应的转换即可得到最终的码字。如上文中直流系数的中间格式是（3）（7），3根据直流系数亮度的霍夫曼表得到100，7根据VLI表可以转换成111；同样地，交流系数中间格式（0，4），8；（0，4）查交流系数亮度霍夫曼表得到1011，8根据VLI表转换成1000；解码是编码的一个逆过程，根据Huffman表把码字转换成原来的数据，需要注意的是Huffman解码过程是无损的，解码后的数据和编码前的数据是一致的。

## 2.8 JPEG 文件格式

JPEG图像格式使用一系列的标记码（Marker Code）来标识和描述图像文件中不同部分的信息。它们以字节形式出现在JPEG文件中，用于标识图像数据的起始位置和各个数据块之间，以便解码器识别和解释图像数据。表2-6给出了部分JPEG标记码。

表2-6 JPEG标记码

| 标记码 | 数值 | 含义 |
| --- | --- | --- |
| SOI | FFD8 | 文件起始标志 |
| APP0 | FFE0 | 指定文件格式和其他应用特定信息 |
| DQT | FFDB | 定义量化表 |
| SOF0 | FFC0 | 帧头，包含图像基本信息 |
| DHT | FFC4 | 定义Huffman表 |
| SOS | FFDA | 扫描头，描述如何从图像提取数据 |
| DRI | FFDD | 定义重启间隔 |
| EOI | FFD9 | 文件结束标志 |

这些标记码帮助解析器识别和解释JPEG图像文件的组成和内容。JPEG文件中的不同段使用不同的标记码，这些标记码出现的顺序和包含的数据类型根据JPEG标准进行规定。图像数据通常存储在SOI和EOI之间，通过不同的标记段进行描述和控制。

## 2.9 本章小结

本章介绍了JPEG编解码的基本原理和流程，首先是图片的预处理部分，如颜色空间模型的转换，降采样；接着是图片压缩的核心部分，如二维离散余弦变换、量化、Z扫描、熵编码等，最后则对JPEG图像的文件格式进行解释。

# 第 3 章 图像编解码模块的研究与设计

本章节开始对JPEG编解码的各个模块进行硬件电路的设计实现，由于解码是编码的一个逆过程，所以着重介绍了预处理模块和编码中的每个模块，对于解码模块只说明了比较重要的Huffman解码模块和2D-IDCT模块。

## 3.1 JPEG 编码的基本架构

JPEG编码一种用于图像压缩的标准，常见的有两种压缩模式，一种是基线压缩，基线压缩适用于大多数应用场景，但在压缩比较高的情况下可能会引入一些可见的失真。另一种是渐进压缩，渐进压缩允许图像逐渐加载。在这种模式下，图像被分成多个扫描，每个扫描包含图像的不同频率的信息。渐进压缩对于网络较慢的情况或希望逐步显示图像的应用场景比较有用。本文采用的是常规的基线压缩模式。在这种模式下，图像数据首先经过预处理模块，接着进行二维离散余弦变换和量化处理，最后使用Huffman表进行熵编码以及字节填充等，该模式的基本框架如图3-1所示。

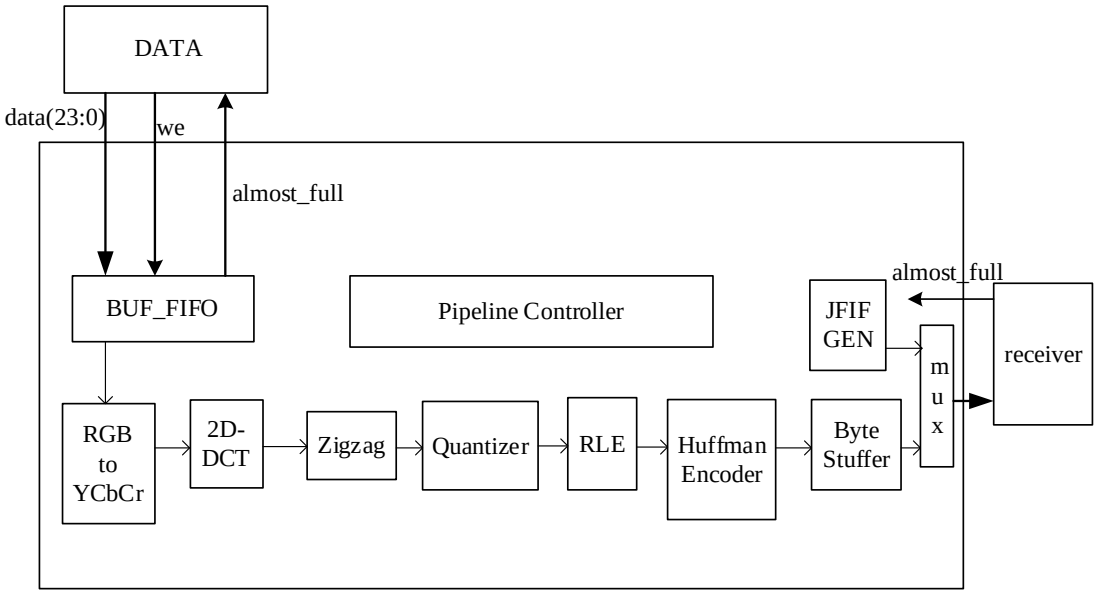



图3-1 JPEG编码基本框架

## 3.2 图像预处理模块

JPEG通常使用YCbCr色彩空间模型，而输入的图片格式一般是RGB颜色模型，为了进一步提升压缩效果减少数据量，所以需要先对图片进行预处理再进

行接下来的编码压缩操作。本文采取自顶向下的方法设计JPEG压缩的各个模块，并采用Verilog HDL硬件描述语言实现电路设计。

### 3.2.1 色彩空间转换模块

由于人眼对亮度的感知比色度更为敏感，通过将亮度和色度分离，JPEG能够有效地对图像进行压缩，其转换公式如上文公式(2-1)所示；因为公式中有浮点数，为了减少资源的消耗，所以将矩阵中的数值扩大256倍，再把得到的YCbCr结果右移8位得到最终需要的YCbCr的值，从而避免使用乘法器减少了硬件资源消耗。

RGB的取值范围是[0，255]，但需要YCbCr的取值范围控制在[-128，127]，因此还需要加一个偏移量。首先对输入的RGB数据分为三个颜色通道，每种颜色分量与扩大256倍的矩阵系数相乘，再把相乘后的结果与其他两种颜色分量结果相加减，最后再给一个偏移量就得到最终的YCbCr的值，其基本实现过程如图3-2所示。

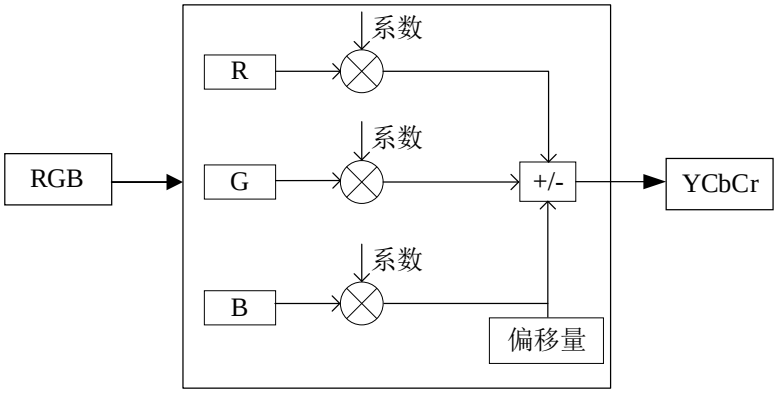



图3-2 RGB2YCbCr的实现过程图

图3-3是在Vivado中实现颜色空间转换功能的仿真图。

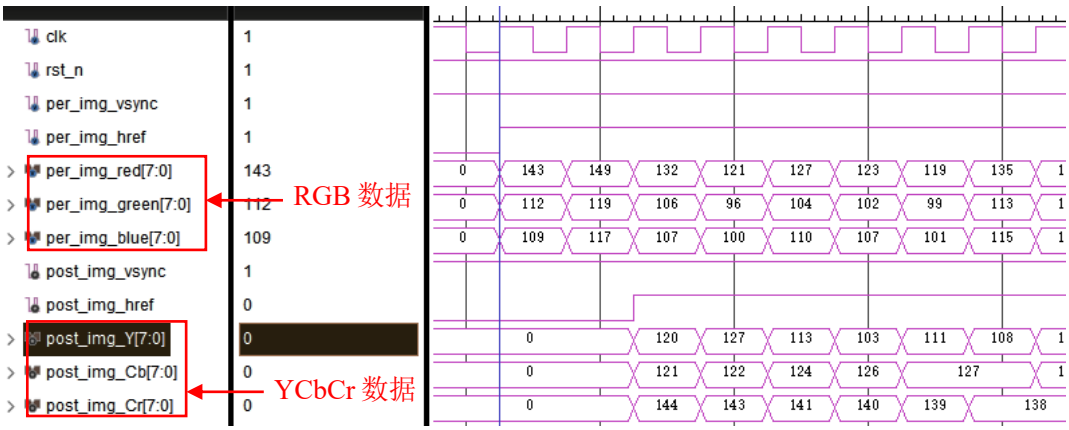



图3-3 RGB2YCbCr的仿真波形图

图3-3中的pre_img_red、pre_img_green、pre_img_blue分别是输入的RGB数值，post_img_Y、post_img_Cb、post_img_Cr是转换后Y、Cb、Cr的值。为了检验转换结果的正确性在Matlab中用同样的RGB数据进行了验证。输入的RGB数据如图3-4所示，其Matlab结果如图3-5所示。

| | VarName1 | VarName2 | VarName3 | VarName4 | VarName5 | VarName6 | VarName7 | VarName8 | VarName9 | VarName10 | VarName11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 分类 | 数值 | 数值 | 分类 | 数值 | 数值 | 分类 | 数值 | 数值 | 分类 | 数值 |
| 1 | 8f | 70 | 6d | 95 | 77 | 75 | 84 | 6a | 6b | 79 | 60 |
| 2 | 8e | 6f | 6c | 93 | 75 | 73 | 87 | 6a | 6c | 79 | 60 |
| 3 | 8f | 6d | 6b | 92 | 73 | 71 | 8c | 6e | 70 | 80 | 65 |
| 4 | 8c | 6a | 68 | 91 | 6f | 6e | 91 | 71 | 72 | 8a | 6b |
| 5 | 90 | 6d | 69 | 93 | 70 | 6e | 95 | 73 | 72 | 8d | 6d |
| 6 | 9c | 77 | 71 | 9c | 76 | 73 | 9a | 77 | 73 | 94 | 71 |
| 7 | a0 | 7b | 73 | a2 | 7d | 75 | a1 | 7c | 76 | 9d | 78 |
| 8 | 9c | 77 | 6e | a2 | 7b | 74 | a4 | 7d | 76 | a2 | 7d |
| 9 | a4 | 7f | 77 | a7 | 82 | 7a | a4 | 7f | 77 | 9f | 7a |
| 10 | a2 | 7d | 77 | a5 | 80 | 78 | a5 | 80 | 78 | a0 | 7b |
| 11 | a0 | 7b | 75 | a3 | 7e | 76 | a5 | 80 | 78 | a2 | 7d |
| 12 | a0 | 7b | 75 | a1 | 7c | 76 | a2 | 7d | 75 | a0 | 7e |

图3-4 Matlab中RGB的输入数据

| | VarName1 | VarName2 | VarName3 | VarName4 | VarName5 | VarName6 | VarName7 | VarName8 | VarName9 |
|---|---|---|---|---|---|---|---|---|---|
| | 数值 | 数值 | 数值 | 数值 | 数值 | 数值 | 数值 | 数值 | 数值 |
| 1 | 78 | 79 | 90 | 7f | 7a | 8f | 71 | 7c | 8d |
| 2 | 77 | 79 | 90 | 7d | 7a | 8f | 72 | 7c | 8e |
| 3 | 76 | 79 | 91 | 7b | 7a | 90 | 76 | 7c | 8f |
| 4 | 73 | 79 | 91 | 78 | 7a | 91 | 7a | 7b | 90 |
| 5 | 76 | 78 | 92 | 79 | 79 | 92 | 7c | 7a | 91 |
| 6 | 80 | 77 | 93 | 80 | 78 | 93 | 80 | 78 | 92 |
| 7 | 84 | 76 | 93 | 86 | 76 | 93 | 85 | 77 | 93 |
| 8 | 80 | 75 | 93 | 85 | 76 | 94 | 87 | 76 | 94 |
| 9 | 88 | 76 | 93 | 8b | 76 | 93 | 88 | 76 | 93 |
| 10 | 86 | 77 | 93 | 89 | 76 | 93 | 89 | 76 | 93 |

图3-5 Matlab中RGB2YCbCr的结果

Matlab中数据是用十六进制表示，经过对比发现与在Vivado中仿真结果一样，说明仿真无误。

### 3.2.2 下采样模块

JPEG采用色度子采样的目的是在降低图像文件大小的同时尽量保持图像质量。它仅降低色度分量的分辨率[32]而亮度分量的分辨率则保持不变。常见的色度采样比例包括4:4:4、4:2:2和4:2:0。这里采用的是4:2:2的采样比例，这种子采样方式有助于减小图像文件的大小，同时保持相对较好的视觉质量。为了验证4:2:2下采样模式下图片效果，在进行硬件设计之前在Matlab中演示图片的效果，图3-6显示了Matlab中进行4:2:2下采样之后的图片与原图之间的对比。


图3-6 原图和下采样之后重构图片的对比

从图3-6中可以观察到二者在整体质量方面基本没有差异，但由于色度上采用了4:2:2的下采样，在竹林背景的色彩方面有着细小差距。

在Matlab中验证其效果之后，在Vivado中通过Verilog语言进行硬件电路的设计，图3-7是在Vivado中下采样的仿真图。

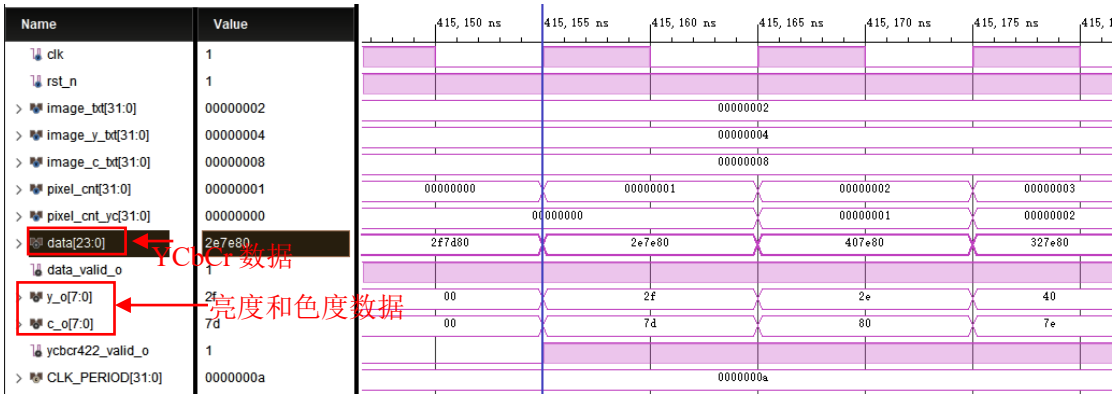



图3-7 Vivado中下采样的波形仿真图

Data[23:0]是输入的YCbCr数据，其中data[23:16]表示亮度分量Y、data[15:8]表示色度分量Cb、data[7:0]表示色度分量Cr；y_o[7:0]和c_o[7:0]分别是下采样之后的亮度和色度分量。

## 3.3 JPEG 编码各模块设计

经过上述的预处理步骤后，可以对图像数据进行编码操作，本文讨论的是JPEG压缩算法，它是一种有损压缩算法，通过对数据进行二维离散余弦变换、量化、熵编码等步骤来减少文件大小。下面是它的各个模块的设计实现。

