分类号：TN492　　学校代码：10363
密　级：公开　　学　　号：2210330165

安徽工程大学
Anhui Polytechnic University

硕士学位论文

题目　基于 FPGA 实时视频缩放系统的研究

论 文 作 者　完海
指 导 教 师　张肖强
学科（专业）　电子信息
研 究 方 向　人工智能与系统集成

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：2210330165

题　目 基于 FPGA 实时视频缩放系统的研究

题　目 Research on Real-time Video Scale System Based on FPGA

学生姓名：完海

指导教师：张肖强

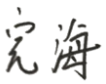
专　　业：电子信息

研究方向：人工智能与系统集成

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

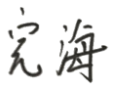
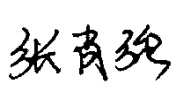
安徽工程大学学位论文版权使用授权书

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

基于FPGA的实时视频缩放系统的研究

摘 要

随着数字视频图像技术的快速发展，实时视频处理系统对于性能需求越来越高。视频缩放技术是视频处理中重要技术之一，对实时性能有着更高的要求，由于传统基于PC端软件的视频缩放算法在性能上存在瓶颈，比如实时显示能力差、内存带宽的限制以及资源利用不充分等问题，而基于现场可编程门阵列（FPGA, Field-Programmable Gate Array）的硬件加速技术能够提供更高的计算效率和实时性，因此FPGA成为了实现高效视频缩放的重要技术。

本文主要研究实时处理视频缩放的技术，研究基于FPGA改进双线性算法架构的视频缩放，主要内容如下：

（1）对不同图像算法的研究和学习，主要研究最邻近算法，双线性算法以及双三次算法，通过Matlab仿真软件对三种相同分辨率的不同图片进行仿真实验，仿真结果通过主观和客观的图像质量评价标准进行比对，最后，权衡了不同算法的运算复杂度和仿真结果的图像质量，选择了双线性算法实现实时视频缩放系统。

（2）提出了均值双线性插值算法及其架构。首先，对输入图像数据进行缓存，获取插值算法所需的四个邻近像素点数据，然后对双线性缩放算法邻近的四个像素点进行均值处理，即加权系数取为1，最后，通过移位的方法取得目标像素值，该方法在硬件设计上取消了乘法器的使用，最后以移位的方法获得目标像素值，提高了算法运算速率，通过Verilog硬件描述语言设计双线性算法模块改进和双线性算法模块，通过综合后的资源消耗情况可以看出，查找表

（LUT, Look-Up Table）和触发器（FF, Flip-Flop）资源的使用降低了40%和30%。

（3）根据改进双线性算法模块的测试效果，搭建了基于改进双线性算法的视频缩放系统，验证了该算法的正确性，对视频缩放系统中各个模块通过内置的逻辑分析仪（ILA, Integrated Logic Analyzer）进行仿真验证，最终实现了板级验证，能够实现从分辨率640×480到1920×1080之间的缩放，验证了整个缩放系统的性能，最后输出的图像没有出现失真情况，可以满足人们日常的需要。图像缩放模块的图像处理帧率为60fps，能够满足实时处理视频图像缩放。

关键词：视频缩放技术，双线性插值，FPGA，视频缩放系统，仿真验证

Research on Real-time Video Scale System Based on FPGA

ABSTRACT

With the rapid development of digital video image technology, the real-time video processing system has higher and higher performance requirements. Video scaling technology is one of the important technologies in video processing, which has higher requirements for real-time performance. Due to the traditional PC-based software video scaling algorithms, there are bottlenecks in performance, such as poor real-time display capability, memory bandwidth limitations, and underutilization of resources, etc., and the hardware acceleration technology based on FPGA (Field-Programmable Gate Array) can provide higher computational efficiency and real-time performance, so FPGA has become the most effective way to achieve high efficiency video scaling. Array (FPGA, Field-Programmable Gate Array) based hardware acceleration technology can provide higher computational efficiency and real-time performance, so FPGA has become an important technology to realize efficient video scaling.

This paper mainly studies the technology of real-time video scaling and video scaling based on FPGA improved bilinear algorithm architecture. The main contents are as follows:

(1) The research and study of different image algorithms, mainly study the nearest neighbor algorithm, bilinear algorithm and bicubic algorithm, through the MATLAB simulation software for three different pictures of the same resolution simulation experiments, the simulation results through the subjective and objective evaluation criteria of the quality of the image to compare, and finally, weighed the different

algorithms of the arithmetic complexity and the simulation results of the quality of the image, the bilinear algorithm was chosen to implement the real-time video scaling system.

(2) The mean bilinear interpolation algorithm and its architecture are proposed. First, the input image data is cached to obtain the data of four neighboring pixel points required by the interpolation algorithm, then the four pixel points neighboring the bilinear scaling algorithm are averaged, i.e., the weighting coefficients are taken to be 1, and finally the target pixel value is obtained by the shifting method, which eliminates the use of the multiplier in the hardware design, and finally obtains the target pixel value in the shifted way, which improves the algorithm computation rate , by designing bilinear algorithm module improvement and bilinear algorithm module through Verilog hardware description language, it can be seen by the resource consumption after synthesizing that the use of resources of look-up table (LUT, Look-Up Table) and flip-flop (FF, Flip-Flop) is reduced by 40% and 30%.

(3) According to the test effect of the improved bilinear algorithm module, a video scaling system based on the improved bilinear algorithm is built, the correctness of the algorithm is verified, and each module in the video scaling system is simulated and verified by the built-in logic analyzer (ILA, Integrated Logic Analyzer), and the board-level verification is finally realized, which can realize the scaling from the resolution of 640×480 to 1920×1080, and the performance of the whole scaling system is verified, and the final output image does not have the situation of distortion, and it can satisfy the people's daily needs. The image processing frame rate of the image scaling module is 60 frames, which can meet the real-time processing of video image scaling.

Supervised by Associate Professor Xiaoqiang Zhang

KEY WORDS: video scaling technology, bilinear interpolation, FPGA, video scaling system, simulation verification

目 录

第1章 绪论

1.1 研究背景与意义

随着数字媒体技术的快速发展，视频已成为人们日常生活中不可或缺的一部分，广泛应用于娱乐、教育、监控、社交等多个领域。视频内容的质量直接影响用户体验，其中，视频分辨率是一个关键因素。视频缩放技术[1]，包括视频的上采样（放大）和下采样（缩小），是视频预处理、传输和显示中的重要环节，它旨在改变图像的大小，以满足不同的应用需求，例如在网络上传输时减少带宽消耗、适应不同分辨率的显示设备、或者进行图像编辑和美化等。图像缩放技术的研究与发展，不仅对于图像处理理论的深化有重要意义，而且对于促进相关应用领域的技术进步和经济发展具有实际的应用价值。

FPGA现场可编程门阵列是一种特殊的可编程逻辑芯片，在FPGA之前，数字电路的设计主要依靠定制的ASIC（专用集成电路）和通用的逻辑器件（如TTL和CMOS逻辑门）[2]。虽然ASIC性能高，但设计周期长、成本高，不适合低成本或小批量生产的项目。而通用逻辑器件虽然灵活，但设计复杂度高、集成度低、性能有限。FPGA的出现提供了一种高度灵活且性能较高的中间解决方案，可以快速实现复杂的数字逻辑设计，同时具备一定的可重配置性，满足了市场对于快速开发和定制化产品的需求。随着计算机技术和应用的发展，尤其是在图形处理、数字信号处理、通信系统和数据中心等领域，对计算性能的需求急剧增加。FPGA能够提供并行处理能力，适合执行复杂的算法和数据密集型任务，因此成为提升计算性能和效率的重要工具。嵌入式系统和物联网的发展要求硬件具有高度的集成性、低功耗和灵活性。FPGA因其可编程性和高效的并行处理能力，非常适合用于嵌入式系统和物联网设备，可以根据具体应用场景快速定制硬件逻辑，提高系统性能和能效比。总之，FPGA技术的发展不仅推动了电子设计自动化（EDA）技术的进步，还在云计算、边缘计算、等众多领域开辟了广泛的应用前景[3]。

在当今的现代应用场景中，对实时性能的需求确实越来越高，特别是在视频会议和实时监控等领域。在这些场景下，需要快速而准确地对图像进行处理，

以适应不同尺寸的显示设备或网络传输带宽。基于FPGA的图像缩放算法可以提供高效的实时处理，因为FPGA具有可编程性和并行计算能力，能够根据具体需求进行优化设计，充分利用硬件资源。相比于软件实现，基于FPGA的图像缩放算法能够更好地利用硬件资源，提供更高的处理性能和能效比。通过在FPGA上实现图像缩放算法，可以为各种应用提供通用的、可配置的硬件加速器，从而提升系统性能和灵活性[4]，这种方法不仅能够满足实时性能的要求，还能够应用于各种领域，包括数字显示、图像处理和计算机视觉等，为这些领域的发展提供了新的可能性。因此，基于FPGA的图像缩放研究在当今具有重要的应用前景和研究意义。

综上所述，基于FPGA的图像缩放研究具有重要的应用和研究意义。通过充分利用FPGA的可编程性和并行计算能力，图像缩放算法可以在保证高效实时性能的同时实现资源优化，同时具备灵活的算法选择和应用拓展性，可以满足不同领域对图像处理的需求。这种技术的发展对于提升图像处理系统的性能和灵活性，推动相关领域的发展和创新都具有积极意义。

1.2 国内外研究现状

目前，随着数字媒体技术的快速发展，视频处理技术已成为研究的热点之一，特别是在实时视频缩放领域。实时视频缩放是多媒体应用中的关键技术，本节会在现有的图像缩放算法和硬件设计的基础上，总结归纳图像缩放算法和硬件设计的国内和国外现状。

在图像缩放算法中，最常见的的缩放算法是最近邻算法、双线性算法以及双三次插值算法。不过，近些年来有很多学者提出了不同的图像插值的优化方法，在国外，2012年，Durand提出了B样条插值在计算机二维动画有理变换问题中的应用[5]。2015年，Unser、Darwish等人先后在它的方法基础上提出了三次B样条插值算法可以进行图片的缩放计算和数字滤波，这些算法是对几个基函数采用卷积形式而得，图像缩放过程中所使用的插值点阵是通过特定的标准而定的[6-7]。2016年，Su D等人提出了一种“网格图像“算法，该算法可用于静止图像在连续空间中的任意分辨率增强、任意旋转等应用[8]。2010年，Casciola提出了一种基于径向基函数（RBF）的插值方法，该方法是从离散图像数据样本中恢复了连续强度函数，所提出的各向异性RBF插值旨在能够处理数据中的局

部各向异性，例如图像中的边缘结构[9]。Giachetti描述了一种基于两种不同过程组合的新型通用图像插值方法，该方法提供了具有自然外观的插值图像，不会出现影响线性和非线性方法的伪影[10]。Gilman提出了一种新技术，用于将非均匀采样图像重采样到均匀网格上，该网格可用于融合翻译的输入图像，该方法可以实现为低阶的有限脉冲响应滤波器[11]。2013年，Seong G提出一种基于多假设运动估计和纹理优化的运动补偿帧插值（MCFI）算法来提高视频时态分辨率[12]。2019年，Bailey提出了一种执行实时双线性插值的新方法，该方法在镜头畸变校正等应用中非常有用，其中输入坐标遵循跨越多行的弯曲路径[13]。在国内，有不少学者对图像缩放技术进行了深入的研究，2010年，蔡念提出一种基于小波的双线性插值误差补偿算法，该算法增加双线性插值的误差补偿项，利用Sobel算子设定插值点的边缘方向，得到初始放大图像，利用小波提取高频成分，原始图像幅值增强充当低频部分，再经过小波逆变换得到高分辨率图像[14]。龚昌来提出一种改进算法：先计算插值点的双线性插值和最近邻点插值，然后以4个邻点的灰度方差构造权重，将二种插值进行加权融合获得最终插值结果，有效地提高了放大图像的质量[15]。杨朝霞提出利用B样条的尺度方程进行图像缩放的算法，该算法可对不同阶数的B样条有统一而简单的编码，通过设定不同的误差限，可以得到不同精度的缩放结果，并且计算时间可以随着不同的精度要求而得到控制[16]。2017年，陈杰提出了一种新的知识驱动权重估计框架，实现了更快的收敛速度和更高的性能[17]。2010年，盛敏结合混合插值样条(BIS)性质的特点,给出一类新的自适应图像插值算法，该方法可满足任意正实数倍的放缩要求，由于采用单核处理，而且插值像素点时无需求解方程组，因此计算复杂性较小[18]。

早期的图像缩放算法主要集中在软件实现，但在计算机上运行速度较慢，无法满足实时处理的需求。随着对实时视频处理和高清图像处理需求不断增加，研究人员开始考虑将图像缩放算法移植到硬件平台上，以提高计算效率和实时性能。近些年来，科研人员为了提升图像处理的效率也在不断的研究和突破。在国外，2017年，Safinaz提出了一种通用算法（Lanczos算法），用于在不影响图像质量的情况下增强具有给定比例因子的运动图像，该算法使用符合工业中使用的RTL编码的Verilog成功实现。获得的缩放图像质量很高，PSNR值为35dB或更高[19]。Gribbon提出了一种新型的桶形失真校正算法的FPGA实现，重点是