### 3.3.1 二维离散余弦变换模块

$N\times N$大小的二维离散余弦变换，$T_{DCT}(u,v)$公式如(3-1)、(3-2)所示。

$$T_{DCT}(u,v)=\alpha(u)\alpha(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\cos\left[\frac{\pi(2i+1)u}{2N}\right]\cos\left[\frac{\pi(2j+1)v}{2N}\right] \tag{3-1}$$

$$\alpha(u)\alpha(v)=\begin{cases}1/\sqrt{N} & u,v=0\\ 2/\sqrt{N} & \text{其他}\end{cases} \tag{3-2}$$

其中，$T_{DCT}(u,v)$表示为变换后得到的频域矩阵，$u$是水平方向的频率，$v$是垂直方向的频率，对于JPEG标准，$N$=8，$i$、$j$分别表示空间域的一个$N\times N$矩阵的行数、列数。

SDCT是有符号离散余弦变换[33]，因为SDCT是从DCT导出的，所以$T_{SDCT}(u,v)$是将*signum*函数应用在DCT的元素上得到的。

$$T_{SDCT}(u,v)=\frac{1}{\sqrt{N}}sign\{T_{DCT}(u,v)\} \tag{3-3}$$

其中，*signum*函数公式如(3-4)所示。

$$sign\{x\}=\begin{cases}+1 & \text{x>0,}\\ 0 & \text{x=0,}\\ -1 & \text{x<0.}\end{cases} \tag{3-4}$$

SDCT有以下几个优点：（1）所有的元素都是0或者±1。（2）不需要乘法运算和超越表达式。（3）SDCT保留了原始DCT的周期性和频谱结构，保持了良好的去相关和能量压缩特性。因此，SDCT非常适合计算低复杂度的应用。

8×8的SDCT变换矩阵如公式(3-5)所示。

$$T_{SDCT}=\frac{1}{\sqrt{8}}\begin{bmatrix}+1 & +1 & +1 & +1 & +1 & +1 & +1 & +1\\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1\\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1\\ +1 & -1 & -1 & -1 & +1 & +1 & +1 & -1\\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1\\ +1 & -1 & +1 & +1 & -1 & -1 & +1 & -1\\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1\\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1\end{bmatrix} \tag{3-5}$$

$T_{SDCT}$的前向变换几乎是正交的，但是它的逆变换不是正交的。因此该矩阵可以用于前向变换的自适应滤波，本设计基于此提出了一种新的DCT变换矩阵，只需要将SDCT的变换矩阵中的部分项中的元素更改为0，就可以得到所提出的新的变换矩阵，所提出的矩阵包含16个0，具体矩阵如下所示。

$$T_0=\begin{bmatrix}1&1&1&1&1&1&1&1\\1&1&1&0&0&-1&-1&-1\\1&0&0&-1&-1&0&0&1\\1&0&-1&-1&1&1&0&-1\\1&-1&-1&1&1&-1&-1&1\\1&-1&0&1&-1&0&1&-1\\0&-1&1&0&0&1&-1&0\\0&-1&1&-1&1&-1&1&0\end{bmatrix} \tag{3-6}$$

矩阵$T_0$非常简单，其组成元素只有0、1、-1，但是它也有一个很大的缺点就是缺乏正交性，这样就不能直接运用在图像压缩方面，因此利用矩阵极分解理论[34]找一个可以使其正交化的调整矩阵，调整矩阵由下式给出：$D=\sqrt{(T_0\cdot T_0^T)^{-1}}$，$D=diag(1/\sqrt{8},1/\sqrt{6},1/2,1/\sqrt{6},1/\sqrt{8},1/\sqrt{6},1/2,1/\sqrt{6})$。因此，最后的DCT矩阵为：$T=D\cdot T_0$。矩阵$T$有以下几个优点：（1）它是正交的。（2）它的复杂度比较低。（3）正交化矩阵是对角的。

由于矩阵$T$满足正交的性质，所以有$T^{-1}=T^t$，其中右上角的t表示转置运算，这样就可以使用相同的矩阵进行图像编解码，设$X$是图像数据的8×8块，$Y$是在变换域对应的矩阵，那么正向变换为：$Y=TXT^t$；$T$是正交的，那么就可以使用反向变换重建图像，即$X=T^tYT$。

二维离散余弦变换（2D-DCT）将图像信息从空域变换到频域，如果直接按照离散余弦变换的公式来计算，那么对于一个8×8的图像块需要4096次乘法和4096次加法，这样的运算量太大，消耗的资源会比较多，而且图片编码的速度也会大受影响。因此需要一个快速算法来实现2D-DCT。

已知的2D-DCT的快速算法有很多，如Chen算法[35]、Loeffler算法[36]等，这些都是目前使用的比较多的算法，chen氏算法有着比较好的结构以及准确的精度，而且计算的速度也相对较快，所以在设计中采用的是chen氏算法，该算法是通过分解，利用对称性的方式来提高DCT的计算效率方法，它将传统的二维离散余弦变换将其分解成两次的一维离散余弦变换（1D-DCT）分别进行计算，一维离散余弦变换可以用$Y=CX$表示，以8×8图像块为例，原始DCT变换公式展开如下。

$$\begin{pmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \end{pmatrix} = \begin{pmatrix} C0 & C0 & C0 & C0 & C0 & C0 & C0 & C0 \\ C1 & C3 & C5 & C7 & -C7 & -C5 & -C3 & -C1 \\ C2 & C6 & -C6 & -C2 & -C2 & -C6 & C6 & C2 \\ C3 & -C7 & -C1 & -C5 & C5 & C1 & C7 & -C3 \\ C4 & -C4 & -C4 & C4 & C4 & -C4 & -C4 & C4 \\ C5 & -C1 & C7 & C3 & -C3 & -C7 & C1 & -C5 \\ C6 & -C2 & C2 & -C6 & -C6 & C2 & -C2 & C6 \\ C7 & -C5 & C3 & -C1 & C1 & -C3 & C5 & -C7 \end{pmatrix} \begin{pmatrix} x0 \\ x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \end{pmatrix} \tag{3-7}$$

公式中的$C0 \sim C7$是一系列的余弦值，具体数值如公式(3-8)所示，如果直接将数值直接代入公式(3-7)中得到的DCT变换矩阵相对比较复杂，运算过程消耗的资源也会较大，在上文中给出了新的DCT变换矩阵$T = D \cdot T_0$，用新的DCT变换矩阵代替公式(3-7)中的变换矩阵那么就得到了公式(3-9)。

$$\begin{bmatrix} C0 \\ C1 \\ C2 \\ C3 \\ C4 \\ C5 \\ C6 \\ C7 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \cos(4\pi/16) \\ \cos(\pi/16) \\ \cos(2\pi/16) \\ \cos(3\pi/16) \\ \cos(4\pi/16) \\ \cos(5\pi/16) \\ \cos(6\pi/16) \\ \cos(7\pi/16) \end{bmatrix} \tag{3-8}$$

$$\begin{pmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \end{pmatrix} = \begin{pmatrix} T0 & T0 & T0 & T0 & T0 & T0 & T0 & T0 \\ T1 & T1 & T1 & 0 & 0 & -T1 & -T1 & -T1 \\ T2 & 0 & 0 & -T2 & -T2 & 0 & 0 & T2 \\ T1 & 0 & -T1 & -T1 & T1 & T1 & 0 & -T1 \\ T0 & -T0 & -T0 & T0 & T0 & -T0 & -T0 & T0 \\ T1 & -T1 & 0 & T1 & -T1 & 0 & T1 & -T1 \\ 0 & -T2 & T2 & 0 & 0 & T2 & -T2 & 0 \\ 0 & -T1 & T1 & -T1 & T1 & -T1 & T1 & 0 \end{pmatrix} \begin{pmatrix} x0 \\ x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \end{pmatrix} \tag{3-9}$$

其中x为输入的图片像素数据，$[T0, T1, T2] = [1/\sqrt{8}, 1/\sqrt{6}, 1/2]$，经观察可以发现变换矩阵有着对称性，因此可以将其化简，具体如公式(3-10)、(3-11)所示。

$$\begin{pmatrix} y0 \\ y2 \\ y4 \\ y6 \end{pmatrix} = \begin{pmatrix} T0 & T0 & T0 & T0 \\ T2 & 0 & 0 & -T2 \\ T0 & -T0 & -T0 & T0 \\ 0 & -T2 & T2 & 0 \end{pmatrix} \begin{pmatrix} x0+x7 \\ x1+x6 \\ x2+x5 \\ x3+x4 \end{pmatrix} \tag{3-10}$$

$$\begin{pmatrix} y1 \\ y3 \\ y5 \\ y7 \end{pmatrix} = \begin{pmatrix} T1 & T1 & T1 & 0 \\ T1 & 0 & -T1 & -T1 \\ T1 & -T1 & 0 & T1 \\ 0 & -T1 & T1 & -T1 \end{pmatrix} \begin{pmatrix} x0-x7 \\ x1-x6 \\ x2-x5 \\ x3-x4 \end{pmatrix} \tag{3-11}$$

由于矩阵系数有很多浮点数，为了方便运算需要将浮点数的值全都扩大256倍再取整[37]再代入得到公式(3-12)、(3-13)。

$$\begin{pmatrix} y0 \\ y2 \\ y4 \\ y6 \end{pmatrix} = \begin{pmatrix} 91 & 91 & 91 & 91 \\ 128 & 0 & 0 & -128 \\ 91 & -91 & -91 & 91 \\ 0 & -128 & 128 & 0 \end{pmatrix} \begin{pmatrix} x0+x7 \\ x1+x6 \\ x2+x5 \\ x3+x4 \end{pmatrix} \tag{3-12}$$

$$\begin{pmatrix} y1 \\ y3 \\ y5 \\ y7 \end{pmatrix} = \begin{pmatrix} 104 & 104 & 104 & 0 \\ 104 & 0 & -104 & -104 \\ 104 & -104 & 0 & 104 \\ 0 & -104 & 104 & -104 \end{pmatrix} \begin{pmatrix} x0-x7 \\ x1-x6 \\ x2-x5 \\ x3-x4 \end{pmatrix} \tag{3-13}$$

可以发现矩阵比原始的DCT矩阵多出了一些0，这样在硬件电路设计中当遇到利用含有0的乘法和加法运算时直接将结果赋值为0，根据上式进行1D-DCT变换只需要24次乘法和24次加法，减少了乘法器和加法器的调用，带来节省资源的好处，减少了电路的复杂度和延迟。当得到最终的计算结果后需要再把数值舍弃掉低8位，根据对称性设计如图3-8所示的一维离散余弦变换的结构图。

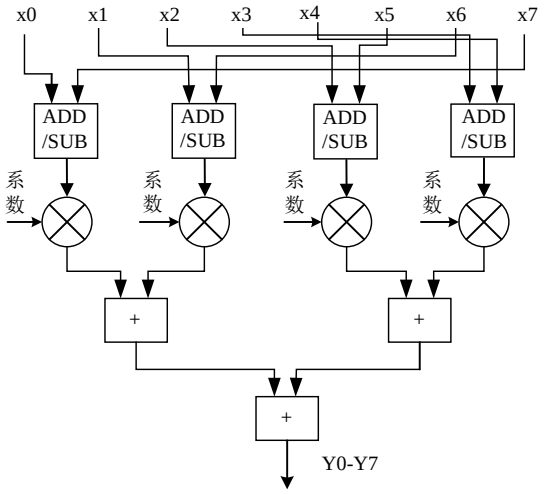



图3-8 一维DCT变换的结构图

当第一级离散余弦变换完成后，需要再对列进行一次离散余弦变换，所以需要在两级离散余弦变换之间加一个转置缓存器来实现矩阵的转置功能，最容易的方法是用8×8个寄存器来完成，虽然这样做控制比较简单不需要读取地址但要耗费较多的逻辑资源，而且FPGA有丰富的RAM资源所以考虑使用RAM。为了更好地让数据连续进行读取采用两块RAM以乒乓操作形式进[38]行读写模式。

图3-9是二维离散余弦变换的硬件模块结构；从图中可知该模块由两个一维离散余弦变换模块加上选择单元以及两块RAM组成，首先8×8的像素数据进入一维离散余弦变换模块进行行变换，变换之后的数据通过输入数据选择模块选择RAM1并将数据缓存到其中，其次切换输入数据选择模块选择RAM2，并将数据缓存到其中，同时输出数据选择模块选择RAM1并将里面的数据按列送到二级DCT进行变换得到2D-DCT的最终结果。RAM1、RAM2交替进行读写操作，像上述过程循环往复保证了数据的无缝处理。

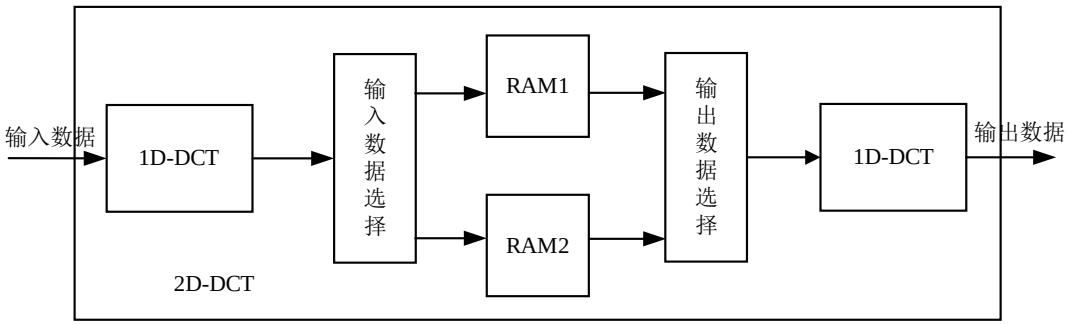



图3-9 2D-DCT变换的硬件模块结构

接下来通过输入一组数据来验证二维离散余弦变换模块的正确性，首先在Matlab中直接用2D-DCT函数算出结果，然后用相同的数据输入到Vivado中的2D-DCT模块进行测试，最后将二者的数据进行对比。图3-10显示的是Matlab的输出，图3-11是Vivado中仿真的结果。

8x8 double

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | -691.3750 | 16.5414 | -8.1776 | 6.5504 | -2.6250 | 2.4224 | -2.4306 | 1.6335 | |
| 2 | 16.6184 | -7.6318 | 6.7264 | -6.4260 | 2.5826 | -1.9785 | 1.7392 | -0.3207 | |
| 3 | -3.0470 | 7.1367 | -4.6365 | 3.1345 | -1.7965 | 1.1797 | -1.1705 | 0.6441 | |
| 4 | 2.9724 | -3.8035 | 1.0147 | -0.6780 | 0.4588 | -1.6795 | 1.3450 | -0.2299 | |
| 5 | -0.3750 | 1.6356 | -1.3742 | 1.6432 | -1.6250 | 0.9476 | -0.3779 | 0.1979 | |
| 6 | 1.6703 | -0.9092 | 1.1185 | -0.3992 | 0.6953 | -0.2132 | -0.0300 | -1.1822 | |
| 7 | -0.4968 | 0.8391 | -0.4205 | -0.1587 | -0.7441 | 0.4356 | 0.1365 | 0.6261 | |
| 8 | 0.5145 | -0.5404 | -0.0917 | 0.3572 | 1.2037 | 0.2131 | -0.0846 | 0.0230 | |
| 9 | | | | | | | | | |

图3-10 Matlab中2D-DCT的结果

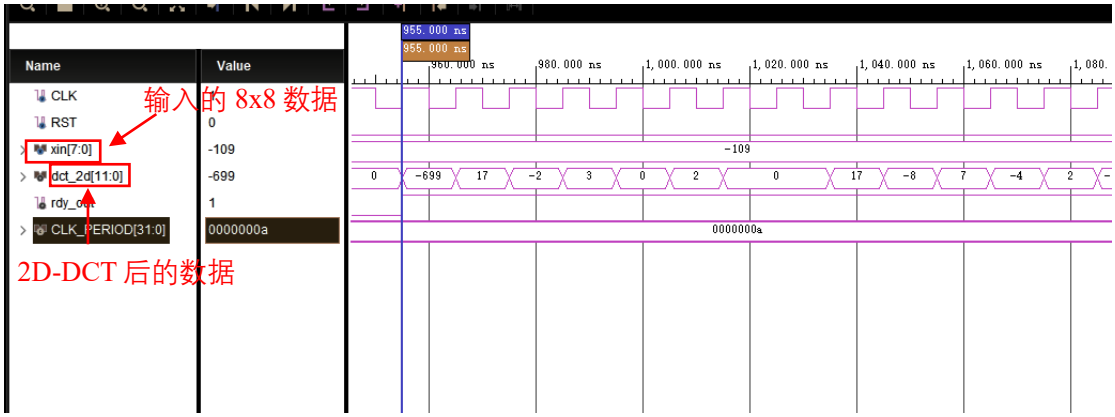



图3-11 Vivado中2D-DCT仿真波形

从上面两幅图中可以得知Matlab中的数据和Vivado仿真中的数据有着微小偏差，而有偏差是因为Matlab中直接调用的2D-DCT函数，而Vivado中是改进后低复杂度的变换矩阵，以及为了抵消浮点数放大256倍而采用的高位截取引起的。

改进后低复杂的变换矩阵的模块进行仿真测试，模块的工作频率设置在100MHz时，完成一个$8\times8$数据的DCT变换需要159个周期，约为2.1微秒，能够满足实时应用场景。

### 3.3.2 量化和 Z 扫描模块

JPEG编码中量化是在保持图像基本质量的情况下丢掉对视觉效果影响不大的高频部分来提高压缩的程度大小，量化就是将输入的DCT系数与量化系数相除并取整。人眼对亮度和色度的感知不一样，量化表也分为两种。在JPEG标准给出了量化表，如公式(2-4)、(2-5)所示；当然这个表并不是固定不变的，根据对图像的质量需求也可以按照标准量化矩阵和质量因数($QF$)[39]来自己制定量化表。计算公式如(3-14)所示。

$$Q(u,v)=\begin{cases}\max\left(\left\lfloor(2-\dfrac{QF}{50})Q_0(u,v)+0.5\right\rfloor,1\right), & 50\le QF\le100\\ \left\lfloor\dfrac{50}{QF}Q_0(u,v)+0.5\right\rfloor, & 0<QF<50\end{cases} \tag{3-14}$$

其中，$Q_0(u,v)$是标准量化表中$(u,v)$位置的量化步长，从公式中可以得知质量因子越大，量化步长就越小，得到的图片质量也就越好。图3-12是不同质量因子下的图片大小和显示效果。

从图3-12中可以看出$QF$越大图片效果也就越好，但是图片大小也会相应的增加，为了更好地平衡压缩和显示效果，综合考虑下选择$QF=50$下的量化表。

| QF=10 | QF=30 | QF=50 | QF=80 | QF=100 |
|---|---|---|---|---|
| 11.98KB | 20.22KB | 27.55KB | 51.17KB | 417.4KB |

图3-12 不同质量因子下重构的图片

如果在量化中直接用除法会比较复杂同时也比较耗费逻辑资源，因此在普通的量化阶段中是取量化值的倒数，用乘法运算来代替除法，然后将得到的结果进行四舍五入取整，但在本设计中提出了一种近似的量化表，表中的量化步长全是2的N次幂的整数，所以这种近似不需要乘法可以直接通过移位运算来实现DCT系数的量化，这里引入了量化表$Q$的近似值，如下式所示。

$$Q = F(Q) \tag{3-15}$$

其中$F(x) = 2^{round(\log_2 x)}, x \in \mathbb{R}$，对于亮度量化矩阵$Q_L$获得下面的近似矩阵。

$$Q_L = \begin{bmatrix} 16 & 8 & 8 & 16 & 32 & 32 & 64 & 64 \\ 16 & 16 & 16 & 16 & 32 & 64 & 64 & 64 \\ 16 & 16 & 16 & 24 & 32 & 64 & 64 & 64 \\ 16 & 16 & 16 & 32 & 64 & 64 & 64 & 64 \\ 16 & 16 & 32 & 64 & 64 & 128 & 128 & 64 \\ 32 & 32 & 64 & 64 & 64 & 128 & 128 & 64 \\ 64 & 64 & 64 & 64 & 128 & 128 & 128 & 128 \\ 64 & 64 & 64 & 128 & 128 & 128 & 128 & 128 \end{bmatrix} \tag{3-16}$$

为了看到新的量化矩阵在实际应用中的效果，本文在Matlab中进行了实际验证，图3-13显示了在不同图片中原量化矩阵和改进后的矩阵显示的效果；其中Cameraman是低频图像，Barbara是中频图像，Baboon是高频图像，图a为原量化矩阵效果图，图b为改进量化表之后的效果图。

从图3-13可以看到原图和改进后的效果图之间的区别基本可以忽略不计，且改进后的量化表在量化阶段不需要乘法运算，减少了资源消耗，且量化矩阵可以直接在JPEG编解码器中使用而无需任何修改。当编码时：$Y_{i,j}^{quant} = Y_{i,j} \gg \log_2 x_{i,j}$，$\gg$表示右移运算，当解码时：$Y_{i,j}^{quant} = Y_{i,j} \ll \log_2 x_{i,j}$，$\ll$

表示左移运算，公式中的$Y_{i,j}$是经过变换的DCT系数，$x_{i,j}$为量化矩阵$Q_L$中第i行第j列的元素值，量化矩阵$Q_L$中的元素全部都是2的整数次幂，所以可以直接利用移位操作在硬件上实现这一步骤。

(a) (b) (a) (b) (a) (b)

Cameraman Barbara Baboon

图3-13 原图与改进后的对比

量化之后8×8矩阵中右下角的很多高频数据变为0，Z字形扫描将这些数据重新排序成一维序列，其中前面的数据为低频部分，后面则为高频数据，这样利于后面的数据编码来进一步压缩数据大小。由于量化和Z字扫描实现相对比较简单可以把二者结合起来一起实现。

首先DCT系数进入量化阶段，将亮度和色度的量化表存放在两块ROM中，根据读地址来读取这些数据并取其对数，最后将DCT系数进行移位得到量化后数据。将量化后的数据存到RAM中，两块RAM进行乒乓操作将数据源源不断地输入到Z扫描阶段，Z扫描同样也是按照乒乓操作进行读写，RAM1接受量化后的数据时，另一RAM在根据ROM输出的地址内容以Z扫描的顺序读取数据，最后将数据排列成长度为64的一维序列。其设计框图如3-14所示。

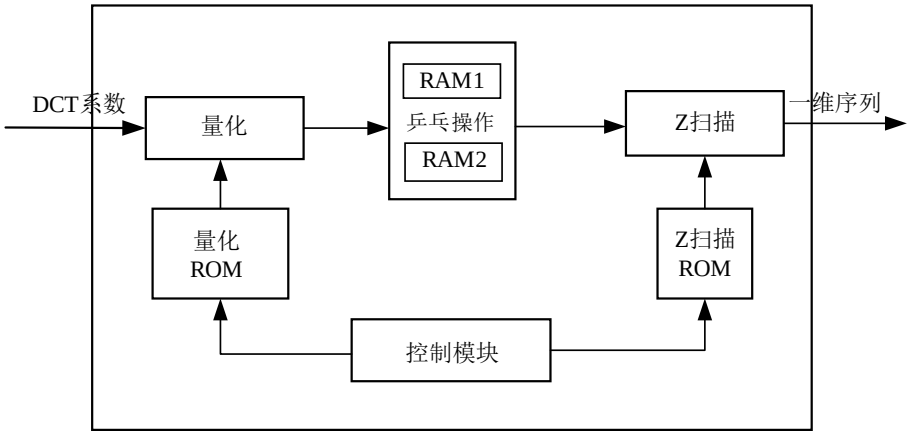



图3-14 量化与Z扫描模块框图

### 3.3.3 熵编码模块

DCT系数在经过量化和Z字形扫描后会出现连续的0，因此行程编码可以实

现压缩数据的目的[40]，利用该编码方式，可以将重复出现的连续字符用两个字节来代替。霍夫曼编码[41]则是把直流系数（DC）和交流系数（AC）的中间格式通过霍夫曼表转换成码字。其具体转换过程在第二章已经详细阐述在此不再赘述。

霍夫曼编码中RUN/SIZE的位数一共有8比特，RUN和SIZE各占四比特位宽。每个8×8的块的直流分量的RUN的值一直是零，SIZE范围是0-11，如果SIZE=0则说明当前DC系数和前一个图像块的DC系数相同。邻近的DC系数的差值范围是-1024～+1023。用二进制的反码来表示。而对于AC系数，RUN的范围是0-F，SIZE的范围是1-A，但需要注意的是有其他的特殊情况，当RUN/SIZE=0/0时，说明剩下的系数均为零并产生EOB信号。当RUN/SIZE=F/0，说明有连续十六个零。

熵编码在JPEG编码中比较复杂因此把它拆分成几个关键部分分别介绍，通过相关理论的介绍可知，在对DC系数进行霍夫曼编码之前需要计算其差值及其反码。计算差值和反码的过程如图3-15所示。

当前输入如果是DC系数则进行锁存同时选择前一个DC系数。对正数而言它的反码就是它本身，若为负数，反码是绝对值按位取反，图中ABS为绝对值函数。

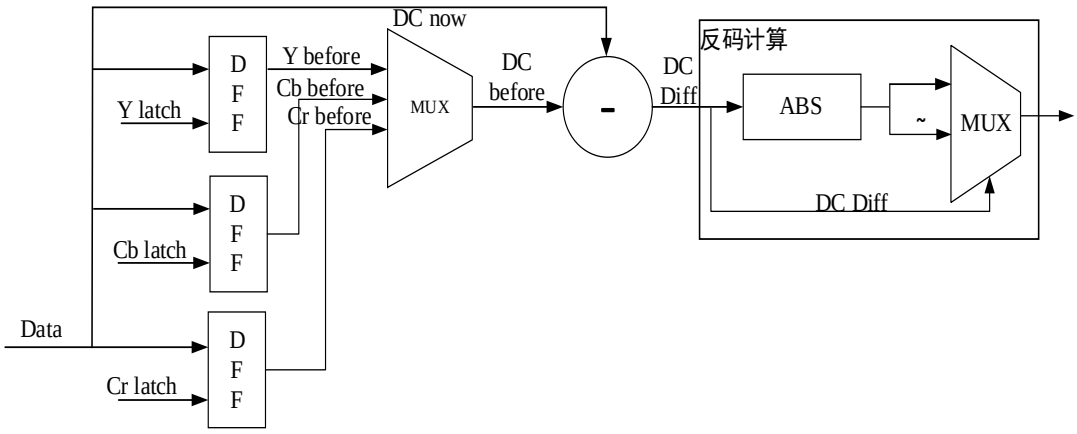



图 3-15 差值及其反码计算

得到差值后可以通过它得到SIZE的值，在这种情况下通常采用多个MUX组成一棵树，通过选择不同的输入路径实现对应的差值。具体的结构图如3-16所示，MUX树根据绝对值大小从高到低判断SIZE大小。

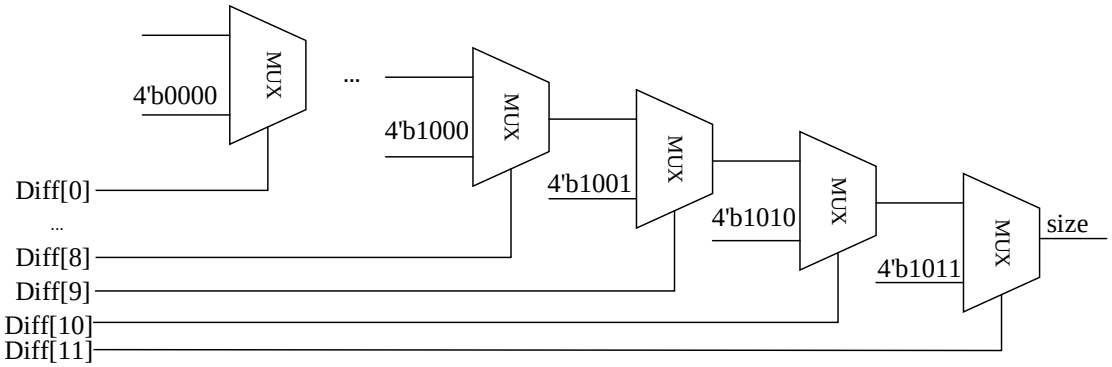



图3-16 SIZE的计算

接下来Huffman编码模块执行编码操作，编码后将并行数据转换成串行比特流，串行比特流输出被打包成字节，然后存储在输出FIFO中。整个编码以8×8块为单位进行的，其整个基本过程如图3-17所示。

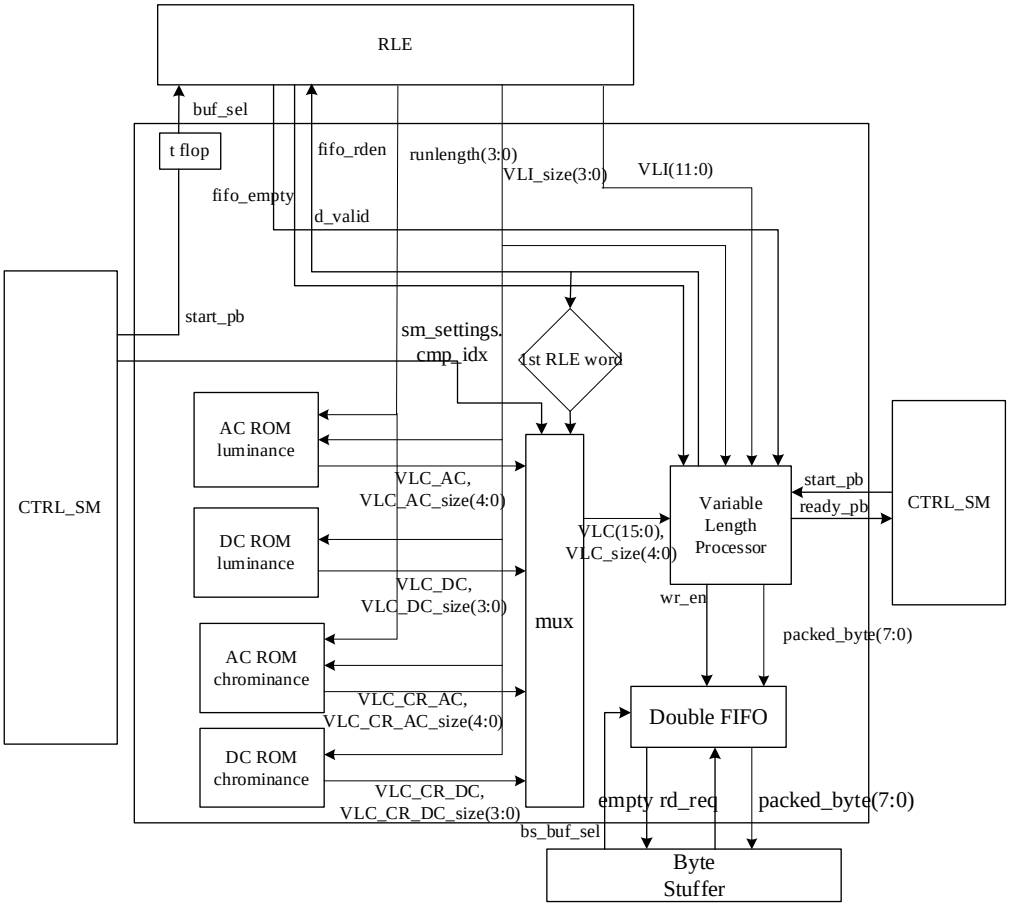



图3-17 Huffman编码结构图

当start_pb有效时开始处理，此时将first_RLE_word置1，当对RLE的第一个数据的fifo读取请求发生时又重置为0，这样在处理DC系数时使用DC ROM，对于其他码字则使用AC ROM，确保在不同情况下使用相应的ROM进行操作。RLE输出的数据由RUNLENGTH和VLI组成，AC/DC ROM将RUNLENGTH/VLI

转换成可变长度编码（VLC），buf_sel交替出现用于输出缓冲器的双重缓冲。

其中上图中的Variable Length Processor是主状态机和核心处理逻辑，由VLP_ctrl有限状态机驱动，VLP负责管理Huffman编码的过程，它通过read_req从RLE读取数据，直到RLE fifo为空。其结构图如图3-18所示。