降低硬件复杂性，并提出了一种实时方法，这种方法使用一种新的行缓冲方法来补偿数据带宽限制，以尝试获取插值所需的像素[20]。在国内，2008年，刘怡设计通过FPGA对输入图像的实时运算实现对图像的缩放变换，然后在VGA监视器上显示，在FPGA中实现不再依赖DSP信号处理，很大程度上降低了系统体积[21]。林志强提出了一种新型的图像插值方法——扩展线性插值，设计了一种具有数字图像缩放的低成本硬件架构，以满足实时性需求[22]。张弘提出将双线性插值算法通过并行改造,将算法中的计算和存储需求与FPGA芯片内部的可用资源相映射，并且做到简化处理、资源节约、速度达标等优势[23]。

因此，在FPAG硬件平台上，能够实现高效的实时缩放处理，通过缩放算法对比，双线性插值算法与双三次插值算法的缩放效果相差不大，但在硬件设计的复杂度上，双三次插值的硬件设计比双线性插值要更为复杂，双线性插值在硬件上的处理速率更为快速，在硬件资源消耗方面上，双线性插值相对于双三次插值大大减少了资源消耗[24]。

1.3 研究内容

本文研究的主要内容是以FPGA作为硬件平台，搭建基于双线性缩放算法的视频缩放系统，通过对几种缩放算法的研究，权衡了几种算法在图像质量、硬件资源消耗和设计复杂度几个方面上的比较，最终选择了双线性缩放算法作为本文的研究对象。本文研究了基于均值双线性缩放算法的优化方法，首先，采用四个BRAM对图像进行行缓存，缓存双线性插值邻近的四个像素数据，然后对双线性缩放算法邻近的四个像素点进行均值处理，将双线性插值算法中的加权系数取为1，最后，通过移位的方法取得目标像素值，该方法在硬件设计上取消了乘法器的使用，在MATLAB软件上对改进的双线性算法进行仿真，对三张分辨率为640×480的不同图片缩小到分辨率为320×240，计算缩放后图像的PSNR和MSE值，与常规的双线性缩放算法进行对比，最终结果相差无几。在vivado软件搭建双线性缩放模块和改进的缩放模块，通过对两种算法模块综合后得到的底层资源进行对比，改进的双线性插值模块取没有使用乘法器，LUTS和FF分别减少了40%和30%。最后，以Verilog硬件描述语言对视频缩放系统中各个模块进行设计，在vivado软件中构建整个视频缩放系统通过综合后进行验证。

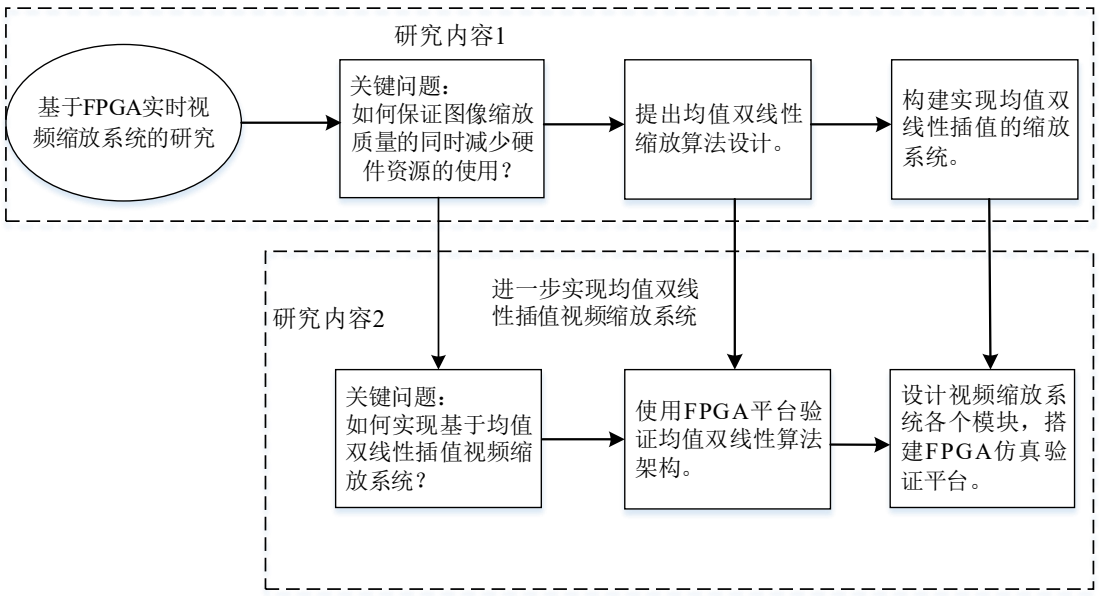


图1-1 研究内容框架

本文具体的研究内容如下：

（1）对不同图像算法的研究和学习，主要研究最邻近算法，双线性算法以及双三次算法，Matlab仿真软件对三种相同分辨率的不同图片进行仿真实验，仿真结果通过主观和客观的图像质量评价标准进行比对，最后，权衡了不同算法的运算复杂度和仿真结果的图像质量，折衷选择了双线性算法作为本文的算法研究。

（2）通过图像缩放算法的研究，提出了均值双线性插值算法及其架构。首先，采用四个BRAM对图像进行行缓存，缓存双线性插值邻近的四个像素数据，然后对双线性缩放算法邻近的四个像素点进行均值处理，最后，通过移位的方法取得目标像素值，该方法在硬件设计上取消了乘法器的使用，提高了算法运算速率，通过Verilog硬件描述语言设计双线性算法模块改进和双线性算法模块，通过综合后的资源消耗情况，降低了LUT和FF资源的使用。

（3）根据均值双线性算法模块的测试效果，搭建了基于改进双线性算法的视频缩放系统，验证了该算法的正确性，对视频缩放系统中各个模块通过内置的ILA（逻辑分析仪）进行仿真验证，最终实现了板级验证，实现了视频分辨率从640×480缩放至分辨率为1920×1080之间的缩放，验证了整个缩放系统系统的性能。

1.4 章节安排

本文是基于FPGA实时视频缩放系统展开研究，在结构上分为五个章节，每个章节的具体内容如下：

（1）第一章为绪论，介绍了视频缩放的背景与意义，总结了国内外对图像缩放在硬件上的发展现状，提出了本文所要研究的内容，然后对论文的进行结构上的安排。

（2）第二章对不同插值算法的理论研究和Matlab仿真实验，首先，对缩放技术的理论知识和原理的研究，然后对已有的插值算法进行了详细介绍，如最邻近插值，双线性插值以及双三次插值，并对这三个算法通过Matlab软件实现仿真实验，最后，通过主观和客观的图像质量评估标准，分析了这三个算法缩放后图像的质量，对仿真结果进行分析对比。

（3）第三章对双线性插值模块的FPGA实现的描述，对缩放模块进行硬件设计和仿真测试，然后，对双线性插值算法模块进行改进，利用Matlab软件实现对均值双线性算法的仿真实验，与双线性插值的仿真结果进行实验对比，最后，搭建均值双线性算法模块，对进行设计实现和仿真实验，验证均值双线性算法的可行性。

（4）第四章是基于FPGA视频缩放系统验证平台的搭建，设计验证平台中所需要的模块，包括摄像头视频图像采集模块、DDR3图像数据缓存模块、HDMI视频传输模块，使用vivado自带的逻辑分析仪ILA对设计的模块进行验证分析，最后通过板级验证实现视频缩放实验。

（5）第五章是对全文研究的总结和对未来的展望。

第2章 图像缩放算法原理与MATLAB实现

2.1 图像缩放技术

图像缩放技术是图像处理领域的一个核心组成部分，主要通过运用各种插值方法重新采样图像，来创造出分辨率更高或更低的新图像版本[25]。这项技术旨在调整图像的分辨率，无论是提升还是降低，均通过缩放算法根据原图的像素值为新图像的不同位置分配适当的像素值。此过程核心依赖于插值算法，该算法对原图进行重采样，生成新的插值图像，从而有效完成图像的尺寸调整。整个图像缩放过程是一个复杂的计算过程，其中插值算法起到关键作用。这些算法能够有效地处理图像中的像素值，以产生平滑、真实的图像。在重构图像时，新的坐标对应的新像素值通过插值算法计算而来，保持了图像在缩放过程中的连续性和准确性。图像缩放技术在图像处理中的广泛应用主要体现在提升图像质量和适应不同显示环境方面。无论是在数字摄影、医学影像还是娱乐产业，图像缩放都是不可或缺的步骤，通过选择合适的缩放算法，可以在不同场景中实现最佳的图像展示效果[26]。因此，图像缩放技术的深入研究对于提高图像处理的效率和质量具有重要意义。

图像缩放的核心在于应用图像插值技术，以插值方法来拟合曲线[27]。通常，原始图像在缩放过程中会经历采样和量化步骤，这可能会导致数据信息的丢失。为了补偿这一信息缺失，就需要对采样和量化之后的数据进行有效的重建[28]。这项复杂的任务涉及处理离散化的源信号图像，并构建尽可能贴近原始图像函数的连续信号，然后，根据目标缩放比例，对该连续信号进行再量化和重采样，从而获得完整的目标图像离散数据。这个步骤非常关键，因为它确保了图像缩放过程中图像信息能够被准确重建，有效地弥补了采样量化带来的数据损失[29]。通过对新的离散图像信号的重新构建，能够获得一个经过缩放处理的目标图像，既保持了其精度也保持了其完整性。这一复杂但关键的步骤对于实现高品质的图像缩放至关重要。插值函数的选择对于准确地重建原图像、重采样和重构后目标图像的质量有着直接的影响。不同的算法选择对于达到图像缩放的精确度和效果具有决定性作用。

调整图像分辨率本质上是二维空间几何转换，通常称作图像缩放，这一过程受到多种因素的影响，包括图像的清晰度、处理的效率以及最终显示的平滑度[30]。图像缩放的核心技术是利用插值算法，基于已知的像素点来估算目标像素点的值。因此，在深入探讨图像缩放技术之前，理解和研究插值理论是非常重要的一步。

2.1.1 图像重构理论

图像重构是数字图像处理领域的重要技术之一，其目标在于通过处理、修改或还原图像来提升其质量、清晰度或其他视觉特性。该领域涵盖了多种技术和方法，涉及信号处理、数学建模、统计学等多个学科的理论与方法。

图像调整大小的过程涉及两种主要操作：放大和缩小。这一过程大多算法依据图像重建理论，这涉及到使用数学理论对数据点进行拟合，目的是根据一组已知的离散数据点重构出连续的函数。这样做的目标是确保重构出的函数与原始图像尽可能地接近。通过对这个连续函数进行离散化采样，得到最终调整大小后的图像。简而言之，图像重建过程就是对图像中的点阵重新整合，该过程如公式（2-1）所示。

$$\Lambda = \{x \in R^{\kappa} |\ x = \sum_{k=1}^{k} n_{\kappa} V_{\kappa}, \forall n_{\kappa} \in Z\} \tag{2-1}$$

R^K代表 K 维空间,基矢量 $V^K \in R^K, k \in \{1,2,\cdots,K\}$，矩阵 $[V]=[V^1,V^2,\cdots,V^K]$。

2.1.2 理想图像插值函数

插值函数的选择对于准确重建源图像具有关键作用。通过对空间域的分析，可以将图像缩放视为对已知的原始图像的像素值与插值算法的基函数进行乘积和求和运算，这个过程本质上是卷积处理，可以通过公式(2-2)来描述，接着，根据目标图像的比例对连续的源图像信号进行量化和重采样，最终得到待测图像的像素值[31]。在这一过程中，插值函数的选择对于重建连续源图像信号的准确性起着关键作用。

$$S(x) = \sum_{i=\lfloor x \rfloor - a+1}^{\lfloor x \rfloor + a} s_i h(x-i) \tag{2-2}$$

公式（2-2）中，$S(x)$ 代表位置在 x 处通过插值计算得到的像素值，$\lfloor x \rfloor$ 用

于表示将 x 的值向下取整至最近的整数，i 用于标记原始图像中被选为参考点的位置，$h(x)$ 则是被选用来进行插值的函数。根据傅里叶变换的理论，可以通过在频域上将信号进行相乘的方式来实现空间域信号的卷积操作，理想插值函数频谱特性如图2-1所示。

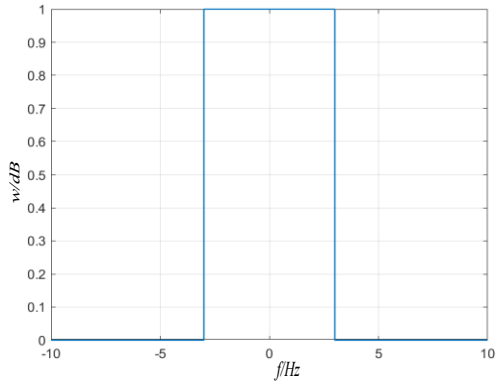


图2-1 *Sinc*函数频谱图

接下来，基于理想插值函数在频率领域的属性，通过傅立叶逆变换的方法得到其在时域的数学表达如公式（2-3）所示。其空域函数图如图2-2所示。

$$h(x)=\sin cx=\begin{cases}\dfrac{\sin(\pi x)}{\pi x}, & x\neq 0\\ 1, & x=0\end{cases} \tag{2-3}$$