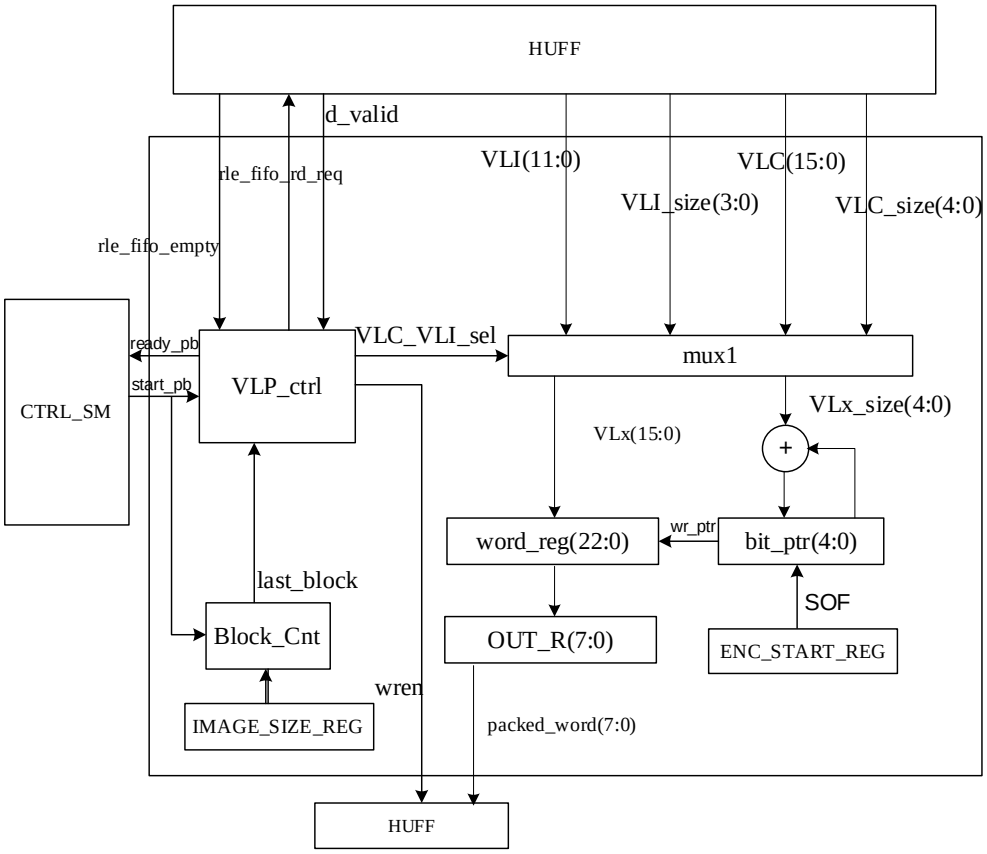



图3-18 VLP的结构图

VLP首先对VLC编码对进行串行化，然后是VLI编码对，生成比特流并划分为多个输出字节。图3-18中的bit_ptr(4:0)是五位的指针，word_reg寄存器用于在处理过程中临时存储数据，bit_ptr记录word_reg中上一次使用的位，当SOF(Start of Frame)有效时，即在图像帧开始的时候bit_ptr被重置为零，当start_pb有效时bit_ptr不变，即在处理图像帧的开始之外的情况下保持其上一次使用的状态。VLC/VLI的串行位映射到字寄存器word_reg，从bit_ptr位置开始，通过bit_ptr右移当前VLC/VLI词中的位数，bit_ptr从word_reg的MSB开始，bit_ptr=0表示word_reg的MSB。在当前块的所有VLC/VLI码字被串行化并存储到输出FIFO后，ready_pb信号激活表示输出FIFO已经准备好接收数据。

### 3.3.4 码流组装模块

当完成对图像数据的一系列压缩之后，还需要将数据按照一定的格式输出才能形成最终的JPEG码流，目前使用比较广泛的是JFIF(JPEG File Interchange),它定义了一种通用的JPEG文件格式[42]，以简化JPEG图像的交换和共享。图3-19为码流组装的结构示意图。

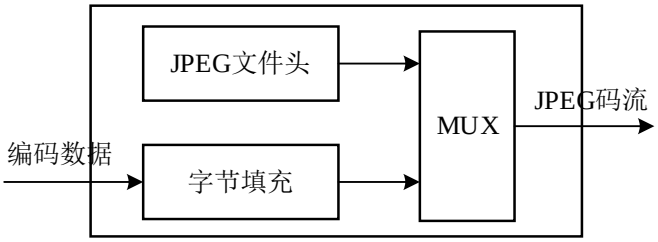



图3-19 码流重组结构

为了确保在熵编码段内不会出现错误标记，字节填充模块会对从Huffman编码模块读取的每个字节都进行检查，检查是否含有0XFF字节，如果有0XFF字节则必须用0X00字节进行填充。例如读取的数据是以下字节：AA CC BB FF 22 33，那么经过字节填充后会变成：AA CC BB FF 00 22 33。编码刚开始时MUX模块输出标准JFIF头，之后开始输出图像真正数据，在完成一帧图像数据后产生EOI标记（End of Image）代表一帧图像数据编码的完成。

## 3.4 JPEG 解码的基本架构

在完成图像的编码工作以后，图像数据被压缩成一连串的码字，为了将这些数据码流还原成图像还需要进行JPEG解码，JPEG解码是将编码压缩的图像数据还原为原始图像的一个过程，是编码的一个逆过程。它的整体结构如图3-20所示。

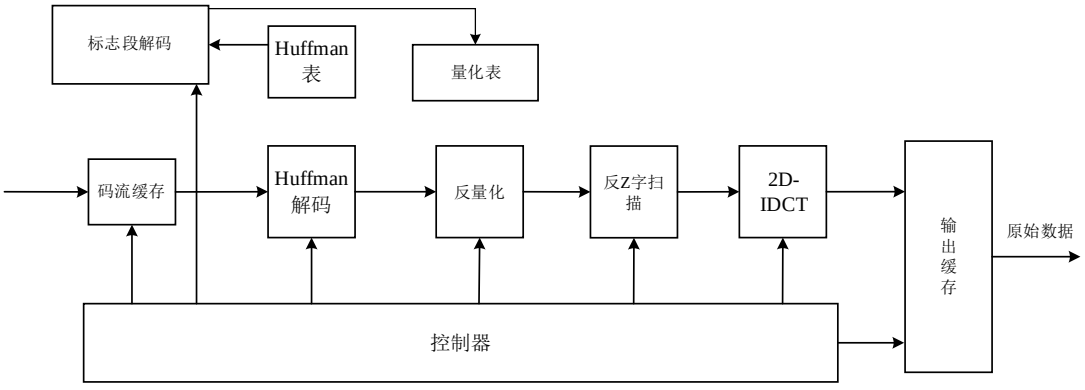



图3-20 JPEG解码框图

## 3.5 JPEG 解码各模块设计

由于JPEG解码是编码的一个逆过程，所以JPEG解码的各模块和编码的模块也基本相同，在这里着重介绍了比较重要的Huffman模块以及2D-IDCT模块。

### 3.5.1 Huffman 解码模块

霍夫曼编码的原理和详细过程在上文中已经叙述，这里直接给出整体Huffman解码的设计，如图3-21所示。

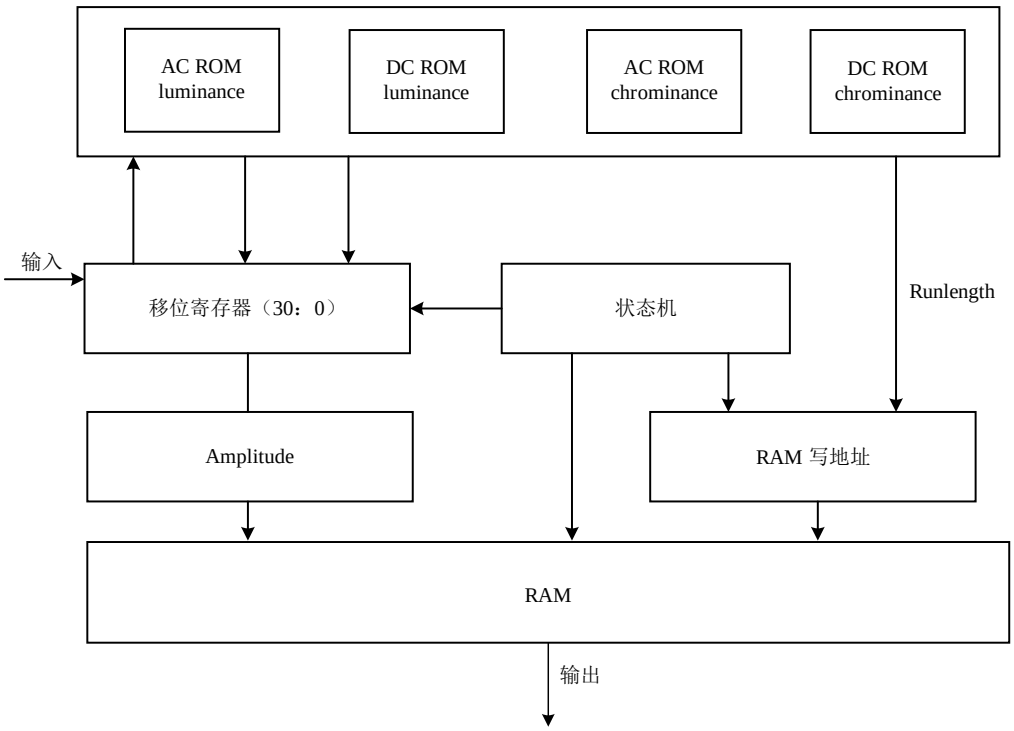



图3-21 Huffman解码模块结构图

由上图可知该模块主要是由霍夫曼查找表，缓存RAM，移位寄存器以及状态机组成。其中AC ROM、DC ROM存储的霍夫曼编码表；缓存RAM含有多个区域确保数据正确读取；移位寄存器用来缓存数据；状态机控制整个流程。

首先数据输入到移位寄存器，移位寄存器的高十六位进行Huffman查表，找出其中的VLC、Runlength和Size。接着移位寄存器开始移位以去除其中的VLC码字，然后根据VLI Size截取移位寄存器相应位置得到VLI数值，同时将其转化为Amplitude，并且由Runlength生成RAM写地址，最后查看数据是否解码完成并跳转到最开始的状态。

### 3.5.2 2D-IDCT 模块

2D-IDCT是2D-DCT的逆变换，用于将图像从频域还原到空间域以恢复原始图像，两者的公式也是一样，只不过二者函数位置有着些许不同，所以一样可以采用chen氏算法实现。2D-IDCT的数学表达式为公式(3-17)所示。

$$f(x,y)=\frac{2}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}C(u)C(v)F(u,v)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \quad (3\text{-}17)$$

其中 $f(x,y)$ 是还原的图像像素值， $F(u,v)$ 是2D-DCT的频率系数，$C(u)C(v)$是权重系数，定义如公式(2-3)所示，和2D-DCT一样不去直接计算而是利用行列分解将2D-IDCT分解为两个1D-IDCT，先对行进行1D-IDCT再对列变换。由chen氏算法2D-IDCT可以分解如下式所示。

$$\begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix} = (P+M) \quad (3\text{-}18)$$

$$\begin{bmatrix} f(7) \\ f(6) \\ f(5) \\ f(4) \end{bmatrix} = (P-M) \quad (3\text{-}19)$$

$$P=\begin{bmatrix} c_0 & c_2 & c_0 & 0 \\ c_0 & 0 & -c_0 & -c_2 \\ c_0 & 0 & -c_0 & c_2 \\ c_0 & -c_2 & c_0 & 0 \end{bmatrix}\begin{bmatrix} F(0) \\ F(2) \\ F(4) \\ F(6) \end{bmatrix} \quad (3\text{-}20)$$

$$M=\begin{bmatrix} c_1 & c_1 & c_1 & 0 \\ c_1 & 0 & -c_1 & -c_1 \\ c_1 & -c_1 & 0 & c_1 \\ 0 & -c_1 & c_1 & -c_1 \end{bmatrix}\begin{bmatrix} F(1) \\ F(3) \\ F(5) \\ F(7) \end{bmatrix} \quad (3\text{-}21)$$

其中$[c_0,c_1,c_2]=[1/\sqrt{8},1/\sqrt{6},1/2]$，利用陈氏分解后只需要计算P、M两个矩阵就可以得到1D-IDCT的结果。需要注意的是在计算的时候将系数扩大256倍形成整数以降低运算复杂度。

利用行列分解，整个2D-IDCT模块设计结构如图3-22所示。模块由串并转换模块、1D-IDCT模块、转置RAM、缓存器组成。首先输入的数据进入串并转换模块，将每8个数据即8×8图像块的一行数据一起输出到1D-IDCT模块，完成

1D-IDCT之后数据到达RAM模块，RAM1和RAM2执行乒乓操作不断地接受和处理数据，控制器根据RAM地址用来切换RAM1和RAM2之间的读写状态，将RAM中已经完成行变换的数据按列送到二级1D-IDCT，最终将2D-IDCT的结果输出。

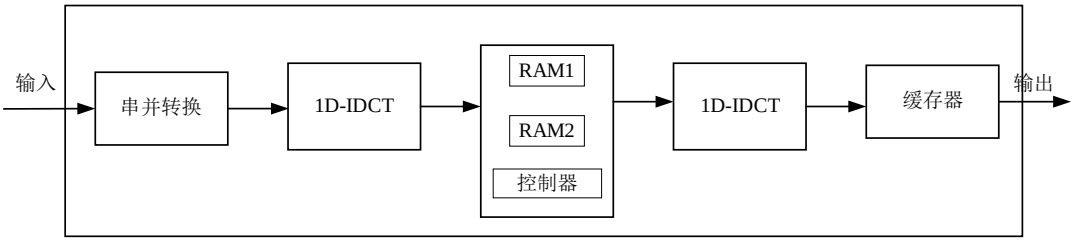



图3-22 2D-IDCT的结构

### 3.5.3 其他模块

关于其他模块如反量化模块、反Z字形扫描模块相对比较简单而且原理和量化以及Z扫描模块相同，因此在这只做简短介绍。

反量化模块：在码字经过熵解码模块之后会输出长度为64的频域系数，将这些频域系数与对应位置的量化表中的步长进行相乘。在反量化模块中，将量化表中的步长存储在RAM中，每输入一个频域系数就从RAM中取出相应的步长，由于量化表中数据全都是2的整数次幂，所以同样可以直接利用移位就可以得到反量化之后的频域系数。

反Z字形扫描模块：和Z扫描的结构和流程一样，只不过此时是将Z字扫描顺序改为反Z字形扫描的顺序。用双口RAM来存储反量化之后的数据，由于在反量化阶段很多数据的值都为0，所以直接将RAM的初始值赋值为0，当开始反Z字形扫描时，把非零值给到正确地址的寄存器当中，而其他寄存器的值仍为零，这样就不再需要单独对零进行存储，加快了反Z字形扫描的处理速度。

## 3.6 本章小结

本章详细地对JPEG的编码各个模块地设计和部分优化设计进行了介绍，包括2D-DCT模块、量化和Z扫描模块、熵编码模块以及码流组装模块。解码部分主要对Huffman解码及2D-IDCT的设计进行说明介绍，并对其中一些模块进行了Matlab和Vivado的仿真验证了其正确性。

# 第 4 章 仿真与系统验证

在完成M-JPEG各模块相关代码设计之后需要对图像编解码的IP核进行仿真测试，以及在硬件平台上验证整个系统的Verilog设计的正确性和实际工作的表现。此章节对编解码IP核的功能进行了仿真与FPGA板载验证。

## 4.1 M-JPEG 编解码 IP 核的功能仿真

对M-JPPEG编解码的IP核搭建仿真测试平台的目的是验证硬件设计是否能按预期工作并评估其性能，更早地发现和解决问题减少在后期修改设计所需要的时间和成本。

### 4.1.1 搭建仿真测试平台

本文分别对M-JPEG编码系统的IP核和M-JPEG解码系统IP核进行了测试台（Testbench，tb）设计来验证生成数据的正确性。在JPEG编码功能正确的前提下M-JPEG编码IP核的测试台产生的数据恰好可以用来当作M-JPEG解码IP测试台的数据输入。因此最先要做的是对M-JPEG编码IP核的测试验证，首先用Matlab对一幅分辨率为1280×720的图片进行RGB数据的生成，并将数据保存在一个名为Image_1280_720的文件中，之后在测试工作台中用readmemh指令来读取RGB数据并以十六进制加载到内存数组中，把它用作tb文件中的输入数据，经过编码后生成的码字再由fopen命令写入到一个指定的文件夹中，最后用软件将压缩后的数据还原成图片并与原图进行对比。M-JPEG的编码测试工作台如图4-1所示。