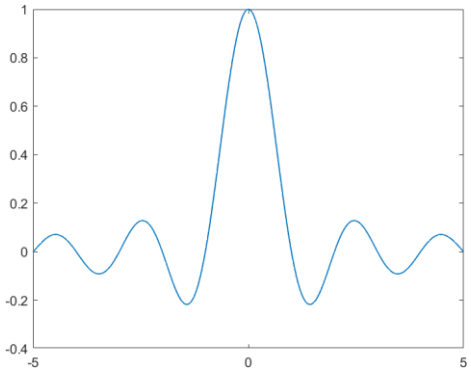


图2-2 *Sinc*函数

理想插值函数在空域上呈现出一种特殊的特性，即它在无限范围内延伸。然而，在实际应用中，由于计算资源和实用性的考虑，无法实现完全的 *Sinc* 函数无限扩展。为了在有限的条件下实现类似理想插值的效果，通常采用逼近 *Sinc* 函数特性的插值基函数。

常见的插值基函数包括多项式函数、矩形函数和三角函数[32]。这些函数在插值过程中起到关键的作用，通过它们的乘积来逼近目标函数，从而实现对信

号或图像的插值。多项式函数通常用于拟合复杂的数据，而矩形函数和三角函数则在特定情境下展现出更为优越的性能。

在比较不同理论算法时，评估算法好坏的一个有效方法是分析各种插值基函数的特性曲线与理想插值函数的拟合程度[33]。这种比较有助于确定在特定应用场景中选择合适的插值算法，以实现最佳的重建效果，通过对插值基函数的选择和理论比较的深入研究，能够更好地理解插值算法的性能，并为实际应用提供可靠的指导。

2.2 图像缩放算法

在处理图像缩放时，面临着对图像缩放算法的合理选择，特别是在涉及视频数据时，其庞大的运算量以及对实时性的迫切需求成为考虑的关键因素。在这一背景下，利用FPGA进行图像缩放是一种有效的解决方案，然而，FPGA本身的逻辑资源存在一定的限制，这需要在算法选择上进行权衡。

在算法选择方面，不仅需要考虑精度，还要综合考虑算法的复杂性以及对硬件资源的占用情况。尽管高精度的图像缩放算法在一些场景中可能是理想的选择，但其复杂性导致其在FPGA上占用更多的逻辑资源，可能与实时性的要求相冲突。

因此，针对实时性和硬件资源的双重考量，线性插值算法成为一个值得关注的选择。这类算法在保持适度精度的同时，相对较为简单，能够更好地适应FPGA的硬件资源限制。在图像缩放的过程中，线性插值算法能够在满足实时性需求的同时，有效减少对逻辑资源的占用，从而提供一种平衡性能和资源利用的解决方案。

2.2.1 最邻近插值算法

最邻近插值（Nearest-neighbor interpolation）是一种图像插值方法，用于在图像处理中调整图像的大小或改变其分辨率，该方法的基本思想是，对于目标图像中的每个像素，都选择源图像中距离最近的像素的值作为其插值结果[34]。具体而言，对于目标图像中的每个像素，最邻近插值会找到在源图像中距离最近的像素，并将其像素值赋给目标像素。这种插值方法实现简单，计算速度快，但是，在放大图像的时候，可能会导致图像的锯齿状效果。

从计算的角度来看，最邻近插值操作利用零阶多项式相当于在数值处理中引入了矩形窗函数[35]。在插值操作中，零阶多项式的使用使得对原始数据的近似更加简单，但会引起的平滑度低和信息损失。矩形窗函数的表达式如式（2-4）所示。

$$h(x)=\begin{cases}1 & 0<|x|<0.5\\ 0 & |x|>0.5\end{cases} \tag{2-4}$$

最邻近插值算法根本上是通过复制最近的像素点的像素值作为目标像素点的方法，其操作流程如图2-3所示。

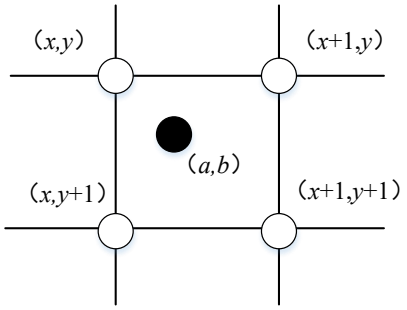


图2-3 最邻近插值示意图

图中的黑点表示目标插值点，其坐标表示为(a,b)，而白点则表示与目标插值点距离最近的4个已知像素点分别为(x,y)、$(x,y+1)$、$(x+1,y)$、$(x+1,y+1)$，像素值分别表示为$f(x,y)$，$f(x,y+1)$，$f(x+1,y)$，$f(x+1,y+1)$。在实际运算中，首先沿水平方向进行一次距离比较，得到距离最近的两个像素值，然后在垂直方向再进行距离比较，选择距离最近的像素值，最终确定目标插值点的像素值。

最邻近插值算法的插值基函数及其频谱图如图2-4和图2-5所示。

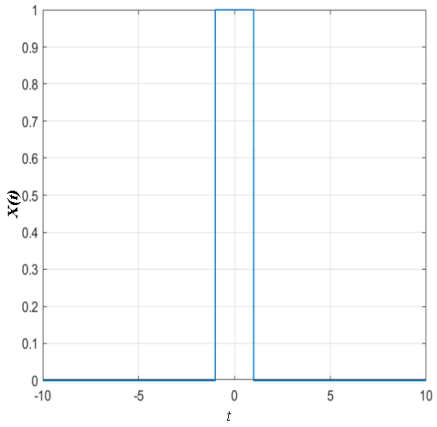


图2-4 最邻近插值示意图

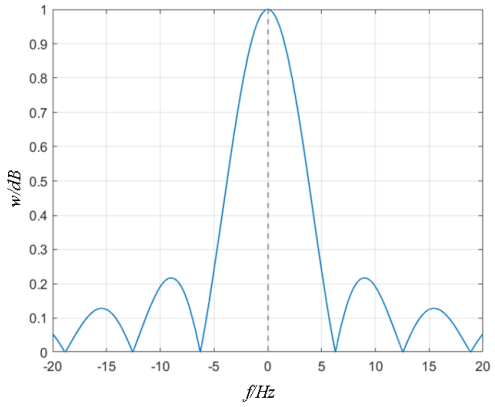


图2-5 最邻近插值函数傅里叶变换频谱图

从图中的傅立叶变换频谱图可见，第二峰值大约为21%，显示与理想低通滤波的明显差异。最邻近插值算法在图像缩放中，尤其是在放大操作中，可能导致边缘锯齿和像素化效应，因其处理过程仅复制最近的像素值。虽然在对缩放质量要求不高或像素差异小的情况下这种算法处理速度快，但在高放大倍数下不建议使用，以避免质量下降。

最近邻插值算法具有简单的计算过程、低芯片资源消耗和快速运行的优势，使其在硬件实现上相对容易[36]。最邻近插值的优点是简单快速，用于一些对于图像质量要求不高实时应用场景。在实际应用中，需权衡硬件实现的便捷性和插值图像质量之间的取舍，根据具体需求选择适当的插值算法。

2.2.2 双线性插值算法

双线性插值算法也选择了2×2的邻近点作为参照点，双线性插值的示意图如图2-6所示。双线性插值的基本思想是利用目标像素位置在水平和垂直方向上的权重，来从源图像中的四个最邻近像素中计算插值结果[37]。这四个最邻近像素构成一个矩形，目标像素的位置落在这个矩形内部，插值的权重是通过目标像素与最邻近像素之间的相对距离来确定的[38]。

双线性插值的步骤如下：

1）计算目标像素在源图像中的位置，得到其在水平和垂直方向上的坐标（通常使用线性映射）。

2）找到目标位置周围的四个最邻近像素，通常是左上、右上、左下和右下四个像素。

3）计算目标像素与这四个最邻近像素之间的相对距离，得到在水平和垂直

方向上的权重。

4）通过对四个最近邻像素的值进行加权求和，计算出最终的插值结果。具体的计算过程如图2-6所示。

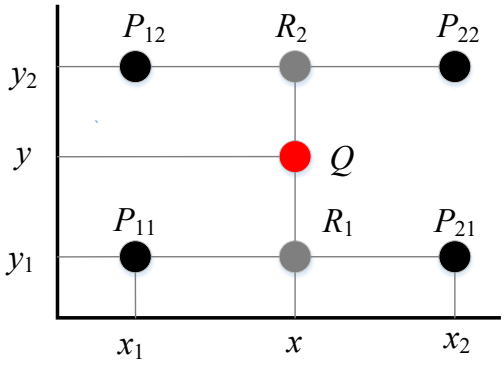


图2-6 双线性插值示意图图

其中红色圆点为待测像素点，它映射到源图像的点为$Q(x,y)$，它的像素值为$f(x,y)$，选取待测像素点Q邻近的四个像素点P_{11}、P_{12}、P_{21}、P_{22}，首先在水平方向上进行两次线性插值得到两个像素点$f(x,y_1)$，$f(x,y_2)$，如公式（2-5）和（2-6）所示。

$$f(x,y_1)\approx\frac{x_2-x}{x_2-x_1}f(P_{11})+\frac{x-x_1}{x_2-x_1}f(P_{21}) \tag{2-5}$$

$$f(x,y_2)\approx\frac{x_2-x}{x_2-x_1}f(P_{12})+\frac{x-x_1}{x_2-x_1}f(P_{22}) \tag{2-6}$$

然后在垂直方向上再进行一次线性插值，即对两个像素点$f(x,y_1)$，$f(x,y_2)$再进行插值计算得到$f(x,y)$，公式如（2-7）所示。

$$f(x,y)\approx\frac{y_2-y}{y_2-y_1}f(x,y_1)+\frac{y-y_1}{y_2-y_1}f(x,y_2) \tag{2-7}$$

最后将式（2-5）、（2-6）带入式（2-7）中通过计算可得目标像素值如公式（2-8）所示：

$$f(x,y)=\frac{1}{(x_2-x_1)(y_2-y_1)}\begin{bmatrix}x_2-x & x-x_1\end{bmatrix}\begin{bmatrix}f(P_{11}) & f(P_{12})\\ f(P_{21}) & f(P_{22})\end{bmatrix}\begin{bmatrix}y_2-y\\ y-y_1\end{bmatrix} \tag{2-8}$$

通过先进行垂直方向上的线性插值，然后再对水平方向上进行线性插值运算，得到的结果与先水平后垂直的顺序是一样的，这一点可以通过验证得以证实。这种顺序无关性为图像插值提供了更大的灵活性，同时也使得算法实现更

为简便。双线性插值算法更加精细地处理图像的连续变化，该算法的关键在于通过对原始像素点之间线性关系的双线性加权计算，推导得到目标像素点的值。该插值算法的插值基函数如公式（2-9）所示。

$$h(x)=\begin{cases}1-|x| & -1<x<1\\ 0 & else\end{cases} \tag{2-9}$$

双线性插值算法的时域图和频谱图图如图2-7和图2-8所示。

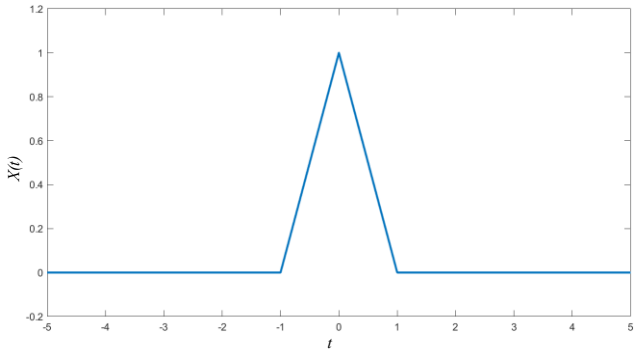


图2-7 双线性插值函数

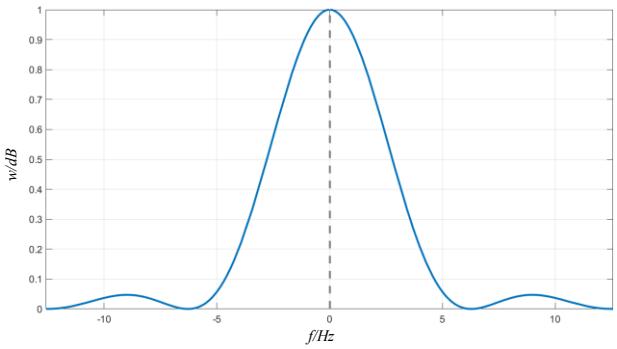


图2-8 双线性插值函数频谱图

从双线性插值算法的时域和频谱图分析可知，其基函数显示出了出色的带阻低通滤波效果，有效实现图像缩放的平滑性[39]。该算法考量了目标像素与其四邻域像素的距离关系，即目标像素与某邻域像素越接近，其对目标像素的影响及权重越重，这相较于最邻近插值方法，明显减少了缩放图像的马赛克和锯齿现象，保持了图像的平滑性和连贯性[40]。因其在图像缩放质量、效率、硬件要求和处理速度方面的优良平衡，双线性插值特别适合于在FPGA硬件平台上开展实时视频图像缩放相关的研究。

结合双线性插值原理，搭建双线新插值电路图如图2-9所示，首先，在水平方向上，进行两次的线性插值运算获得两个中间点R_0和R_1，然后，在垂直方向

上，对两个中间点再进行一次线性插值运算，最终获得目标值 $F(x,y)$，在硬件实现上其所需要的乘法器为8个，加法器为7个。

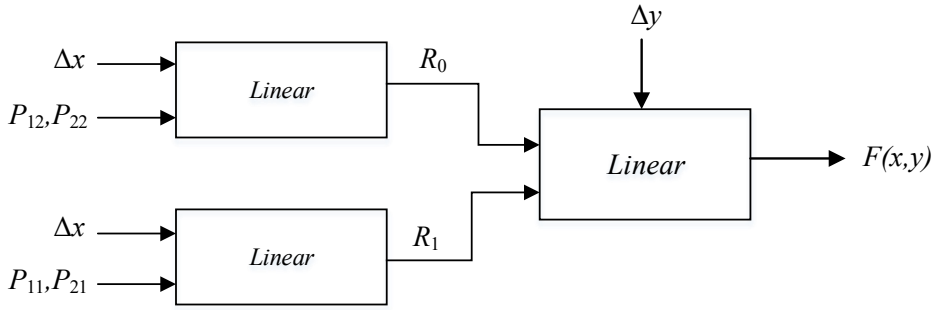


图2-9 双线性插值电路图

2.2.3 双三次插值算法

双三次插值（Bicubic interpolation）是一种更高阶次的插值方法，相对于双线性插值而言，提供了更为精确和平滑的结果[41]。这种插值方法在图像处理中常用于图像的缩放、旋转以及其他形变操作。与双线性插值类似，双三次插值也是通过目标像素位置周围的最邻近像素来进行插值的。不同之处在于，双三次插值利用更多的邻近像素，并使用三次多项式进行插值计算，双三次插值算法是选取待测像素点邻近的16个像素点作为参考点[42]，双三次插值算法示意图如图2-10所示。这也说明此算法比前两个算法的计算复杂度更大，但在图像缩放效果上，双三次插值算法可以得到更高插值效果。

双三次插值函数如公式（2-10）所示。

$$S(x)=\begin{cases}(a+2)|x|^3-(a+3)|x|^2+1 & ,\ 0\le|x|<1\\ a|x|^3-5a|x|^2+8a|x|-4a & ,\ 1\le|x|\le 2\\ 0 & ,\ 2\le|x|\end{cases} \tag{2-10}$$

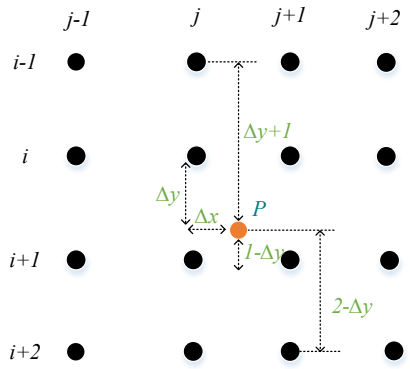


图2-10 双三次插值算法示意图

上式中由于变量a会有多种双三次插值基函数，而通过Keys的推算[43]，得出a=-0.5为最优解，能够通过最好的插值效果。如公式（2-11）所示。

$$S(x)=\begin{cases}\frac{3}{2}|x|^3-\frac{5}{2}|x|^2+1, & 0\le|x|<1\\ -\frac{1}{2}|x|^3+\frac{5}{2}|x|^2-4|x|+2, & 1\le|x|\le 2\\ 0, & 2\le|x|\end{cases} \tag{2-11}$$

双三次插值算法是一种更高阶的插值技术，该算法利用邻近16个像素点的颜色数据，考虑了像素间的颜色变化幅度，实现了高度精准的插值结果。对于待插值点$F(x,y)$的像素值如公式（2-12）所示。

$$F(x,y)=A\bullet B\bullet C \tag{2-12}$$

其中A、B、C等参数为矩阵。这些参数的具体表达式如公式（2-13）所示：

$$\begin{aligned}A&=\begin{bmatrix}S(1+\Delta x) & S(\Delta x) & S(1-\Delta x) & S(2-\Delta x)\end{bmatrix}\\ B&=\begin{bmatrix}f(i-1,j-1) & f(i-1,j) & f(i-1,j+1) & f(i-1,j+2)\\ f(i,j-1) & f(i,j) & f(i,j+1) & f(i,j+2)\\ f(i+1,j-1) & f(i+1,j) & f(i+1,j+1) & f(i+1,j+2)\\ f(i+2,j-1) & f(i+2,j) & f(i+2,j+1) & f(i+2,j+2)\end{bmatrix}\\ C&=\begin{bmatrix}S(1+\Delta y) & S(\Delta y) & S(1-\Delta y) & S(2-\Delta y)\end{bmatrix}^{\mathrm{T}}\end{aligned} \tag{2-13}$$

双三次插值函数和相应的傅立叶频谱如图2-11和图2-12中所示。该插值函数的傅立叶频谱呈现出较好的特性，其中第二峰值约为第一峰值的1%，这说明插值函数在抵抗抖动方面表现出色。通过观察主要波峰，可以看到高频成分的减弱相对缓慢，这有助于维持图像中的高频细节，整个频谱图显示了该插值函数的优秀低通滤波器特性[44]。

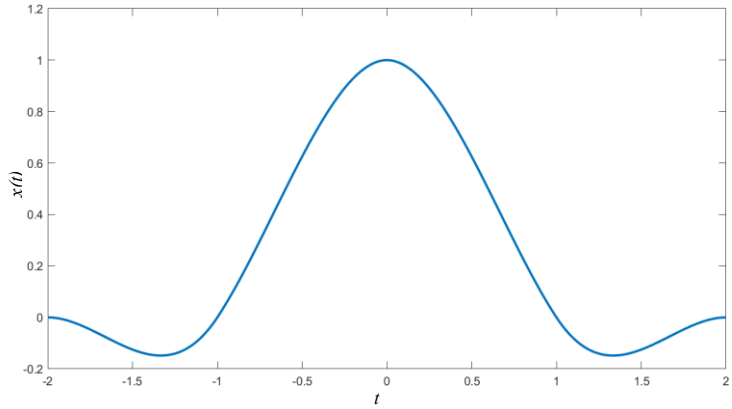


图2-11 双三次插值基函数

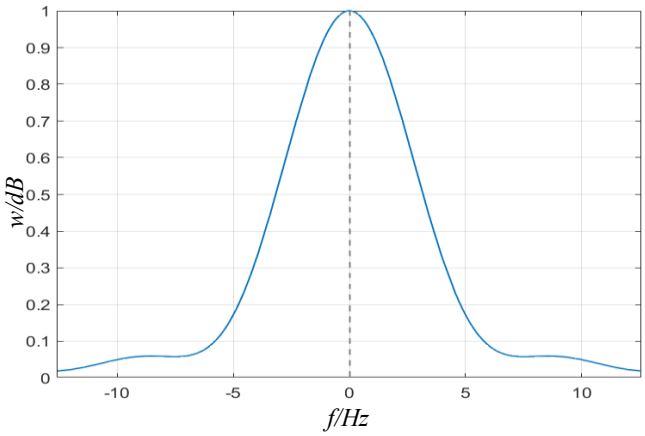


图2-12 双三次插值函数傅里叶变换频谱图

双三次插值算法需要大量的乘法和加法运算，双三次插值的电路结构图如图2-13所示。首先，在垂直方向上进行四次插值运算，图下方是垂直方向上的四个插值系数运算过程，然后通过垂直方向上的插值运算获得四个中间点(S_0,S_1,S_2,S_3)，最后，对四个中间点再进行一次插值运算获得待测点$F(x,y)$，在硬件设计上，与双线性插值算法相比需要大量的逻辑单元和乘法器的使用，乘法器需要用到36个，从而大大降低了运算速率。

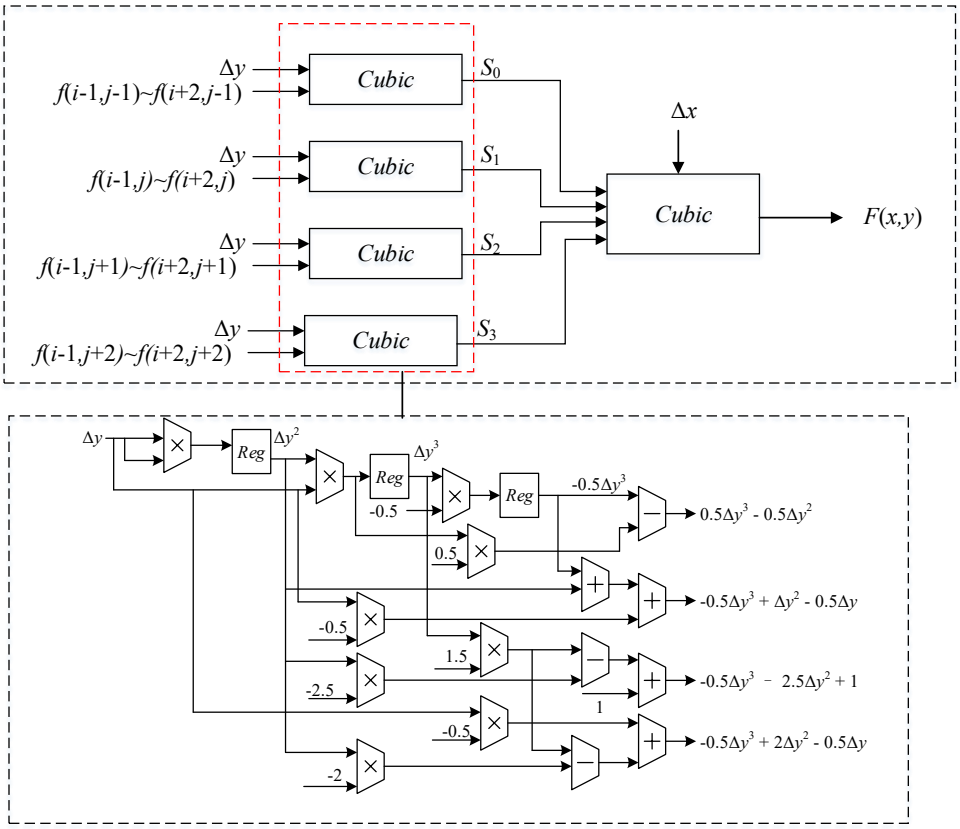


图2-13 双三次插值电路结构

虽然双三次插值算法在处理图像边缘时展现出优秀的曲面平滑过渡，但是，为了实现这种平滑过渡，算法的复杂性显著提升。因此，该算法更适合在软件上实现，但在FPGA硬件平台上的资源占用较多，实现起来相对繁琐，因此并不适用于处理高速的数字视频流。

2.3 图像缩放算法评价标准

在图像处理领域内，图像质量评估是至关重要的一环，它不仅在算法的效率分析上发挥着决定性作用，还在对比不同图像处理系统的性能时提供了关键的评价标准。随着图像处理应用需求的日益增长和多样化，学术界和工业界相继提出了一系列用于图像质量评价的标准和方法，并不断对其进行精细化改进。根据评估过程中观察者参与的程度，这些方法主要分为主观评价和客观评价两大类[45]。

2.3.1 主观质量评价

主观图像质量评价是一种以人类观察者的主观参与为基础的图像评估方法，其重点在于综合考虑人眼对图像的视觉感知和认知[46]。这一评价方法通常通过主观实验，要求受试者对图像的整体质量进行主观打分或提供质量等级，以捕捉人们在视觉上对图像的主观感受。由于主观图像质量评价考虑了人类感知的主观性和个体差异，因此被认为是一种直观而准确的图像评估手段。主观评估结果更全面地反映了人们对图像的实际感受，为图像处理算法和系统性能评估提供了更为详尽和深入的参照。

这种评价方法在实际应用中具有广泛的适用性，特别是在需要考虑人类主观感觉和感知差异的领域，如视觉传媒和医学图像分析等。通过主观图像质量评价，可以更好地了解图像处理结果对人的视觉体验的影响，为优化算法和系统提供有针对性的改进方向。因此，主观图像质量评价在图像处理领域的研究和实际应用中发挥着不可替代的作用。

2.3.2 客观质量评价

客观质量评价标准是图像处理和视频编解码领域中的关键工具，用于定量地衡量图像或视频处理算法的性能和质量。客观评价方法通过特定的数学模型进

行评估，利用该模型中定义的特征信息来作为评估图像质量的标准，以客观的方式分析和测量图像或视频的各个方面，如清晰度、失真、对比度等，旨在演示对人眼对图像感知的理解，客观图像质量评估的主要目标是构建出能够准确自主判定图像质量的计算模型[47]，目的在于让计算机能够模拟人眼对图像的感知能力。图像质量的具体评估流程如图2-14所示。在评估系统中，原始图像及其缩放图片被并行导入系统中。随后，系统会自动提取这些图像中的相同区域，无论是局部还是整体，以便识别并比较特定的特征细节。此比较过程包括产生相似度显著图，该图进而利用相似度参数*P*来间接评估图像质量[48]。此方法目的在于实现图像质量评价的自动化，确保了评估的客观性和准确性。