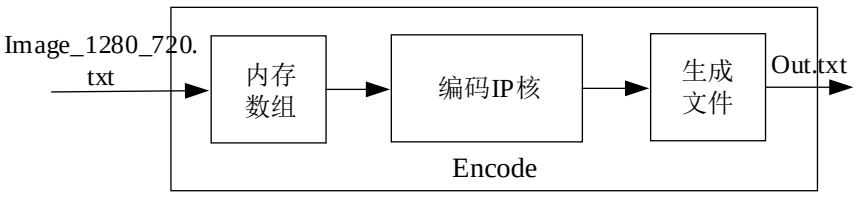



图4-1 编码测试工作台

生成的Out_txt文件不仅可以经过软件转换成.jpg格式的图片来验证编码的正确性，还可以将它用作解码的输入文件来对解码的IP核进行功能仿真，解码IP核测试工作台工作过程与编码测试工作台相同，输入文件经过解码后生成

Out2.txt文件，M-JPEG解码测试工作台如图4-2所示。

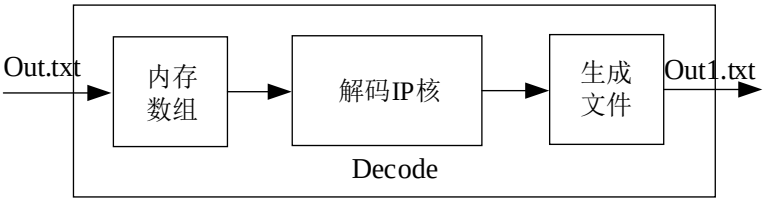



图4-2 解码测试工作台

### 4.1.2 仿真测试结果

为了测试M-JPEG编码IP核的速度，特意选择了一张分辨率为1280×720的图片，利用Matlab软件将图片转换成RGB十六进制数据作为编码工作测试台的输入，将1280×720分辨率的图像数据输入到编码器IP中，此时仿真结果如图4-3所示；end_of_file_signal为一帧图像结束的标志信号，由图4-3中可以看到整张图片编码消耗的时间约为15.111ms，测试系统时钟选的是74.25MHz。可以计算出M-JPEG编码器的IP核对所选取的图片的处理速度达到66.17帧/秒。

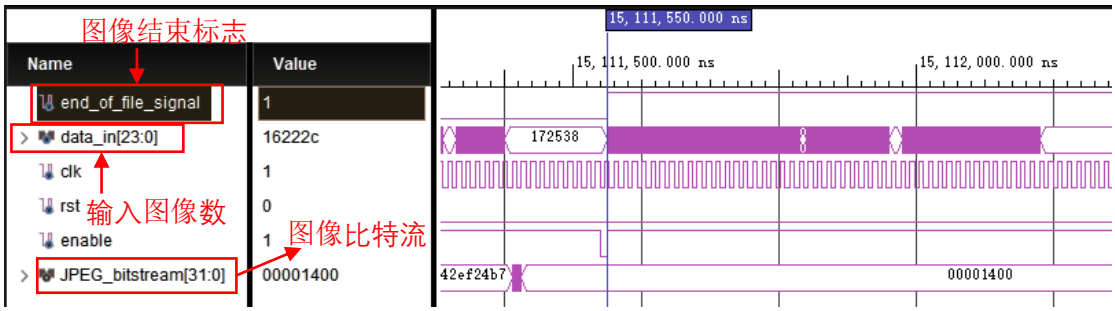
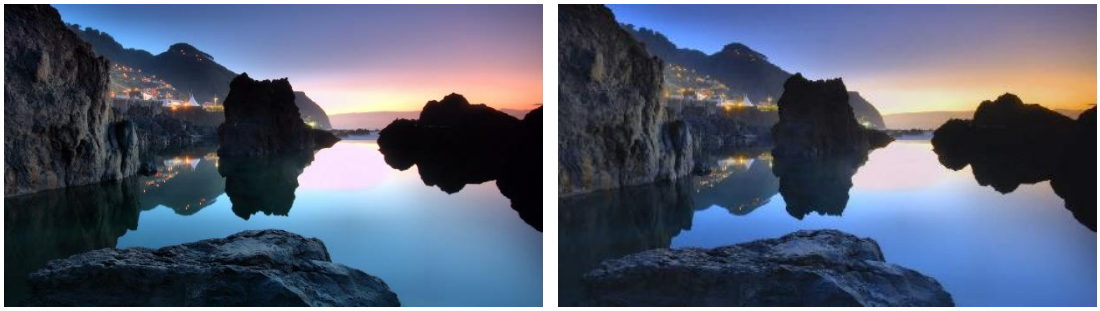



图4-3 编码器的仿真波形结果

对M-JPEG编码IP核的处理能力做完仿真测试以后，接下来要对它编码出的结果的正确性进行验证。编码测试台完成工作后它生成了一个txt文件，使用软件对生成的仿真文件转换成.jpg格式的图片，然后打开其图片看是否能正常显示，并与未实现编码压缩前的图片进行视觉上的对比和文件大小上的对比，对比结果如图4-4所示，其中图a是编码之前的图片，图b是编码后并经过软件还原的图片，对比发现在细节方面经肉眼观察二者并无差异，在色彩上有着细微差距，这是由于在编码时下采样造成的。而在图片大小方面，编码后的图片要比原图缩小了约49倍。



(a)原图　　(b)编码后的图片

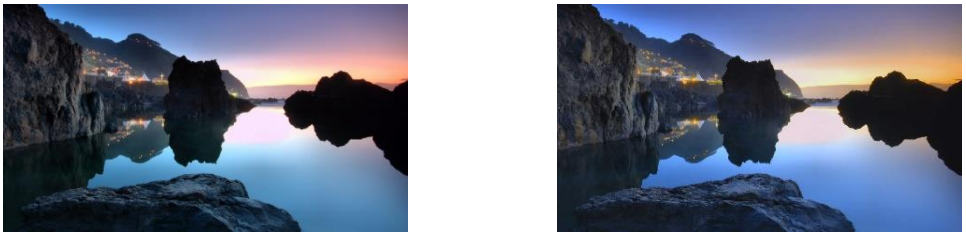
2.63MB　　55.2KB

图4-4 原图与编码后图片之间的对比

在对M-JPEG编码IP核的功能正确性及性能进行验证之后，接下来开始对M-JPEG解码IP核功能和性能开始仿真测试。把编码IP核产生的数据用作解码的输入，图4-5是解码器的仿真波形，可以看到一张图片解码完成的时间约是15.933ms，可以计算出M-JPEG解码器的IP核对所选取的图片的处理速度达到62.76帧/秒。


图4-5 解码器的仿真波形结果

用M-JPEG解码IP核解码后生成的RGB数据还原成的图片如图4-6中右图，视觉效果与原图并无明显差异。



原图　　解码图片

图4-6 解码后图片与原图之间的对比

在对M-JPEG编解码IP的仿真测试中不仅使用了上述图片进行测试，还用其他分辨率的图片也进行了测试仿真，这些图片在编解码IP下的仿真结果都能正确地还原出图片也再一次证明了编解码IP核的功能无误。

## 4.2 基于 FPGA 的验证平台设计

在对M-JEPG编解码IP核的功能和性能进行仿真以后，还需要对其编解码系统在硬件平台上进行验证测试。硬件验证部分基于Xilinx A7系列，具体型号为xc7a100tfgg484的FPGA开发平台，对M-JPEG编解码IP核进行实际的硬件测试，其具体的硬件系统搭建如图4-7所示，整个硬件系统由编码和解码两大部分组成，外界图像经过摄像头采集再转换为RGB格式，并将数据存储到DDR3(Double Date Rate3 SDRAM)中，接着DDR3将数据送到编码IP核中处理，经编码形成的数据流通过以太网发送到解码端，解码端的DDR3接收发送来的数据并将其给到解码IP核核进行解码，最后将DDR3中解码后的数据通过HDMI接口输出在显示器上显示图像。

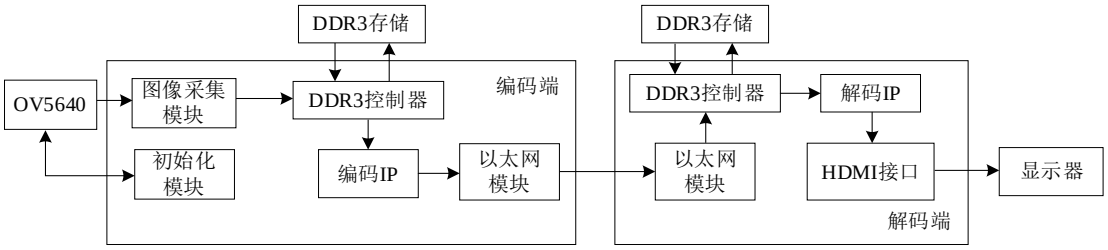



图4-7 整体硬件系统框架

## 4.3 验证所需模块的设计

在Verilog仿真中，M-JPEG编解码IP的输入可以直接从PC端读取文件，输出也可以直接写到PC端的文件。在进行实际电路的验证上，输入的来源和存放都成了问题，因此在实际中还需要加入其它模块的设计。图像编码的输入可以利用摄像头来采集外界图像，对于数据存放问题，少量的数据可以直接放在FPGA的RAM、ROM中，如编解码过程中的量化表、Huffman表等，但是对一张较高分辨率的图像来说所需要的存储数据量是比较大的，所以直接存放在RAM、ROM中就变得不合适了。所以在编码IP核的硬件测试中需要把数据缓存在DDR3中，并将编码后的数据通过以太网接口传输给PC端进行验证编码IP的功能是否正确，解码IP核验证则需要从DDR3中读取编码之后的数据进行解码，通

过HDMI接口传输数据最终在显示器上显示图像。因此，还需要其他模块共同构成整个验证系统。

### 4.3.1 视频采集模块设计

OV5640使用的是两线式SCCB接口总线，SCCB控制器是用于连接应用处理器和图像传感器的组件，主要负责实现应用处理器向图像传感器中的特定寄存器写入数据的功能，以实现对传感器的配置和控制和判断图像处理器的工作状态。本设计采用的是DVP接口[43]。其中包括了时钟信号、行有效信号、场同步信号、8位或更多位的数据信号。图4-8为OV5640在DVP接口模式下输出一行图像数据的时序图。

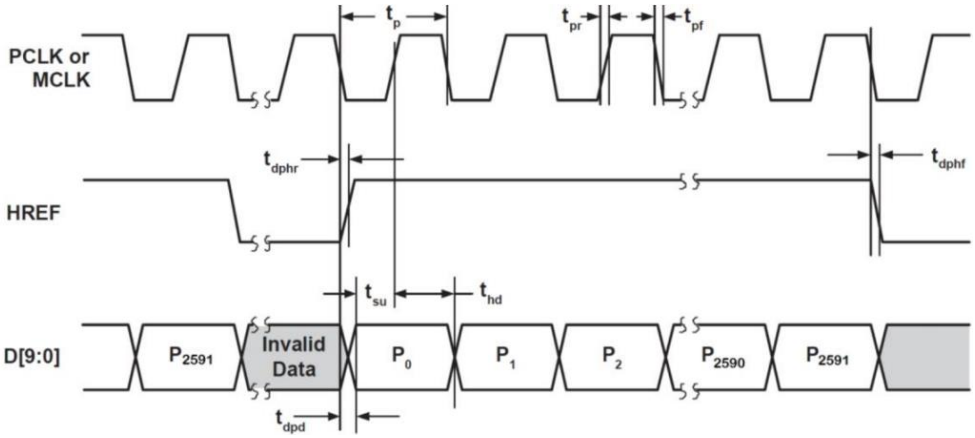



图4-8 OV5640输出时序图

PCLK为像素时钟，HREF是行同步信号，当设置HSYNC信号极性为高有效时，高电平期间输出一行图像。考虑摄像头的性能以及为了更好地平衡图像质量和速度之间的关系，本设计采用的是720p的格式，即1280×720的分辨率，总水平大小为1892，总垂直大小为740，且本设计中帧率要求为60，所以像素时钟为$1892\times740\times60\times2\approx168MHz$。

接着对OV5640模块进行设计，其结构框图如图4-9所示。

由于摄像头OV5640在720p的格式下，一帧图像的数据大小达到了$1280\times720\times16=14745600bit\approx14Mbit$，带宽为$14Mbit\times60Hz=840Mbit/s$。从Xilinx提供的7Series器件手册可以发现，xc7a100t的片内存储资源为4860Kbit，两个芯片都不够存储图像数据。因此只能使用板载的外部存储器DDR3来缓存数据，板载的2片DDR3容量为4096Mbit(1块ddr3的容量为2048Mbit)，2块DDR3带宽为25.6Gbit/s，能够实现图像的存储。

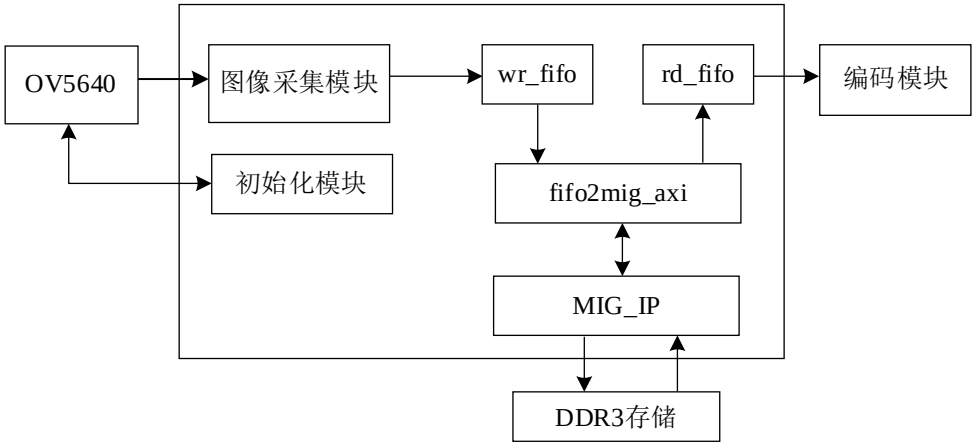



图 4-9 OV5640 模块设计

由图4-9可知顶层模块包括图像采集模块，初始化模块以及DDR3模块。其中的图像采集模块是将DVP接口输出的8位数据转化成RGB565的16位数据，初始化模块是利用I2C控制器对寄存器进行配置，DDR3则是对图像数据进行存储。

在摄像头正常工作之前要先配置寄存器来初始化传感器。由于SCCB接口实现比较简单且兼容I2C总线[44]，因此只需要使用成熟的I2C的控制器配合简单的控制逻辑就可以实现。表4-1是部分寄存器的配置。

表4-1 部分寄存器配置

| 地址 | 寄存器名称 | 数值 | 相关描述 |
|---|---|---|---|
| 0x3808 | TIMING DVPHO | 0x05 | Bit[3:0]：输出水平宽度高4位 |
| 0x3809 | TIMING DVPHO | 0x00 | Bit[7:0]：输出水平宽度低8位 |
| 0x380A | TIMING DVPVO | 0x02 | Bit[2:0]：输出垂直高度高3位 |
| 0x380B | TIMING DVPVO | 0xd0 | Bit[7:0]：输出垂直高度低8位 |
| 0x380C | TIMING HTS | 0x07 | Bit[3:0]：总水平尺寸高4位 |
| 0x380D | TIMING HTS | 0x64 | Bit[7:0]：总水平尺寸低8位 |
| 0x380E | TIMING VTS | 0x02 | Bit[7:0]：总垂直尺寸高8位 |
| 0x380F | TIMING VTS | 0xe4 | Bit[7:0]：总垂直尺寸低8位 |
| 0x4300 | FORMAT CONTROL 00 | 0x61 | Bit[7:4]：数据输出格式<br>Bit[3:0]输出顺序 |
| 0x3035 | SC PLL CONTRL1 | 0x11 | Bit[7:4]：时钟分频器<br>Bit[3:0]：MIPI的比例分频器 |
| 0x3036 | SC PLL CONTRL2 | 0xd2 | Bit[7:0]：PLL乘法器 |