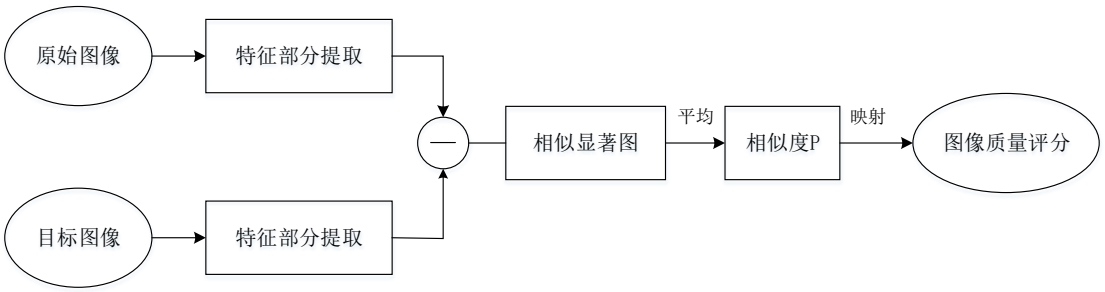


图2-14 图像客观评价流程图

对于图像特征参数的提取，目前有多种有效的方法。均方误差[49]（Mean Squared Error，MSE）和峰值信噪比[50]（Peak Signal to Noise Ratio，PSNR）是两种被广泛采用的图像质量评估技术，因其计算过程的简便而广泛应用。这两种方法主要通过比较参考图像与失真图像在空间域内各个像素点的差异来提取特征信息。具体而言，它们通过统计图像像素差的方式，来评价图像间的相似度或差异，进而对图像质量进行量化评估。此类评估技术通过分析像素级差异来实现对图像质量的精确判断。

*PSNR*是一项客观评估图像质量的指标，其计算基于对比原始图像与经过算法处理后图像在相同像素点上的数值差异。其数学表达式如公式（2-14）所示。

$$PSNR = 10\log_{10}\left(\frac{255^2}{\frac{1}{MN}\sum_{n}^{N-1}\sum_{m}^{M-1}\left[F(m,n)-G(m,n)\right]^2}\right) \tag{2-14}$$

其中 $F(m,n)$ 表示原始图像的像素值，$G(m,n)$ 表示经过缩放算法处理后的目标图像，其分辨率大小为 $M\times N$，其尺寸范围在0~255之间。

*MSE*是衡量原始图像与经过缩放处理后的图像之间像素值差异的均方误差，如公式（2-15）所示。

$$MSE=\frac{1}{MN}\sum_{n}^{N-1}\sum_{m}^{M-1}\left[F\left(m,n\right)-G\left(m,n\right)\right]^{2} \tag{2-15}$$

在图像缩放后，若处理结果图像与原始图像的相似性越高，均方误差（*MSE*）就会减少而峰值信噪比（*PSNR*）会显著增加，这表明引入的失真极低，从而缩放后的图像质量较高。一般而言，*PSNR*值超过35dB时，人眼难以识别出图像间的差异；当*PSNR*值超过25dB时，图像差异表现微小。然而，人对色彩和亮度的感知差异可能导致客观评价与主观感受之间的不同，这些客观质量评价指标扮演着至关重要的角色，为算法优化和参数调整提供了可靠的量化参考，从而确保最终成果能够满足预期的图像质量要求[51]。

2.3.3 缩放图像结果分析

通过使用Matlab软件平台，本研究实现了三种不同的图像缩小算法，对选定的图像进行了缩小处理，将它们的尺寸从640×480缩减至320×240。所选图像包括附录中的图1（古楼）、图2（山丘）、图3（山水画）、图4（花田）和图5（动物）。三种不同算法处理后的图片见附录中图6至图20。

通过主观评估对比三种图像缩放技术的处理结果，发现最邻近插值在图像质量方面表现最不理想，尤其是其引发的马赛克效应较为明显，同时在图像边缘处容易出现锯齿现象。相较之下，双线性插值算法在减少图像不连续性方面做出了改进，使图像呈现更加平滑的视觉效果。而在这三种算法中，双三次插值算法提供了最高的图像质量，显示出明显优于其他两种方法的性能。因此，从主观评价的视角出发，双三次插值算法的效果最好，双线性插值的图像效果次之，最邻近插值图像效果最差。

从客观评价标准进行分析，首先将不同缩放算法处理后的图像放大到与原始图像相同的尺寸，接着，将这些不同算法处理后的图像与原始图像作为公式（2-14）和公式（2-15）中的$G(m,n)$和$F(m,n)$，通过这过程能够计算出客观图像质量标准指标值。本文采用*PSNR*和*MSE*作为客观图像质量的标准，其值如表2-1所示。

表2-1 图像质量标准指标*PSNR/MSE*

图像类型（*PSNR/MSE*）	最邻近插值算法	双线性插值算法	双三次插值算法
古楼	33.263/30.673	33.388/29.739	33.440/29.447
山丘	35.166/19.790	35.375/18.862	35.419/18.672
山水画	33.259/30.700	33.360/29.994	33.378/29.872
花田	34.939/20.853	35.349/18.976	35.408/18.720
动物	33.278/30.570	33.410/29.653	33.572/28.564

分析表格数据显示，在图像缩放对比中，双三次插值算法的*PSNR*值最高，其次是双线性，最邻近插值的*PSNR*最低。通过*MSE*分析表明，最邻近的*MSE*最高，双线性和双三次较低且接近。综合主观与客观评价，最邻近插值的图像质量低于双线性和双三次插值。

2.4 本章小结

本章深入探讨了三种不同算法的理论基础，并通过Matlab软件对它们执行了仿真实验。通过综合主观与客观评价标准，本研究对比分析了这三种算法缩放图像的效果。研究结果表明，最近邻插值算法在处理图像时往往会产生较为显著的马赛克效应，导致图像质量明显下降；双线性插值算法能够使图像变得更平滑；双三次插值算法则能生成更清晰的图像，但其实现较为复杂，且需要消耗更多的硬件设计资源。这些结论为选取合适的图像缩放算法提供了宝贵的参考。综上所述，后续研究将进一步深入探讨双线性插值算法。

第3章 缩放算法模块的 FPGA 实现

根据上一章对不同图像插值模块的研究分析，本章选择了双线性插值算法进行深入研究，将会搭建常规的双线性插值模块和改进的双线性插值模块，首先对常规的双线性插值缩放模块设计相应的硬件架构，将双线性插值模块划分成三个模块，分别是数据储存模块、插值参数生成模块和插值运算模块，并使用vivado软件对搭建的缩放模块进行仿真实验，并获取缩放模块的底层资源消耗。然后提出了改进的双线性插值算法，对邻近的四个像素点的像素值取平均值代替目标像素点的像素值，通过Matlab软件实现常规双线性插值和改进双线性算法，对640×480图像进行1/2缩放处理，并对处理后的图片进行主观和客观图像质量标准比较分析，得出两种算法缩放处理后的图像质量效果相差无几，然后搭建改进的双线性插值模块，由于改进的双线性插值并没有使用乘法器进行插值运算，而是取邻近四个像素点的像素值的平均值，然后对所运算的平均值通过移位操作最终获得目标像素值，最后与常规的双线性插值模块的资源消耗进行对比，改进的双线性插值算法模块有效的减少了硬件资源的消耗。

3.1 双线性插值算法模块

通过上一章对不同算法的介绍和实验对比，在权衡算法复杂程度和硬件实现的资源消耗等方面后，最终选择双线性插值算法进行研究。双线性插值模块的整体框图如图3-1所示，将双线性插值模块分为了数据储存模块、插值参数生成模块和插值运算模块三个模块，数据缓存模块是通过四个RAM缓存双线性插值算法所需的四个邻近像素数据，通过RAM的读写控制模块控制像素数据的存储，中间通过FIFO进行RAM之间的缓存，插值参数生成模块是通过RAM读控制模块读取目标像素点左上角的邻近像素点的坐标，根据目标像素点坐标和水平垂直的缩放比例，获取插值运算中的所需的四个插值系数，插值运算模块通过获取的的四个插值系数与邻近的四个像素点进行运算，最终获得目标像素点的像素值。

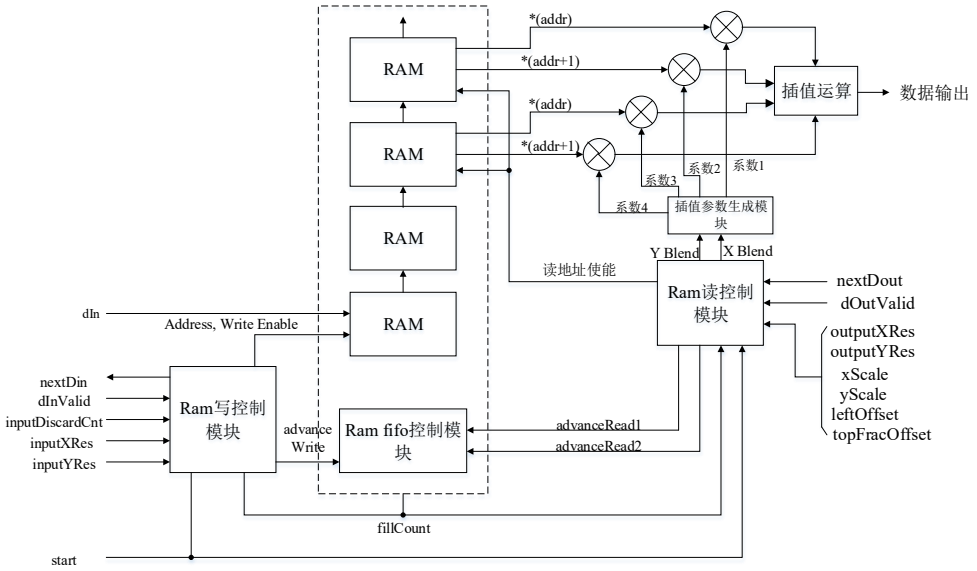


图3-1 缩放模块的整体框图

3.1.1 数据储存模块

双线性插值需要使用周围的4个像素来确定目标像素的值，在设计中，使用2个RAM可以满足目标像素周围4个像素的计算要求。但是，为了提高处理速度，决定使用4个双接口RAM，即内存fifo控制模块，实现更高效的数据读写乒乓操作。为了保证处理效率，在第一次计算中保证完整的数据存储在4个ram中，才能开始计算。四个RAM阵列的RTL图如图3-2所示

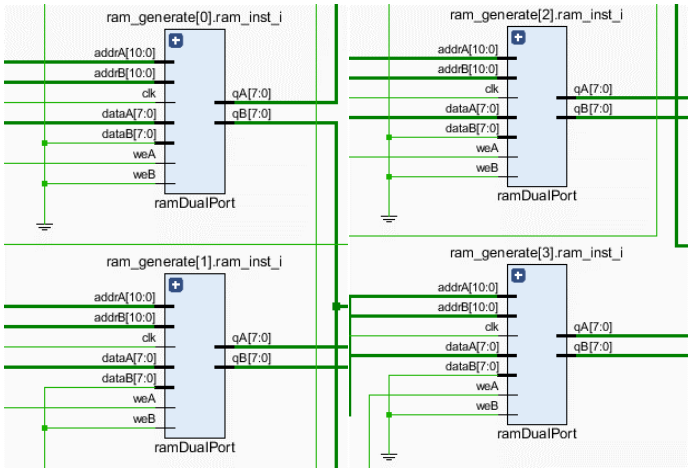


图3-2 四个RAM阵列的RTL图

3.1.2 插值参数生成模块

系统利用接收到的图像像素值数据确定输入图像中水平和垂直有效像素的数量，而输出分辨率大小由用户命令控制。该过程通过计算公式（3-1）和（3-2）实现，其中水平缩放系数为x_{scale}，垂直缩放系数为y_{scale}，$scrx_{res}$和$scry_{res}$表示

原始图像在水平和垂直上的像素数量，$detx_{res}$和$dety_{res}$代表目标图像在水平和垂直上的像素数量。

$$y_{scale}=\frac{scry_{res}}{dety_{res}} \tag{3-1}$$

$$x_{scale}=\frac{scrx_{res}}{detx_{res}} \tag{3-2}$$

其中双线性插值算法的运算如公式（3-3）所示：

$$Dout=a_1P_1+a_2P_2+a_3P_3+a_4P_4 \tag{3-3}$$

在该方程中，P_1、P_2、P_3、P_4是目标像素点最邻近的四个原始像素点的像素值，这四个像素点得值储存在RFIFO中，接下来，采用RAM建立的2×2阵列，有效地读出这四个像素点数据，还需要计算出插值系数a_1、a_2、a_3、a_4。

所求的插值系数如公式（3-4）所示：

$$\begin{cases}a_1=x_{blend}\times y_{blend}\\ a_2=(1-x_{blend})\times y_{blend}\\ a_3=x_{blend}\times(1-y_{blend})\\ a_3=(1-x_{blend})\times(1-y_{blend})\end{cases} \tag{3-4}$$

式中，*xblend*和*yblend*表示目标插值像素点与左上角邻近像素点的水平和垂直距离。设目标像素点缩放后坐的标为(x,y)，则*xblend*和*yblend*的计算公式如（3-5）所示。

$$\begin{cases}x_{blend}=x\times x_{scale}-[x\times x_{scale}]\\ y_{blend}=y\times y_{scale}-[y\times y_{scale}]\end{cases} \tag{3-5}$$

将公式（3-5）计算出的xblend和yblend的值带入到公式（3-4）中得出相应的插值系数a_1、a_2、a_3、a_4。最后将得到的插值系数带入到公式（3-3）中计算得到目标像素的像素值。

通过以上的分析可得，要求目标图像某点的像素值，首先需要找到对应的浮点坐标。将缩放比例$i_scaler_x_ratio$和$i_scaler_y_ratio$输入到RAM读控制模块，以这两个缩放比例当成步幅进行叠加，然后由RAM读控制模块输出浮

点坐标 $x_{ScaleAmount}$ 和 $y_{ScaleAmount}$。随后，通过移位的方式提取浮点坐标的整数部分 X_{pixlow}、Y_{pixlow} 以及小数部分 X_{blend}、Y_{blend}，得到浮点坐标的小数部分后，再利用浮点坐标的小数部分和基数1，可以计算得到4个系数的值，分别为 *Coeff*00、*Coeff*01、*Coeff*10。这一系列操作为后续的插值计算获得了相应的浮点坐标。

3.1.3 插值运算模块

在获取浮点坐标邻近的4个像素点系数后，将其与从RFIFO中读出的原图像的像素值进行乘法和加法运算。原图的像素值分别为*readData*00、*readData*10、*readData*01、*readData*11。首先，通过对这4个像素点与其对应系数进行乘法运算，得到*product*00、*product*10、*product*10、*product*11。随后，将这4个值相加，最终得到输出信号*o_vout_data*，即为目标图像的像素值。这一系列的计算步骤在插值过程中扮演了关键的角色，用于准确计算目标图像上各点的像素值。

双线性插值缩放计算流程如图3-3所示。首先，等待像素数据传入，当有效数据到达时，将双线性插值所需要的四个邻近像素数据存储到四个RAM中。设定当RAM存满四行视频数据后，系统缩放模块开始进行计算。最终，将计算得到的数据取整并输出。这一流程保证了系统在充分积累足够数据后再进行插值计算，以提高计算的准确性和效率。

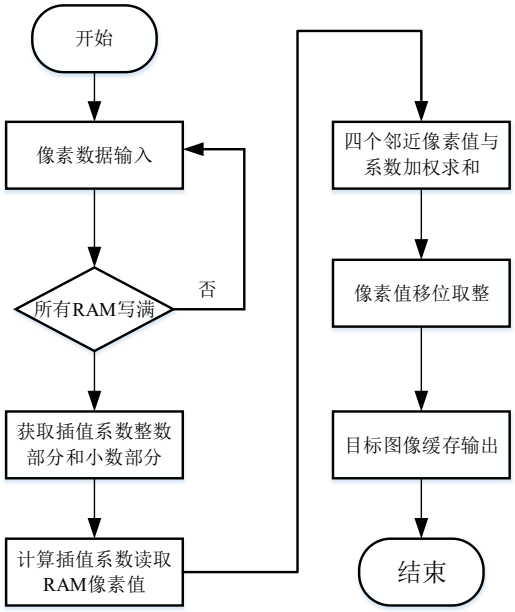


图3-3 双线性插值模块计算流程图

通过使用Vivado软件内置的仿真功能，对双线性插值算法模块执行了仿真测试，波形仿真如图3-4所示，该仿真波形是以640×480为输入像素值数据，经放大处理后得到1024×768的像素值数据输出。在仿真中，用到四个RAM模块储存双线性插值模块所需的四个近邻像素点的数据。在图中，*dout* 表示缩放后的数据，*coeff* 为目标像素点与四个邻近像素点的距离作为加权系数，x_{scale}、y_{scale} 为水平和垂直方向的缩放比例，*inputxres*、*inputyres* 表示原始图像的水平和垂直像素数，而 *outputxres*、*outputyres* 表示缩放后的图像的水平和垂直方向的像素数量。

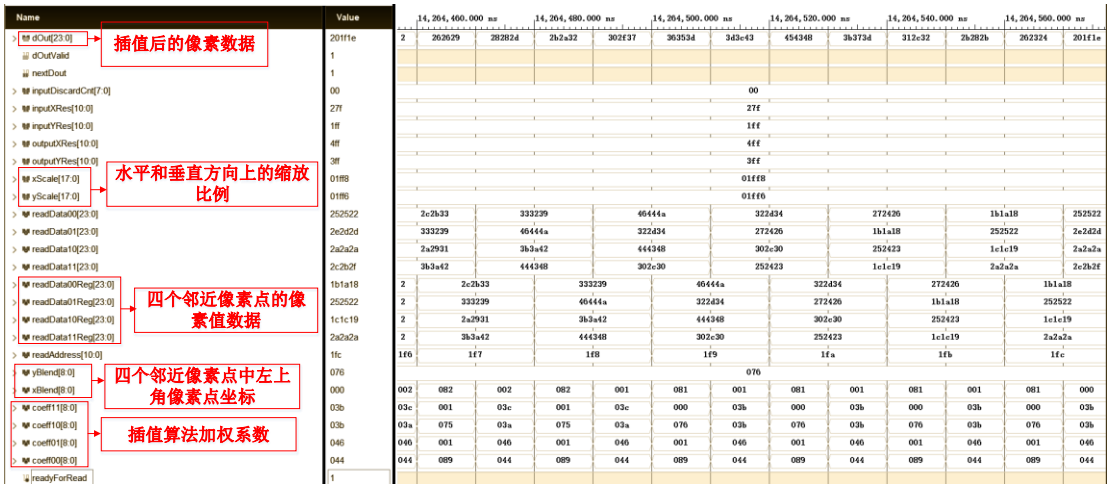


图3-4 双线性缩放算法波形仿真

最终生成的目标图像的像素值与通过Matlab软件进行双线性插值处理的结果相对应，使用Vivado软件进行综合后，得到了关于底层资源消耗的报告，如图3-5所示，在常规的双线性缩放算法模块中，对于逻辑功能产生器（LUT）、储存单元（BRAM）、触发器（FF）、信号处理器（DSP）、输入输出端口（IO口）的资源使用。

Resource	Estimation	Available	Utilization %
LUT	295	20800	1.42
LUTRAM	12	9600	0.13
FF	530	41600	1.27
BRAM	4	50	8.00
DSP	12	90	13.33
IO	23	250	9.20

图3-5 常规双线性缩放模块资源消耗

3.2 改进双线性插值算法模块

3.2.1 改进双线性缩放算法的原理

与常规的双线性插值算法相同，该方法同样需要利用目标位置周围的四个像素点的像素值进行插值运算。然而，与常规的双线性插值不同的是，这种改进方法采用了一种更简化的计算方式。传统的双线性插值需要通过目标像素点与邻近像素点的距离计算出加权系数，然后将这些系数与相应邻近的四个像素点的像素值进行乘积再求和运算，最终得到待测图像的像素值。而改进的双线性插值则将这些加权系数取为1，直接对邻近的四个像素值进行简单的求和运算，最后通过移位运算得到目标图像的像素值。其运算公式如（3-6）所示。

$$f(x,y)=\frac{1}{4}\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}f(P_{11}) & f(P_{12})\\ f(P_{21}) & f(P_{22})\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} \tag{3-6}$$

式中 $f(x,y)$ 代表待测的目标像素值，$f(P_{11})$、$f(P_{12})$、$f(P_{21})$、$f(P_{22})$ 代表目标像素点所邻近的四个像素点的像素值。

3.2.2 改进双线性插值算法的matlab仿真

根据改进的双线性插值的原理，利用Matlab对所提出的算法进行仿真，为了避免后续的插值运算的坐标会超过原始图像的边界，会对原始图像的底层扩充一行，右侧也会扩充一列。由于所计算的像素值是基于浮点的，因此需要对图像转换为double类型，此次处理的原始图像大小为640×480，缩小后的目标图像大小为320×240，最后通过matlab对古楼、山丘、山水画、花田和动物实现改进的双线性插值算法仿真如图3-6、图3-7、图3-8、图3-9、图3-10所示。

根据第二章图像缩放后图像质量标准指标可以看出，从主观评价的角度来看，经过图像缩放处理后，改进的双线性插值和常规双线性插值算法处理后的图像是差别不大，从客观的图像质量上可以看出，峰值信噪比（*PSNR*）和均方误差（*MSE*）数值的对比上，改进的双线性插值算法与双线性插值算法处理后的图像差距非常相近，在人眼可接受的图像质量范围内，在图像边缘部分相对比较平滑，不会使人眼产生不适的感觉。

图3-6 Matlab实现改进双线性插值仿真图（古楼）

图3-7 Matlab实现改进双线性插值仿真图（山丘）

图3-8 Matlab实现改进双线性插值仿真图（山水画）

图3-9 Matlab实现改进双线性插值仿真图（花田）

图3-10 Matlab实现改进双线性插值仿真图（动物）

使用客观评价指标评估图像质量时，峰值信噪比（*PSNR*）的值越高，意味着缩放处理后的的图像质量越优；同时，均方误差（*MSE*）的值越低，表明缩放算法的性能越出色。对于三张分辨率为640×480的图像，经过双线性插值算法和改进的双线性插值算法1/2缩小处理后，输出分辨率为320×240的图像。进一步分析两种算法处理后图像的缩放质量，*PSNR* 和 *MSE* 如表3-1所示。比较两种缩放算法处理后不同图像的 *PSNR* 和 *MSE* 值，可以得出它们之间的差距并不明显。因此，综合考虑主观和客观图像质量标准指标后发现，双线性缩放算法与改进的双线性插值算法处理后的图像效果相差无几。

表3-1 两种算法的*PSNR/MSE*

图像类型（*PSNR/MSE*）	双线性插值算法	改进双线性插值
古楼	33.388/29.739	33.362/29.983
山丘	35.375/18.862	35.321/19.098
山水画	33.360/29.994	33.330/30.205
花田	35.349/18.976	35.367/18.894
动物	33.410/29.653	33.425/29.549

3.2.3 改进双线性插值算法FPGA实现

跟双线性插值相似，在进行改进双线性插值运算过程也需要邻近的四个像素，所以需要缓存两行的像素，本设计使用了双端口BRAM对输入图像的像素值进行行缓存，所输入的图像分辨率为640×480，即一行为640个像素点，所以所要缓存一行的像素至少需要640个BRAM空间。另外BRAM每行设置了1024个地址空间，即BRAM最多可缓存行像素数量在1024以内的图像，然后将其深度设置为4096，这样可以缓存四行的像素。BRAM的空间示意图如图3-11所示。

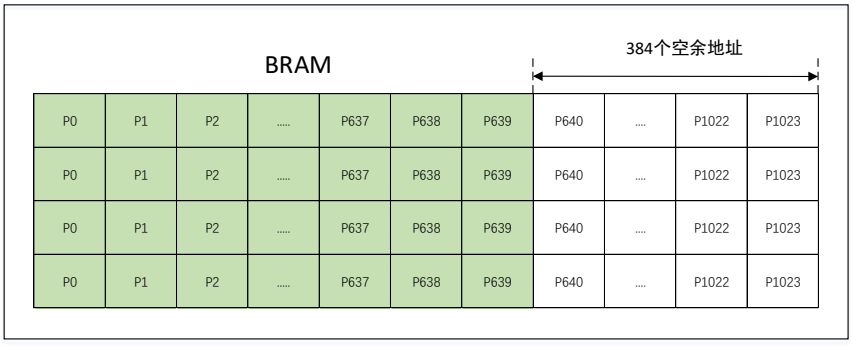


图3-11 BRAM空间示意图

改进的双线性插值会从BRAM中读取目标像素点邻近四个像素，为了有效地缓存这四个像素，采用了四个BRAM对输入图像进行行缓存，每个BRAM分别保存邻近4个像素中的一个，对于BRAM的行数据储存方式如图3-12所示。

图3-12 BRAM的行储存方式

将原始图像输入到BRAM之前，需要将存入的图像像素地址与BRAM相对应，根据原始图像的场同步信号*per_img_vsync*和行同步信号*per_img_href*，需要对原始图像进行列统计（*img_hs_cnt*）和行统计（*img_vs_cnt*），对原始图像的宽度和高度进行计数，BRAM地址的计算如公式（3-7）所示。

$$Bram\_waddr = \{img\_vs\_cnt[2:1],10'b0\} + img\_hs\_cnt \tag{3-7}$$

式中 img_hs_cnt 代表行内像素的地址，*img_vs_cnt*[2:1]代表不同行的基地址。

当原始图像的每行像素值存入到BRAM后，将行统计信号img_vs_cnt作为标签储存在异步FIFO中，在后期进行改进双线性插值的缩放处理的时候，将会根据BRAM中是否有算法中所需要的两行像素值。在进行改进双线性算法之前，需要计算原始图像与目标图像在水平和垂直方向上的比率，即目标图像映射到目标图像的坐标步进，分别为*c_x_ratio*和c_y_ratio。实验采用的原始图像分辨率为640×480，将比率设定为16位小数，目标图像的分辨率为1024×768，其水平方向上的比率*c_x_ratio*和垂直方向上的比率*c_y_ratio*，如式（3-8）和式（3-9）所示，式中*c_src_width*和*c_src_heigth*代表原始图像的宽度和长度，*c_dst_width*和*c_dst_heigth*代表目标图像的长度和宽度。

$$\begin{aligned} c\_x\_ratio &= \text{floor}\left(\frac{c\_src\_img\_width}{c\_dst\_img\_width}\times 2^{16}\right) \\ &= \text{floor}\left(\frac{640}{1024}\times 2^{16}\right)=40960 \end{aligned} \tag{3-8}$$

$$\begin{aligned} c\_x\_ratio &= \text{floor}\left(\frac{c\_src\_img\_heigth}{c\_dst\_img\_heigth}\times 2^{16}\right) \\ &= \text{floor}\left(\frac{640}{1024}\times 2^{16}\right)=40960 \end{aligned} \tag{3-9}$$