地址0x3808-0x380B的寄存器是用来配置输出图像的水平和垂直方向的像素点数，本设计中需要的图像像素的分辨率为1280×720，用十六进制表示为：0500×02d0，将它们的高位和低位分别配置在相应的寄存器当中即可，同理，

地址为0x380C-0x380F寄存器中配置的数字对应的是总水平尺寸1892（0x0764）总垂直尺寸740（0x02e4）；地址为0x4300的寄存器是用来配置数据的格式和输出顺序，其中的高四位表示数据输出格式，有RAW、YUV、RGB等格式，0x61=0110 0001，高四位是6对应的是RGB565格式，低四位为1对应的是{r[4:0]，g[5:3]}，{g[2:0]，b[4:0]}的顺序；地址0x3035寄存器使用的默认的0x11数值，此时时钟频率为800KHz，地址0x3036寄存器用来配置锁相环乘法器，当取值范围在4-127范围内的时候可以取任意整数，在128-252范围内则只能是偶数，在这里0xd2转为十进制是210，$800KHz \times 210 = 168MHz$，恰好是720p分辨率下所需要的像素时钟大小。

对OV5640完成相关寄存器配置以后开始对外界的图像进行采集，图像采集模块的设计框图如4-10所示。

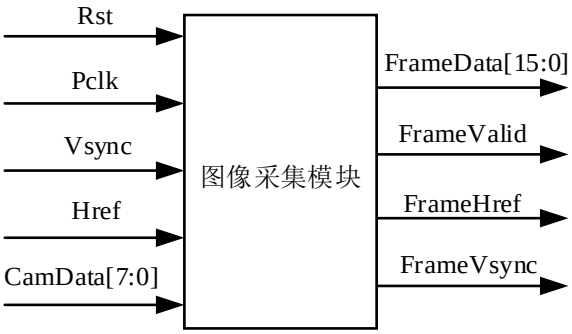



图4-10 图像采集模块各接口

图像采集模块的作用就是实现将输入的两个8位的图像数据拼接成1个16位的数据，在该模块中输出的FrameValid是FrameData数据的有效信号，因为输入的CamData端口需要两个时钟周期才能传输一个RGB565的16位数据，所以在FrameData端口上的数据只有1个时钟周期是有效的。FrameValid在连续的两个周期中只有一个时钟周期是高电平而另外一个周期是低电平，以此来保证在下一级使用FrameData数据时，每两个时钟周期只使用1次。在实际的应用中FrameData在FrameValid信号为高电平的时候，输出的是前两个时钟周期的值拼接成的。图4-11是上板验证并利用集成逻辑分析器(Integrated Logic Analyzer，ILA)抓取一些信号的波形图，图中cmos_frame_href为行同步信号，标志一行数据的开始结束，cmos_frame_vsync是场同步信号，标志一帧图像的开始结束，cmos_frame_data则为输出的图像数据。

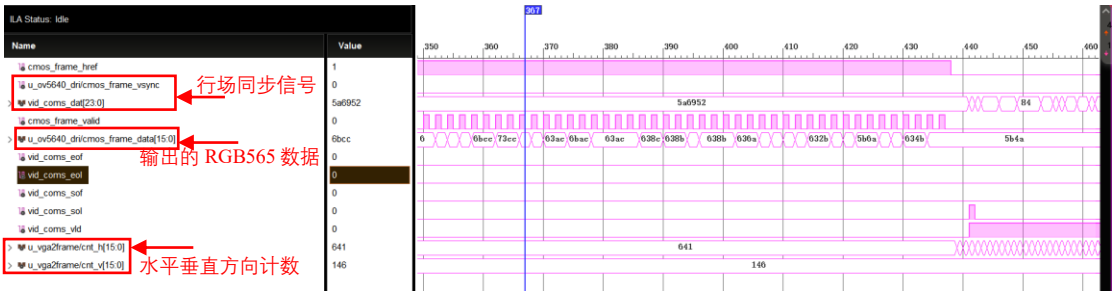



图4-11 OV5640的ILA抓取信号图

### 4.3.2 DDR3 存储模块

DDR3的控制时序比较复杂，如果直接编写代码进行控制会很麻烦，所以在这直接使用的是Xilinx公司的DDR控制器MIG（Memory Interface Generator），它是一种由Xilinx提供的IP核。MIG IP的作用是为FPGA设计师提供一种方便的方式来生成用于连接FPGA和外部存储器（如DDR3/DDR4 SDRAM）的接口逻辑[45]，可以直接使用进行读写操作，降低了开发的难度。下图4-12是7系列的MIG IP的结构框图。

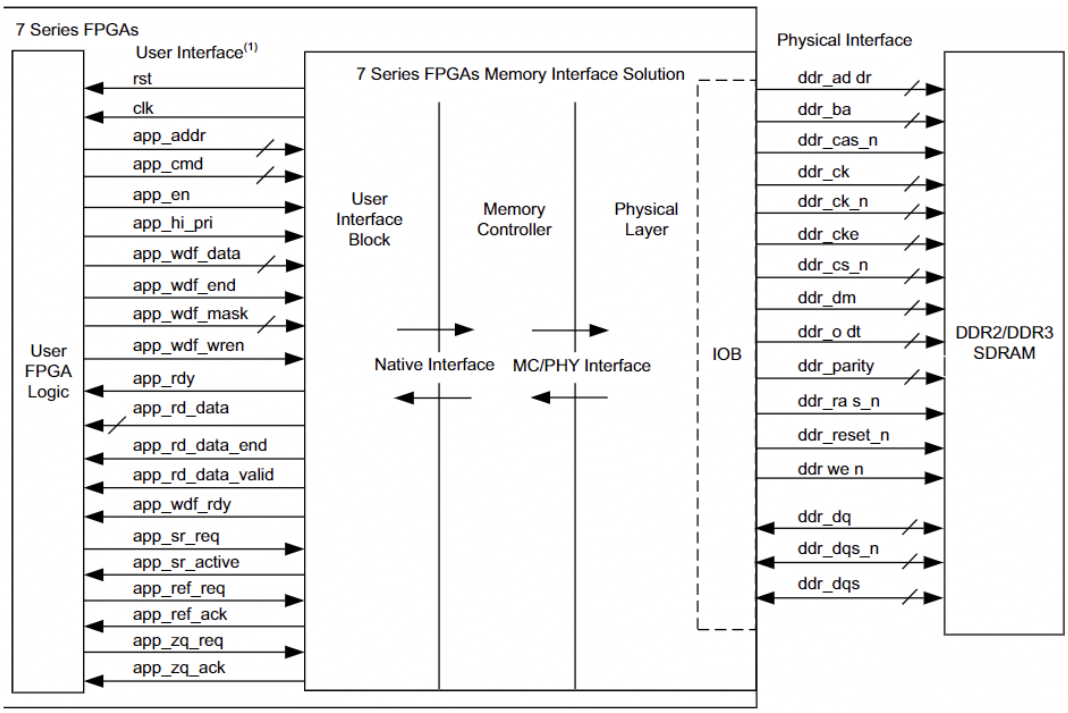



图4-12 MIG IP核的结构框图

其中左边是用户接口，是FPGA与MIG交互的接口，右边是DDR的物理接口，可以借助用户接口端的操作时序来实现对DDR的读写操作。IP核用户接口端的信号及说明如表4-2所示。

表4-2 MIG IP用户接口部分端口信号

| 信号 | 端口说明 |
| --- | --- |
| ui_clk | 用户时钟，MIG IP提供 |
| ui_clk_sync_rst | 复位信号 |
| app_rdy | MIG IP读写命令接收准备 |
| app_en | MIG IP命令写入使能 |
| app_cmd[2:0] | 读写控制命令，读为1写为0 |
| app_addr[26:0] | 用户地址输入 |
| app_wdf_wren | 数据写使能 |
| app_wdf_rdy | 数据接收准备完成 |
| app_wdf_end | 突发写过程的最后一个时钟 |
| app_wdf_data[26:0] | 用户写数据输入 |
| app_rd_data[26:0] | 用户读取 IP 核发来的数据 |
| app_rd_data_valid | 读数据有效信号 |

app_cmd是控制读写的命令，DDR3无论是进行读或者写操作的时候都包含此命令。写命令和读命令的时序相同，以写命令为例，当app_cmd的值等于0的时候进入写命令状态，其时序如图4-13所示，首先观察app_rdy的值，当app_rdy的值为1的时候，说明IP已经做好了可以接收命令的准备，同时拉高app_en使能信号，以及发送命令和地址。接着是写数据的时序图，如图4-14所示。

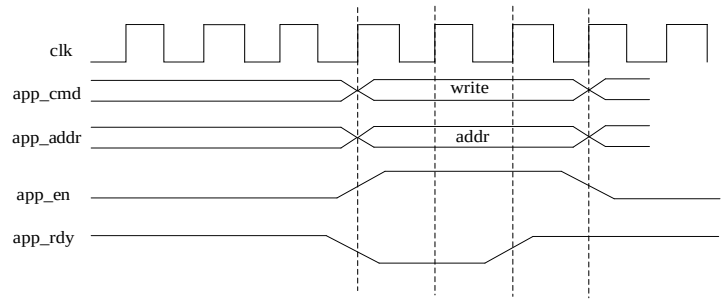



图4-13 写命令时序图

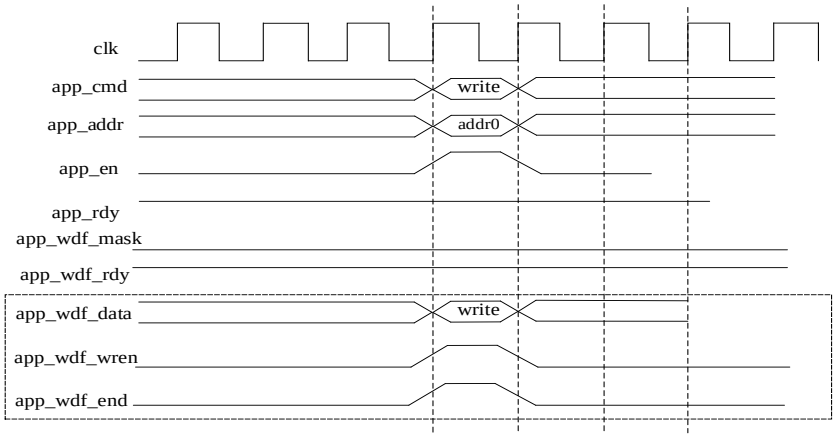



图 4-14 DDR 非背靠背写数据时序图

写数据有三种情况都可以写入，图4-14中虚线标记的部分只是写数据和写命令在同一拍出现，还有两种其他情况，一种是写数据比写命令提前一拍；另一种是写数据比写命令晚两拍；整个写数据的时序过程比较简单，首先看app_wdf_rdy的状态，如果为高电平则IP核已经做好了数据接收的准备，可以接收数据。同时拉高app_wdf_wren信号，并给出写数据app_wdf_data。其中图4-14中的app_wdf_mask信号是屏蔽数据的，当它的信号为高时则屏蔽相应的字节。读数据的过程比写过程还要简单，在发出读命令以后只要有效信号app_rd_data_valid为1就说明总线上数据是有效数据。

这里值得一提的是DDR3的读或者写操作都分为两种状况。第一种是背靠背，就是说无论读或者写在每个周期内都是连贯的。第二种是非背靠背写，它的读写就不再是连贯的，背靠背读写的时序和非背靠背时序完全相同只是传输的数据连续性不同，因此在这不再介绍。一般情况下为充分利用DDR3的性能都使用背靠背的操作。

对整个编解码系统，摄像头采集到的数据是存入到DDR3中，DDR3模块通过MIG IP核控制将图像数据传输到编码IP开始编码，编码完成后的数据再次存储到DDR3中并进行解码操作，最后通过HDMI接口在显示器上显示解码出的图片。所以DDR3在整个系统承载着非常重要的作用，DDR3的设计框图如图4-15所示。

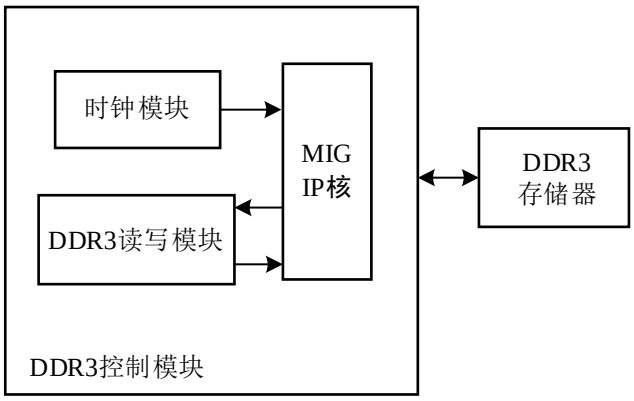



图4-15 DDR3控制模块设计框图

图4-15中DDR3读写模块作用时产生MIG IP用户接口的时序，实现与MIG IP的数据交互；时钟模块是通过锁相环对50MHz输入时钟倍频产生200MHz时钟，以此用来当作MIG IP核的系统时钟以及参考时钟；MIG IP则是与FPGA端进行数据交互以及控制DDR3的读写时序，实现DDR3的读写操作。

接下来对DDR3控制模块进行板载验证，利用ILA来抓取DDR3写的数据，具体的波形图如4-16所示。


图4-16 DDR3写过程波形图

首先判断MIG IP核发送过来的信号app_rdy和app_wdf_rdy，当这两个信号同时为高时拉高app_en和app_wdf_wren，同时给出写命令，即app_cmd为0，此刻正式进行DDR3写过程。写的过程中，写数据app_wdf_data和地址计数wr_addr_cnt在每个时钟自加1，app_addr的数目一次加8。DDR3的读过程和写过程比较相似，首先对MIG IP核发送过来的信号app_rdy进行判断，如果信号为高，在拉高app_en的同时给出读命令，即app_cmd为1，此时开始进行读操作。在进行读操作的时候，app_addr同样每次自加8。图4-17是在ILA上观察到的读信号波形。

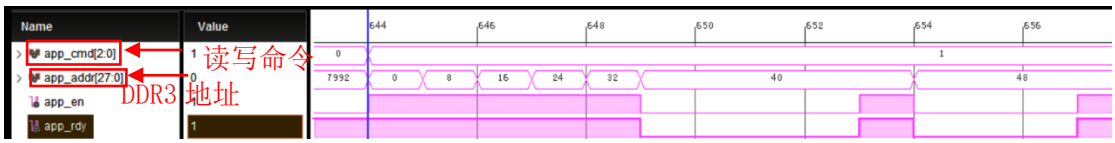



图4-17 DDR3的读时序

### 4.3.3 以太网模块

编码端完成对图像的编码之后要将码流传输到解码端进行图像的还原，由于本设计是对720p格式的图像进行编码处理，且对传输的实时性有较大的要求。尽管UART接口比较简单，但是只适合低速、实时性不强的场景。因此采用的是以太网模块来对数据进行传输，而且设计中只需要点对点的视频传输。UDP具有消耗资源少，效率高等特点非常适合传输音频视频等对实时性要求高的场合，所以本设计采用UDP协议[46]，以太网UDP传输单包数据的格式由图4-18所示。从图中可以看出，以太网的数据包就是对各层协议的逐层封装来实现数据的传输。用户数据打包在UDP协议中，UDP协议又是基于IP协议之上的，IP协议又是走MAC层发送的，它们有着层层递进的包含关系。

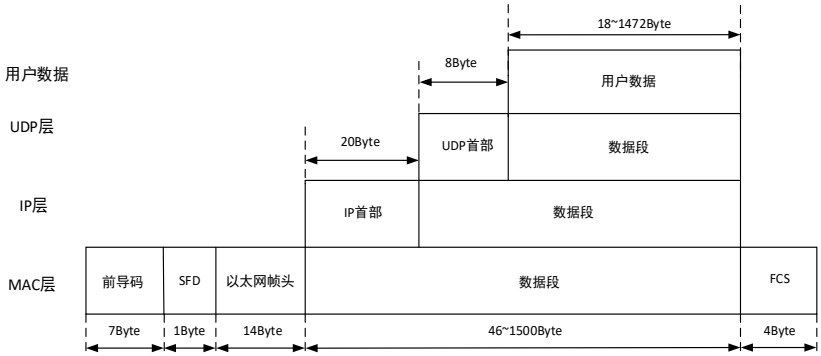



图4-18 以太网UDP传输数据包

对以太网模块进行设计实现，通过PC端向FPGA发送数据，FPGA通过以太网接口接收数据并再将数据发送回去，以此来验证以太网接受和发送模块的正确性，具体的设计框图如4-19所示。

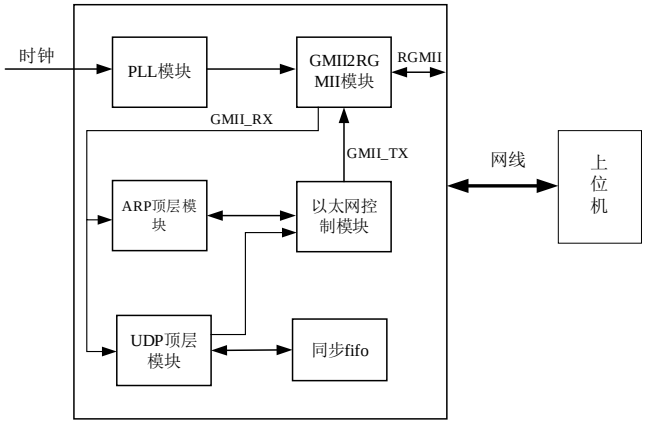



图4-19 以太网UDP设计框图

输入的时钟经PLL之后输出200MHz时钟给到IDELAYCTRL用作参考时钟，GMII2RGMII模块将双沿数据转化为单沿，ARP模块负责获取开发板的MAC地址，以太网控制模块根据接受的协议类型选择GMII的引脚是和ARP顶层模块还是UDP模块连接；UDP顶层模块负责对UDP数据包的接收发送，同步fifo则是为了缓存数据。

图4-20是通过ILK采集到的UDP接收过程的波形图，通过上位机向开发板发送数据，gmii_rx_dv是GMII接口的接收有效信号，gmii_rxd是GMII接收的有效数据，而rec_en是收到数据的有效信号，rec_data是从8位转换成32位的数据，当一个包的数据接收完成后pkt_done拉高。

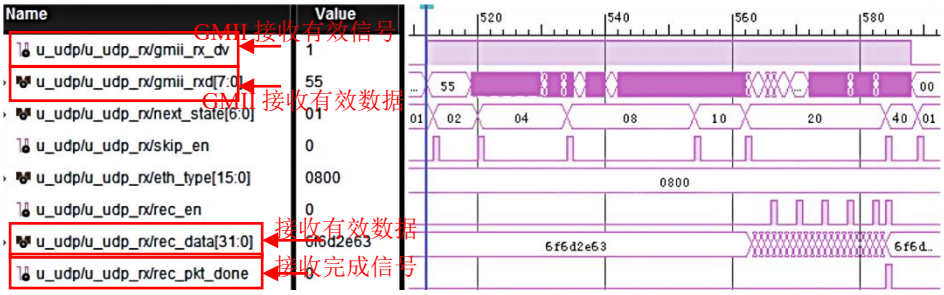



图4-20 UDP接收过程波形图

图4-21为发送过程中ILA抓取的波形图，图中tx_start_en是发送的有效信号，eth_tx_en和eth_tx_data 即为GMII接口的发送接口。在开始发送以太网帧头时crc_en拉高，开始CRC校验的计算，在将要发送有效数据时拉高tx_req（发送数据请求）信号，tx_data即为待发送的有效数据，在所有数据发送完成后输出tx_done（发送完成）信号和crc_clr（CRC校验值复位）信号。

最后通过网口调试助手再进行通信验证，将协议类型选择为UDP协议，地址端口等信息配置无误后，在串口发送端发送MJPEG编解码，可以看到接收端接受的数据可以选择ASCII和十六进制数据，从图4-22看出发送和接收端数据相同。

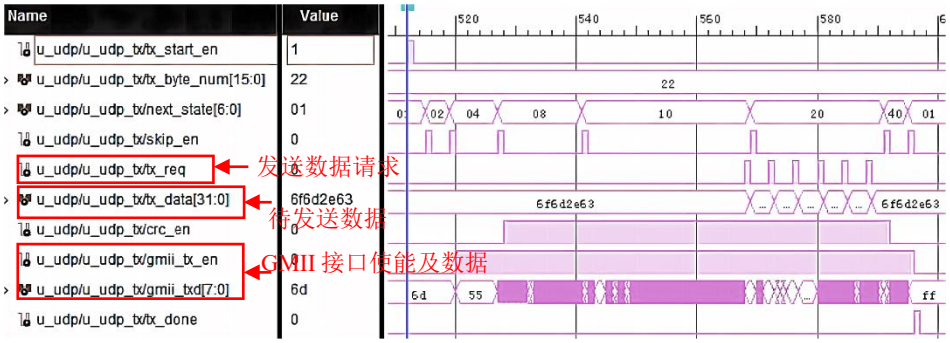



图4-21 UDP发送过程波形图

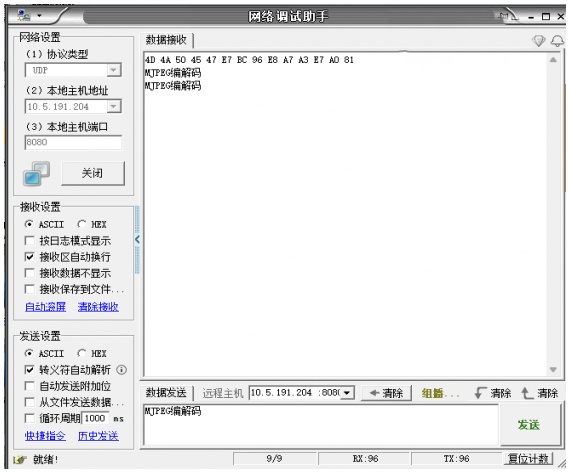



图4-22 网口调试助手界面

### 4.3.4 HDMI 接口模块

HDMI是一种数字音视频接口标准，用于高清晰度影音设备之间的连接。它可以快速传输视频、多声道音频和其他数据，最新发布的HDMI2.1最大的传输速率可达到48Gbps，因此很适合应用于传输高清或无压缩的音频视频信号[47]。

HDMI的接口协议使用的是TMDS的编码方式，图4-23为TMDS传输视频数据的示意图，从图中可以看到发送端将图像的蓝色、绿色、红色数据信息和控制信号被分成了3组，每一组都是由编码器和串行发送器构成的。每个编码器中的输入是由8位的图像数据和2位的控制信号组成。另外还对串行发送时钟进行了输出，作为3个数据/控制通道的同步时钟；串行发送器则是对编码器传输来的10位数据进行串行化处理，这个过程利用了7系列FPGA中专用的硬件资源OSERDESE2（可以实现DDR的双倍传输速率的功能），最后再由TMDS数据通道发送出去。

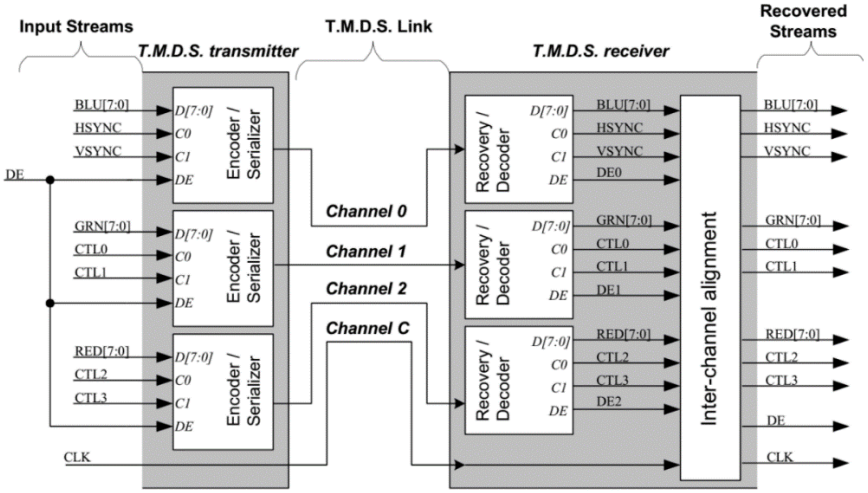



图4-23 TMDS传输通道示意图

由于在本设计中只需要通过HDMI接口显示图像而不需要传输音频信息，所以可以直接把其当成DVI接口来驱动。HDMI顶层模块主要包括显示、驱动以及rgb2dvi模块，其具体的设计框图如4-24所示。

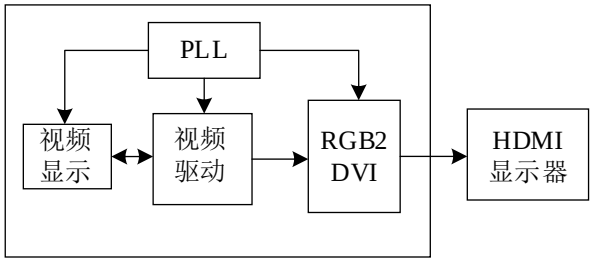



图4-24 HDMI接口显示框图

由于需要10:1转换率的并转串过程，所以需要系统中PLL模块生成各模块74.25MHz的像素时钟信号以及用作TMDS串行编码的371.25MHz的发送时钟，视频显示模块产生RGB格式的图像信息；视频驱动模块输出视频信号、行场同步信号以及视频有效信号；而RGB2DVI模块是将输入进来的图像、控制信号转换TMDS串行数据输出到HDMI接口上。

对于DVI接口而言，只需要将编码器和发送器例化并将图像的数据和控制信号按照DVI规范连接在一起就可以，RGB2DVI的模块设计框图如图4-25所示；

图4-25中看到编码器将像素时钟和图像数据以及控制信号一起发送，而且这里的时钟和像素时钟是相同的，也就是说tmds_clk并不是数据的位同步时钟而是一个完整编码数据的同步时钟。

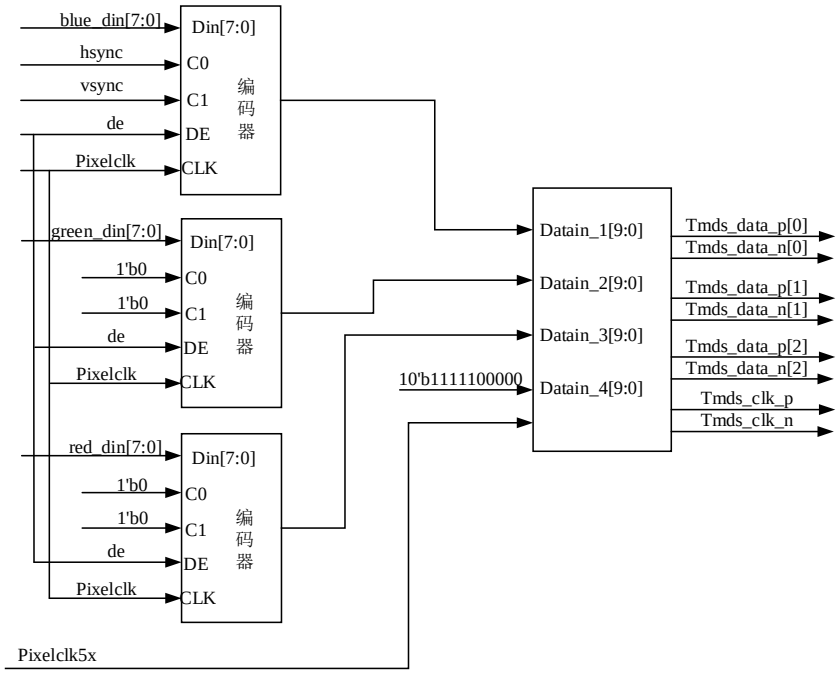



图4-25 GB2DVI设计框图

图4-26上板使用ILA抓取到的部分信号的波形图，其中video_hs、video_vs、video_de分别是行同步信号、场同步信号以及数据使能信号，pixel_x、pixel_y、pixel_data是像素点的横纵坐标以及像素数据。

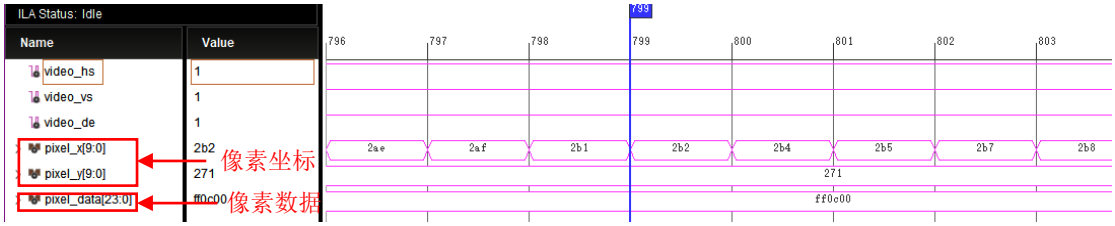



图4-26 HDMI模块波形图

## 4.4 静态及动态图像的板载验证

将编解码的Verilog设计代码经过综合、布局布线之后生成Bit流并将其下载到FPGA中，PC端通过以太网接口向开发板的DDR3内存中传输图片，数据传输结束之后，编码模块读取DDR3中的图片对其进行图像大小压缩，解码完成后图像数据经HDMI接口传至显示器显示解码后的图像，经过编解码最终形成的图像如图4-27所示，其中左侧显示屏显示的是解码后的数据图像，右边为原图。

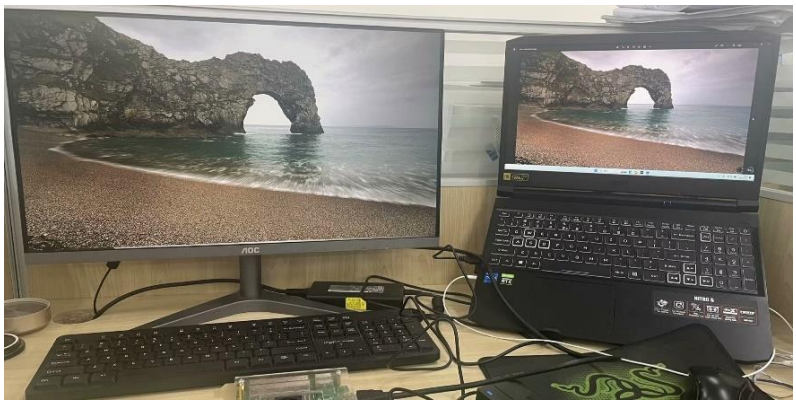


图4-27 图像的板载验证

经过静态的图像验证以后说明整个编解码系统的基本功能是没问题的，但是由于输入的只是一张图片数据，所以只能保证该系统对于一张图片或者几帧图像来说是无误的，并不能说明对连续帧的图像还能保持正确性，因此还需要用动态的图像进行验证。输入的视频图像数据来自前文介绍的OV5640摄像头，输入的视频数据经过FPGA编解码后通过HDMI接口将视频图像显示在屏幕中。最终结果如图4-28所示。