对于目标图像的坐标(x_cnt, y_cnt)和待测图像映射到原始图像的坐标(x_dec, y_dec)是通过状态控制器负责完成的，状态控制器的流程如图3-13所示。

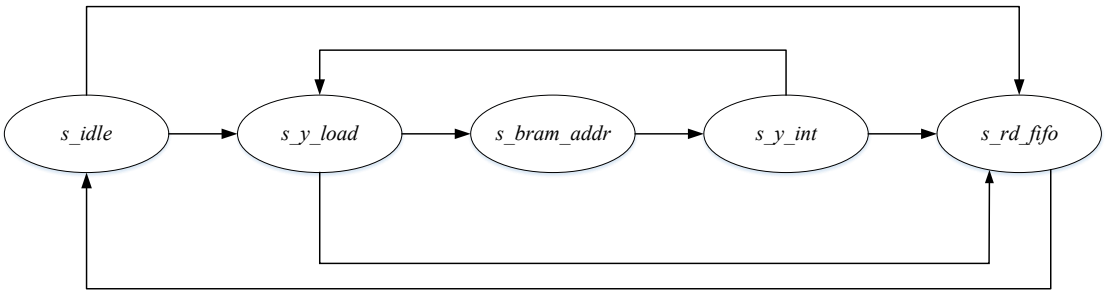


图3-13 状态控制器流程

从图3-11可以看出，开始当状态处于s_idle时，在FIFO缓存中有数据时，倘若FIFO中的原始图像行统计img_vs_cnt不为零处于计数状态，并且已经完成对目标图像最后一行的像素图像插值处理，即列计数y_cnt等于目标图像的高度，则控制器状态机进入*s_rd_fifo*状态，如果没有满足，则进入*s_y_load*状态。

在控制器状态机处于*s_y_load*状态时，获取待测图像映射到原始图像的纵坐标y_dec进行计算其计算结果如公式（3-10）所示。

$$y\_dec = y\_dec[26:16] + y\_dec[15:0] \tag{3-10}$$

$y_dec[26:16]$代表横坐标的整数部分，$y_dec[15:0]$代表横坐标的小数部分，如果y_dec的值达到了原始图像行统计img_vs_cnt则代表BRAM内已经缓存了改进双线性插值算法中所需要的两行像素的数据，状态机就会进入s_bram_addr状态，如果没能满足则进入s_rd_fifo状态。

在控制器状态机处于s_bram_addr状态时，目的是产生原始图像在BRAM

中的横坐标 x_cnt，并且生成待测像素映射在原始图像的像素横坐标 x_dec，完成操作后状态机就会进入 s_y_inc 状态。

在控制器状态机处于 s_y_inc 状态时，产生待测图像的纵坐标 y_cnt，还有生成待测像素的像素点映射到原始图像的纵坐标 y_dec ,当 y_cnt 计数到待测图像的最后一行的时候，控制器状态机则进入 s_rd_fifo 状态，如果没有满足，控制器状态机则进入 s_y_load 状态。

在控制器状态机处于 s_rd_fifo 状态时，存储在 *FIFO* 的数据将会被读出，然后再次进入 s_idle 状态，最终会形成闭环，持续的生成待测图像的像素点坐标 (x_cnt, y_cnt)，还有待测图像映射到原始图像的坐标 (x_dec, y_dec)。

通过控制器状态机获得 (x_dec, y_dec) 后，可以取其 x 坐标和 y 坐标的整数部分 $x\_dec[26:16]$ 和 $y\_dec[26:16]$，将其作为待测像素点邻近四个像素点的左上部分像素点的像素坐标 (x_int, y_int)，并将其转换为 *BRAM* 的读地址 $bram\_addr$，$bram\_mode$ 用于标记邻近四个像素的右上角像素位于原始图像的奇数行还是偶数行，如果是奇数行，则 $bram\_mode$ 用0表示，如果是偶数行，则 $bram\_mode$ 用1表示。将 $bram\_mode$ 获得的邻近四个像素点放置在四个BRAM的读地址中，然后将邻近的四个像素从BRAM的读地址中读出，并映射到待测像素点所需的邻近四个像素的位置上。

在对原始图像进行改进的双线性插值算法处理的过程中，在获得待测点邻近的四个像素时，会出现超出图像边缘的情况，本设计会通过对边界进行复制的方法，解决图像边缘越界的情况，其操作流程如图3-14所示，其中彩色部分为图像的有效部分。

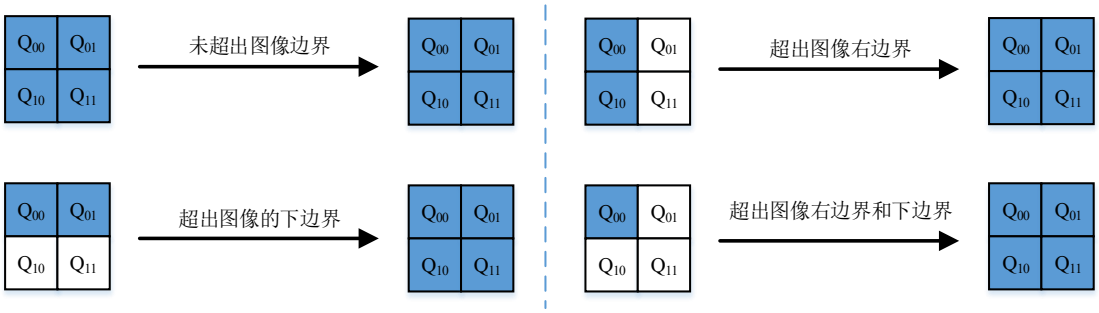


图3-14 图像边界处理流程图

最后将邻近四个像素值进行求和运算，然后对四个邻近像素值求和的结果

通过移位处理，最终获得待测目标像素的像素值。改进双线性插值算法的运算流程如图3-15所示，其中*Pixel_data*00、*Pixel_data*01、*Pixel_data*10、*Pixel_data*11分别代表邻近的四个像素点的像素值。

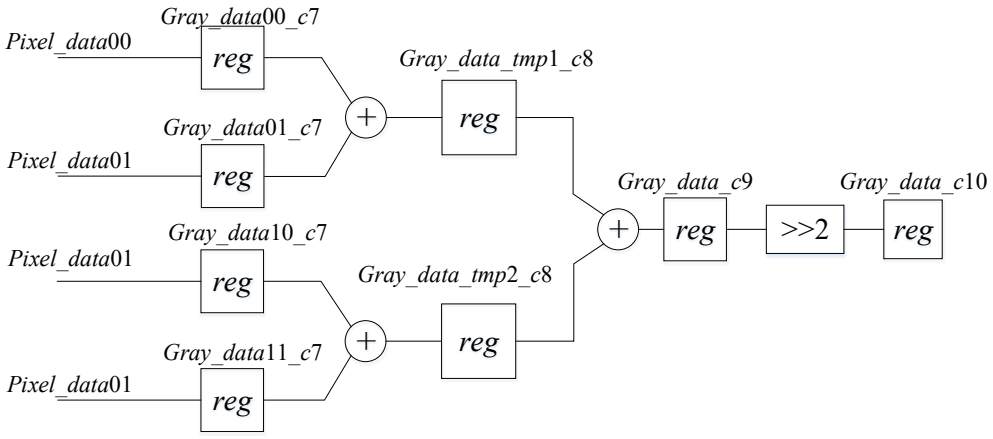


图3-15 改进双线性插值运算流程图

对改进的双线性插值算法完成FPAG的设计后，需要对本文设计的算法进行仿真验证，通过Vivado软件的Simulation工具对搭建好的改进的双线性插值算法模块进行仿真实验，以分辨率为640×480的原始图像作为输入数据，通过改进双线性插值算法放大处理后得到分辨率为1024×768的目标图像作为输出数据，对搭建好的改进双线性插值算法硬件架构进行波形仿真实验，改进的双线性插值算法的仿真波形如图3-16所示。

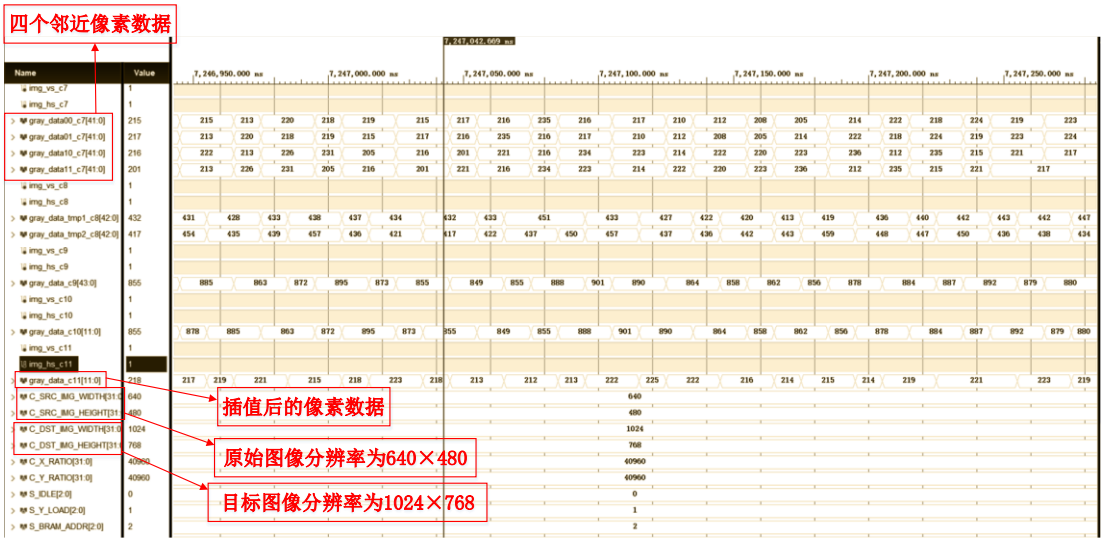


图3-16 改进的双线性插值的仿真波形

实验同双线性插值算法架构类似，最终与Matlab软件执行的改进双线性插值算法缩放后的图像像素数据保持一致，最后所设计的改进的双线性插值模块

的硬件资源消耗如图3-17所示。

Resource	Utilization	Available	Utilization %
LUT	177	20800	0.85
LUTRAM	12	9600	0.13
FF	370	41600	0.89
BRAM	4	50	8.00
IO	23	250	9.20

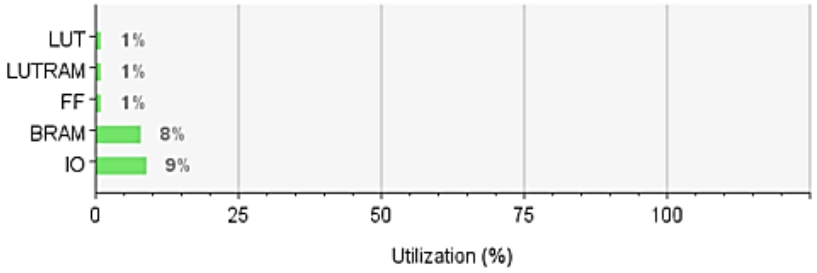


图 3-17 改进双线性插值缩放模块的资源消耗

3.3 资源消耗对比

根据本文的设计内容，主要搭建双线性插值算法硬件架构和改进的双线性插值算法硬件架构，对常规的双线性插值算法的硬件架构是由数据储存模块、插值参数生成模块和插值运算模块构成，对于改进的双线性插值算法的硬件建构与双线性插值算法的硬件架构类似，在插值运算的过程中没有使用乘法器，而是通过取待测像素点邻近的4个像素点像素值的平均值，通过移位操作处理获得待测像素点，双线性插值算法的硬件架构和改进的双线性插值算法的硬件架构的底层资源消耗对比如表3-2所示，在LUT查找表、FF触发器硬件资源消耗上分别减少了40%和30%，改进的双线性插值模块没有使用到DSP数字信号触发器。

表 3-2 两种算法硬件架构底层资源消耗对比

	LUTS	LUTRAM	FF	DSP
双线性插值算法	295	12	530	12
改进双线性插值算法	177	12	370	0

3.4 本章小结

本章在Matlab软件实现改进双线性插值的实验仿真，将分辨率为640×480的原始图片通过改进的双线性插值进行1/2的缩小处理获得目标图像，然后通过主观和客观的图像质量评价标准判断，最后显示的图像质量与常规双线性插值处

理后的图像相似，在人眼的承受范围内。本章重点搭建了双线性插值模块和改进双线性插值模块的硬件架构，改进的双线性插值没有使用乘法器进行插值实现，最后通过移位处理获得最终的目标像素值，然后使用Vivado软件对两个算法模块进行行为级仿真，并通过综合实现得到两个模块的底层资源消耗，结果显示，改进双线性算法模块有效地减少了资源消耗。