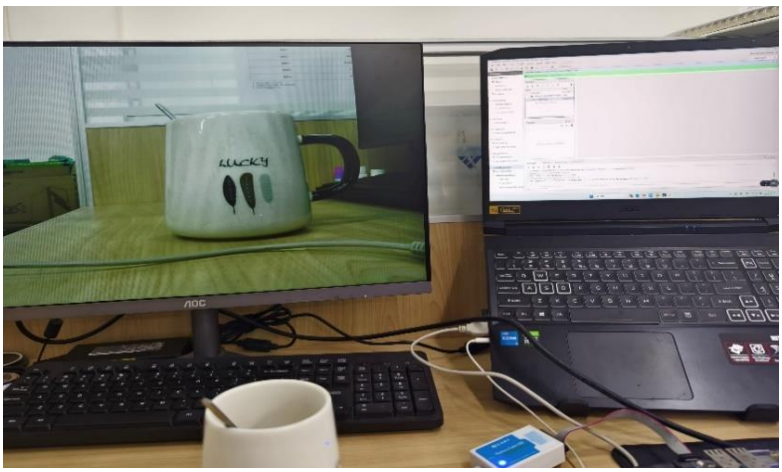



图4-28 图像板载验证

### 4.4.1 实验结果

在Vivado中经过综合之后，可以得到M-JPEG编解码的IP核的资源消耗情况，具体的数值如表4-3、表4-4所示。

表4-3 M-JPEG编码IP核的资源消耗表

| 项目 | 已用资源 | 可用资源 | 资源占用比 |
|---|---|---|---|
| LUT | 6390 | 63400 | 10.07% |
| FF | 7606 | 126800 | 6.00% |
| BRAM | 34 | 135 | 25.19% |
| DSP | 18 | 240 | 7.50% |
| IO | 91 | 285 | 31.93% |

表4-4 M-JPEG解码IP核的资源消耗表

| 项目 | 已用资源 | 可用资源 | 资源占用比 |
|---|---|---|---|
| LUT | 6434 | 63400 | 10.15% |
| FF | 7916 | 126800 | 6.24% |
| BRAM | 42 | 135 | 31.11% |
| DSP | 21 | 240 | 8.75% |
| IO | 96 | 285 | 33.68% |

为了体现本设计减少了逻辑资源消耗的同时拥有较高的性能，又将本文的MJPEG编解码的IP核与其他参考文献的编解码器进行了一些对比，具体对比结果如表4-5、表4-6所示。

表4-5 本设计与其他编码器之间的对比

| 项目 | 文献[48] | 文献[17] | 文献[49] | 文献[24] | 本设计 |
|---|---|---|---|---|---|
| 平台 | CPU | GPU | FPGA | FPGA | FPGA |
| 逻辑单元 | N/A | N/A | 11466 | 4282 | 1598 |
| 系统时钟 | 2.93GHz | 1.4GHz | 150MHz | 148.5MHz | 74.25MHz |
| 图片分辨率 | 2000×1502 | 1960×2048 | 640×480 | 1920×1080 | 1280×720 |
| 帧率（fps） | 19.6 | 8.43 | 162 | 61.99 | 66.17 |
| 吞吐率(Mbit/s) | 141 | 270 | 1195 | 3084 | 1462 |

表4-6 本设计与其他解码器之间的对比

| 项目 | 文献[50] | 文献[51] | 文献[52] | 文献[24] | 本设计 |
|---|---|---|---|---|---|
| 平台 | CPU | GPU | FPGA | FPGA | FPGA |
| 逻辑单元 | N/A | N/A | N/A | 5961 | 1609 |
| 系统时钟 | 50MHz | 575MHz | 100MHz | 148.5MHz | 74.25MHz |
| 图片分辨率 | 320×256 | 1920×1080 | ------ | 1920×1080 | 1280×720 |
| 帧率（fps） | 1.2 | 5.35 | 25 | 52.60 | 66.17 |
| 吞吐率（Mbit/s） | ----- | ----- | 1244 | 2644 | 1387 |

由表4-5以及表4-6可以看到，在FPGA平台上，不论是图像编码器还是图像解码器在性能方面都要优于表格中基于CPU、GPU软件平台的图像编解码器。将本设计的编解码IP与其他FPGA平台的编解码器进行对比，与文献[49]中的编码器相比，本设计的编码IP核有着更低的资源消耗以及更大的吞吐率。文献[24]在吞吐率方面约为本设计的2.11倍，但是在逻辑资源消耗上却是本设计的2.68倍，所以在资源和性能之间的平衡上也有着更好的表现。其次再与其他FPGA平台的解码器进行对比，文献[52]的帧率和吞吐率方面没有本文表现好，文献[24]中解码器的资源消耗大约是本设计的3.70倍，其吞吐率表现是本文的1.91倍，因此和编码器一样，本文在占有的资源方面有着一些优势。综上可知本文的M-JPEG视频编解码IP有着不错的性能表现，在资源消耗方面有着较好的水平，能够实现对分辨率为1280×720视频编解码，且帧率能达到60以上。

## 4.5 本章小结

本章节对MJPEG视频编解码的IP核进行了功能仿真，测试了编解码IP的处理速度；接着介绍整个FPGA验证平台，并对平台所需其他的模块进行设计验证，通过搭建的FPGA平台对静态和动态图片进行编解码验证，最后将设计的MJPEG编解码IP与其他文献的编解码器进行对比分析。

# 第 5 章 总结与展望

## 5.1 全文总结

本文对M-JPEG的压缩标准和实现方法进行了深入研究，采用自顶向下的设计方法在Vivado中使用Verilog HDL语言对整个编解码进行硬件电路设计以及部分模块的优化设计，最终对M-JPEG的IP核进行了仿真测试，并在FPGA硬件平台上进行了M-JPEG视频编解码的实际电路验证。

在本文设计研究中，期间主要完成的工作主要有以下几点：

（1）深入研究调查国内外对M-JPEG编解码的研究方向和成果，对M-JPEG的编解码原理以及流程进行了学习和介绍，为后续的整体方案和优化设计提供了坚实的理论基础。

（2）针对编解码模块中的二维离散余弦变换和量化模块的优化做了相关介绍说明，在硬件上进行了测试验证。

（3）对编解码的各模块采用Verilog HDL语言进行硬件电路的设计，详细地对RGB2YUV、下采样、二维离散余弦变换、量化、Z扫描、熵编码、码流组织模块以及解码中的部分模块进行阐述和设计，并利用Matlab与一些仿真测试的结果进行对比，确保实验数据的正确性。

（4）确定了整个M-JPEG编解码的硬件平台的搭建方案，对实际电路验证所需的其他模块，如OV5640、DDR3、以太网、HDMI等模块的设计和上板验证，并在Vivado中利用ILA观测分析端口信号数据。

（5）对设计的M-JPEG编解码的IP核在Vivado中实现了仿真测试并分析其正确性和性能，最后在FPGA平台上对实际硬件电路进行验证和实验对比。

## 5.2 未来工作展望

本文中M-JPEG编解码的IP核虽然能够实现对分辨率为1280×720视频的编解码工作，但是由于时间关系和个人能力有限，M-JPEG视频编解码的IP核还有一些不足之处，需要在今后的工作中加以改进和完善。

（1）在资源消耗和图像处理速度上，本文在一些模块设计中直接调用了乘法器，这对整个系统的速度和时序方面有着一些影响，后续可以考虑使用硬件

加速器、优化算法等方法来提高整个系统的性能表现。

（2）本设计采用的是有损的JPEG压缩算法，而且在下采样中采取的是4:2:2的采样率，在一些对图像质量要求高的场所，如在医学影像、卫星图像等领域可能不太实用，可以尝试JPEG2000的无损算法对图像进行压缩。

（3）M-JPEG编解码IP核的处理速度和输入图片复杂度有着很大的关系，本设计中测试的图像样本有限，在以后的工作中还可以针对一些复杂图像继续进行测试验证。

# 参考文献

[1] The-Anh,Pham, Delalandre. Effective Decompression of JPEG Document Images [J]. IEEE Transactions on Image Processing, 2016, 25(8): 3655-3670.

[2] 杜玉远, 闫爱云. 基于FPGA的视频编码系统设计与实现[J]. 集成电路应用, 2022, 39(12): 1-3.

[3] 李浙伟. 智能时代 视频编解码技术的发展及行业应用[J]. 中国安防, 2022, 17(7): 86-92.

[4] Kang Y, Xu X. A System and Its Implementation Based on FPGA for Video JPEG 2000 Codec and Network Transmission[C]. Proceedings of the 2021 5th International Conference on Digital Signal Processing, 2021: 260-265.

[5] Wang Q, Cheng Z, Pan X,et al. Rate Control Technology in Video Coding[C]. International Conference on Signal and Information Processing, Networking and Computers, 2020: 889-895.

[6] 邵文泽, 韦志辉. 压缩感知基本理论:回顾与展望[J]. 中国图象图形学报,2012, 17(01): 1-12.

[7] Deshpande R G, Ragha L L. Performance analysis of various video compression standards[J]. IEEE, 2016: 148-151.

[8] Wallace G K. The JPEG still picture transmission standard[J]. IEEE Transaction on Consumer Electronics, 1992, 38(1): 18-34.

[9] 陈双燕. 基于FPGA的JPEG编码器设计[J]. 装备制造技术,2016, 23(8): 252-254+272.

[10] 刘海波, 沈晶, 郭耸. Visual C++数字图像处理技术详解[M]. 机械工业出版社, 2010: 903.

[11] Taubman D S, Marcellin M W, Rabbani M. JPEG2000: Image compression fundamentals, standards and practice[J]. Journal of Electronic Imaging, 2002, 11(2): 286-287.

[12] 宋鸿梅, 徐学庆, 牟海维, 等. 图像无损压缩算法JPEG-LS实现及性能研究[J]. 光学仪器, 2014, 36(4): 315-318+322.

[13] Dziembowski A , Mieloch D , Jeong J Y, et al. Immersive Video Postprocessing for Efficient Video Coding[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2021, 109(9): 1521-1536.

[14] A. Cerveira, L. Agostini, B. Zatt, et al. Memory Profiling of H.266 Versatile Vide

o Coding Standard[C]. 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2020: 1-4.

[15] 邓永红. 视频压缩编解码标准综述[J]. 有线电视技术, 2004, 11(3): 1-6.

[16] 赵立歧. 基于FPGA的MJPEG编码IC设计[D]. 山东大学, 2011.

[17] Shatnawi M K A, Shatnawi H A. A performance model of fast 2D-DCT parallel J PEG encoding using CUDA GPU and SMP-architecture[C]. 2014 IEEE High Per formance Extreme Computing Conference (HPEC). 2014: 1-6.

[18] Wong, Thomas. BiCMOS ASICs: technology and applications. Third Annual IEE E Proceedings on ASIC Seminar and Exhibit. IEEE, 1990: T/12.1-T/12.3.

[19] Mittal N. An Introduction to FPGAs From: Flip-Flops to Applications[J]. Circuit cellar, 2023, 21(390): 14-17.

[20] 包宋建, 孟杨, 许艳英, 等. 基于XC2S600E的MJPEG编码器研究与实现[J]. 重庆文理学院学报(自然科学版), 2011, 30(04): 42-45.

[21] 汪洲. 改进的JPEG图像二维DCT编解码技术研究[D]. 江西理工大学, 2019.

[22] 张雅媛, 孔令罔. 一种基于改进量化表的JPEG图像压缩算法[J]. 包装工程, 20 16,37(13): 189-194.

[23] Chang Y W, Truong T K, Chang Y. Direct mapping architecture for JPEG Huffm an decoder[J]. IEE Proceedings - Communications, 2006, 153(3): 333-340.

[24] 邱操. 高性能JPEG编解码系统的研究与设计[D]. 南京航空航天大学, 2019.

[25] Teja R, Sruthi S, Tomar K S, et al. Verilog implementation of fully pipelined and multiplierless 2D DCT/IDCT JPEG architecture[C]. 2015 Online International Co nference on Green Engineering and Technologies (IC-GET), IEEE, 2016: 1-5.

[26] Shichao Y, Zhizhong H, Xin C. A scalable multi-pipeline JPEG encoding architec ture[C]. 2016 28th International Conference on Microelectronics (ICM), 2016: 36 9-372.

[27] 张赛萍. 视频编码优化关键技术研究[D]. 西安电子科技大学, 2022.

[28] Kitsos, Paris, et al. A high speed FPGA implementation of the 2D DCT for ultra high definition video coding[C]. 2013 18th international conference on digital sig nal processing (DSP). IEEE, 2013: 1-5.

[29] Prathibha E. RGB to YCbCr Color Conversion using VHDL approach[J]. 2012, 1(3): 15-22.

[30] Arya V, Singh J. Image Compression Algorithm Using Two Dimensional Discret e Cosine Transform[J]. Imperial Journal of Interdisciplinary Research (IJIR), 201

6, 2(8):199-203.

[31] 何志应. 高效图像编码及后处理算法研究[D]. 电子科技大学, 2018.

[32] Chen H, Sun M, Steinbach E. Compression of Bayer-Pattern Video Sequences U sing Adjusted Chroma Subsampling[J]. IEEE Transactions on Circuits & Systems for Video Technology, 2009, 19(12):1891-1896.

[33] Haweel T I. A new square wave transform based on the DCT[J]. Signal Processin g, 2001, 81(11):2309-2319.

[34] N. J. Higham, Chapter 1: Theory of Matrix Functions, Functions of Matrices. So c. for Industrial and Appl. 2008: 1-29.

[35] Chen X, Dai Q, Li C. A fast algorithm for computing multidimensional DCT on certain small sizes[J]. Signal Processing IEEE Transactions on, 2003, 51(1): 213-220.

[36] Chung R L, Chen C W, Chen C A,et al. VLSI Implementation of a Cost-Efficient Loeffler DCT Algorithm with Recursive CORDIC for DCT-Based Encoder[J]. El ectronics, 2021, 10(7): 862.

[37] 吝毅. 基于FPGA的实时图像预处理系统研究[D]. 长安大学, 2023.

[38] 施宇. 基于FPGA的视频采集与显示系统的设计和验证[D]. 西安电子科技大学, 2022.

[39] 盛海翔. 基于多级并行的JPEG二次压缩图片解码研究与实现[D]. 杭州电子科技大学, 2023.

[40] Sharmila K, Samy K K. An Efficient Image Compression Method using DCT, Fr actal and Run Length Encoding Techniques[J]. International Journal of Engineeri ng Trends and Technology, 2014, 13(6): 287-290.

[41] Sharma M. Compression Using Huffman Coding[J]. International journal of com puter science and network security, 2010, 10(5): 133-141.

[42] Wiggins, Richard H, H. Christian Davidson, et al. Image file formats: past, prese nt, and future, 2001, 21(3): 789-798.

[43] 刘威宏. 基于Zynq的实时视频监测系统的设计[D]. 中北大学, 2023.

[44] 郭常青. 基于FPGA的实时图像预处理技术研究与实现[D]. 中北大学, 2022.

[45] 张刚, 贾建超, 赵龙. 基于FPGA的DDR3 SDRAM控制器设计及实现[J]. 电子科技, 2014, 27(01): 70-73.

[46] 郝香宇. 以太网物理层编解码电路研究与设计[D]. 电子科技大学, 2022.

[47] 孙海洲. 基于FPGA的8K超高清HDMI接口视频信号源的实现研究[D]. 青岛大学, 2023.

[48] Castells-Rufas D, Joven J, Carrabina J. Scalability of a Parallel JPEG Encoder on Shared Memory Architectures[C]. International Conference on Parallel Processing.IEEE Computer Society, 2010: 502-507.

[49] Corvino R, Diken E, Gamatie A, et al. Transformation-Based Exploration of Data Parallel Architecture for Customizable Hardware: A JPEG Encoder Case Study [C]. Euromicro Conference on Digital System Design.IEEE Computer Society, 2012: 774-781.

[50] 高世明, 孟令军, 李宝刚, 等. 基于NiosII多核处理器的JPEG解码的设计与实现[J]. 电视技术, 2011, 35(5): 42-44.

[51] Yan K, Shan J, Yang E. CUDA-based acceleration of the JPEG decoder[C]. International Conference on Natural Computation. 2013: 1319-1323.

[52] 张志晓, 何明华. JPEG解码器IP核的设计与实现[J]. 电子科技, 2011,24(04):59-63.

# 攻读硕士期间取得学术成果

[1] Xing. Zhang, Xiaoqiang. Zhang, Xinxing. Zheng and Mingyu. Xu, "A low complexity JPEG coding system," 2024 4th International Conference on Neural Networks, Information and Communication (NNICE), Guangzhou, China, 2024, pp. 53-56.

[2] 参与项目：纵向项目，安徽省车载显示集成系统工程研究中心开放基金项目（工程类），高性能无线视频安全传输芯片关键技术研究及产业化，项目编号：VDIS2023A01, 2023.12-2025.11