第4章 视频缩放系统功能验证平台设计

上一章节主要介绍双线性插值模块和改进的双线性模块设计的内容，并详细对两个模块进行了行为级仿真和硬件资源消耗的情况的对比，接下来进一步对改进的双线性插值算法进行验证，搭建了基于FPGA实时视频缩放系统的验证平台，本文通过使用Verilog硬件描述语言设计缩放系统中各个模块，并利用FPGA板载逻辑分析仪ILA对这些模块的功能进行了验证和分析。该系统采用的核心芯片是Xilinx公司出品的Artix-7系列，芯片型号为xc7a35tfgg484-2，缩放模块时钟为148.5MHz，实现实时视频从原始分辨率为640×480到分辨率为1920×1080之间的缩放。本文设计的系统验证平台的开发环境是vivado开发软件工具，在整个验证系统中，分为时钟模块、ov5640摄像头图像采集模块、图像缩放模块、DDR3图像缓存模块和HDMI视频显示模块。

4.1 视频缩放系统验证平台

本文提出了一种实时有效的均值双线性插值模块的方法，减少了硬件资源的消耗，验证平台的设计流程如图4-1所示，在vivado软件中搭建视频缩放系统，整个设计框架完全遵循自上而下的设计，对均值双线性算法进行硬件设计，并对其进行验证仿真。系统中的每个模块采用verilog硬件描述语言定义，并使用vivado软件进行了模块的逻辑功能仿真分析以及综合仿真。最后得出实时缩放系统缩放后的图像效果，对改进的图像缩放算法进行了验证。

整个缩放系统的验证平台图4-1所示，时钟模块（clk_wiz_0）是为HDMI顶层模块、DDR控制模块、图像缩放模块以及ov5640驱动模块提供驱动时钟。ov5640驱动模块负责传感器初始化的启动和完成。传感器初始化完成后，收集到的数据会转化成帧的格式存入DDR3储存器中，然后通过乒乓操作将图像数据导入图像缩放模块，根据相应的比例进行缩放。随后，图像缩放模块缩放后的图像数据转化成VGA视频格式，并传输至HDMI模块中。HDMI顶层模块从图像缩放模块读取图像数据，并命令显示器进行显示。只有在DDR3和传感器初始化完成后，图像数据采集模块才会启动数据输出。这样做是为了避免在初始化过程中将数据写入DDR3。整个系统完成了图像数据的采集、图像数据缓存、

图像缩放和视频图像显示的过程。

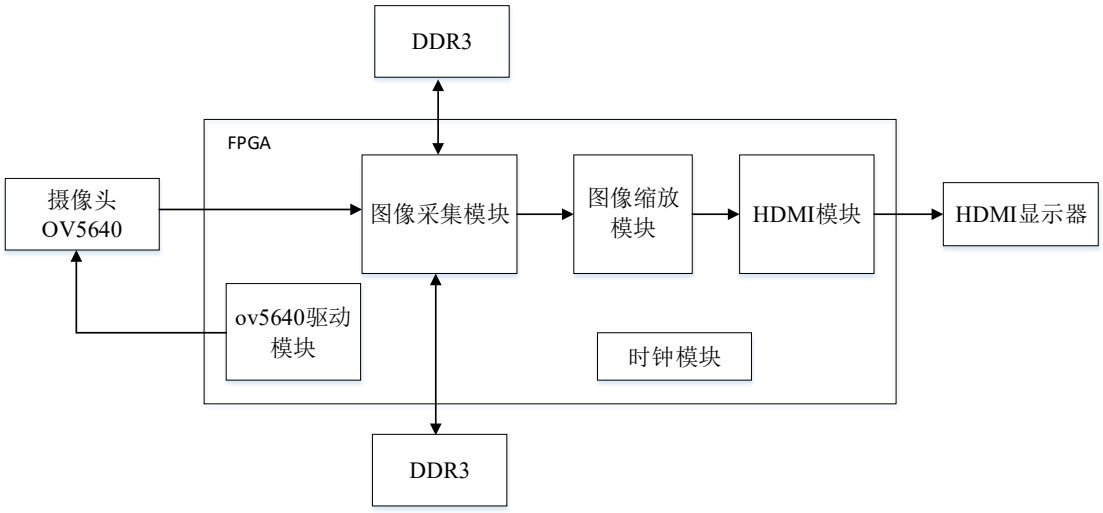


图4-1 整个缩放系统的验证平台

视频图像缩放系统所需要的模块如下：

（1）时钟模块：时钟模块（clk_wiz_0）为HDMI模块、DDR3模块、缩放模块、ov5640驱动模块以及HDMI模块提供时钟信号。

（2）ov5640驱动模块：此模块运用了串行摄像头控制总线（SCCB，Serial Camera Control Bus）接口总线技术，以实现在像素时钟同步的条件下，有效捕捉并同步来自传感器的输出信号[52]。这些信号包含了行场同步信号以及8位宽的数据信号。其核心目标在于将这些信号转化为符合DDR3内存读写操作标准的控制信号，具体包括激活写操作所需的写使能（WE）信号，以及扩展至16位宽度的写数据信号。

（3）图像采集模块：主要负责获取ov5640摄像头采集视频图像，并将采集的RGB565格式的图像数据传输至外设DDR3中储存。

（4）图像缩放模块：对ov5640摄像头采集后的图像视频数据存储至外设的DDR3存储器中，以乒乓操作的方式与图像缩放模块进行数据交互，在图像缩放模块对图像进行缩放处理，在缩放处理阶段，通过均值双线性插值算法搭建硬件设计，并将处理过的图像缩放数据传输到HDMI模块。第三章的详细研究图像缩放处理模块中的计算方法，特别是邻近四个像素的均值方法。该研究旨在深入了解和优化图像缩放的计算过程，以减少硬件资源。通过采用邻近四个像素的均值方法，研究致力于寻找更有效的计算方式，以改进图像缩放的效率，具体研究内容放在第三章。

（5）DDR3控制器：DDR3控制器在系统中负责管理读写控制器模块，与DDR3片外存储器之间的数据交互。它的主要功能包括有效地控制图像传感器产生的图像数据的读写操作。通过使用缓存DDR3控制器可以提高存储器访问效率，优化数据传输速度。在这个过程中，图像传感器生成的图像数据被传送到DDR3片外存储器，通过MIG IP核实现复杂的读写操作。先入先出存储器（First Input First Output，FIFO）也被整合进系统[53]，提供了一个有序的缓冲区，有助于平衡数据流，确保数据传输的顺畅。用户接口通过这一组件与系统进行数据交互，提供了对图像数据处理过程的控制和监控。整个系统中这些组件的协同工作，实现了对图像数据的高效管理和处理。

（6）HDMI模块：HDMI模块负责传输和处理驱动信号以实现图像在HDMI显示器上的呈现。该模块会处理显示参数，确保图像呈现在屏幕上。在这个过程中，场同步信号是必不可少的，它对图像的垂直同步进行调整，保证显示效果的稳定和流畅。数据请求信号是HDMI模块向其他系统组件请求图像数据的信号，确保图像能够按时、按需传输到HDMI显示器。通过这些关键信号的协同作用，HDMI模块成功地驱动HDMI显示器，实现了对图像的实时显示[54]。

4.2 视频图像采集模块验证

4.2.1 ov5640 功能介绍及配置

ov5640传感器作为一款先进的图像传感器，在图像捕获和处理方面拥有多项关键技术。其工作机制包括控制时钟，可通过外部时钟CLK或锁相环（PLL）来同步内部运行。数模转换器和放大器负责将感光阵列捕获到的光电信号转换为模拟信号，并通过放大器提高信噪比。时序发生器以及外部时序信号（VSYNC）协调图像帧的捕获。感光阵列作为核心组件负责光信号的捕获，而SCCB接口提供了与其他组件通信的途径。光电信号转换为模拟信号后，通过数模转换器变为数字信号，然后由图像信号处理器（ISP）进行处理，包括白平衡、自动曝光等，最终形成视频数据流。寄存器和I2C协议为用户提供了配置和调整传感器参数的灵活性。两线式SCCB接口总线则用于高效地与其他系统组件进行通信。

在系统的图像采集模块中，使用ov5640传感器通过2×9排针连接到系统，实

现了与FPGA平台的接口。传感器的具体配置和使用方法可以在传感器手册中找到，这包括了对24MHz时钟晶振的使用，用于提供系统的输入时钟。整个系统的设计目的是实现图像采集并在FPGA平台上进行视频缩放。通过ov5640传感器捕获图像，并通过2×9排针连接到FPGA，系统通过借助传感器手册中的相关信息进行配置。24MHz时钟晶振作为输入时钟，为系统提供精确的时序信号，确保图像采集和处理的准确性和稳定性。

在ov5640系列传感器中，寄存器地址是关键的配置参数，通过它们用户可以定制各种功能，其中之一是配置像素格式。RGB555系列、RGB565系列和YUV系列等不同的像素格式提供了灵活的选择，以满足各种应用场景的需求。原始图像数据（RAW）输出为用户提供了未经压缩处理的高保真图像，本文选择了RGB565格式，ov5640摄像头的部分寄存器配置如表4-1所示，ov5640摄像头采集的固定分辨率为640×480，通过地址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，用十六进制表示：0280×01e0，将它们的高位和低位分别配置在相应的寄存器中。

表4-1 ov5640摄像头部分寄存器配置

地址	寄存器名称	数值	相关描述
0x3008	系统控制	0x02	Bit[7]：软件复位 Bit[6]：软件电源休眠
0x3808	水平输出像素高位	0x02	Bit[3:0]：DVP输出水平像素点数高4位
0x3809	水平输出像素低位	0x80	Bit[7:0]：DVP输出水平像素点数低8位
0x380A	垂直输出像素高位	0x01	Bit[2:0]：DVP输出垂直像素点数高3位
0x380B	垂直输出像素低位	0xe0	Bit[7:0]：输出垂直像素点数低8位
0x380C	总水平尺寸高位	0x03	Bit[3:0]：总水平尺寸高4位
0x380D	总水平尺寸低位	0x20	Bit[7:0]：总水平尺寸低8位
0x380E	总垂直尺寸高位	0x02	Bit[7:0]：总垂直尺寸高8位
0x380F	总垂直尺寸低位	0x0d	Bit[7:0]：总垂直尺寸低8位
0x4300	格式控制	0x61	Bit[7:4]：数据输出格式 Bit[3:0]输出顺序
0x3035	PLL控制分频1	0x11	Bit[7:4]：系统时钟分频器 Bit[3:0]：MIPI的比例分频器
0x3036	PLL控制分频2	0x3F	Bit[7:0]：PLL倍频

地址为0x380C-0x380F寄存器中配置的数字对应的是总水平尺寸800（0x0320）总垂直尺寸525（0x020d）；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，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0x3F转为十进制是63，$800KHz\times63\approx50MHz$，满足分辨率为640×480所需要的像素时钟大小。

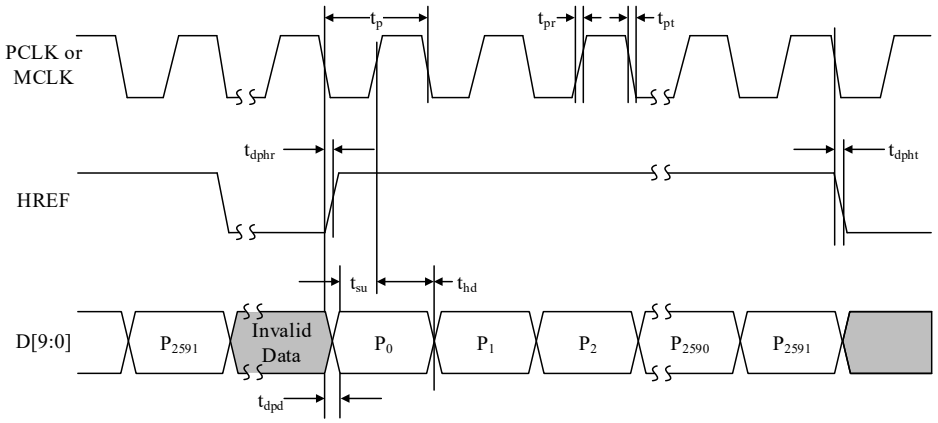


图4-2 ov5640传感器行时序图

由图4-2所示，图像的像素值采用RGB565格式，每个像素以16位表示。一个像素时钟规定了图像数据的传输速率，而数据可以是8位或10位，这提供了在不同应用场景中更灵活的选择。其中$t_p=2*t_{pclk}$表示每像素的颜色输出占用两个字节的存储空间，这对于图像数据的高效处理至关重要。图像大小和帧率是决定图像质量和显示效果的重要标准，这些参数的准确控制能够确保图像清晰度和流畅性。在此设计方案中，将ov5640系列传感器设置为以640*480的分辨率和60fps的帧率进行图像采集。输入时钟频率为24MHz，摄像头的像素输出时钟则为50MHz。采集一帧图像所需要的时间通过公式（4-1）计算得出，采集一帧图像所需要的时间为16.8ms，即采集图像的帧率为60Hz

$$\begin{aligned} t_f &= 800\times525\times t_p = 800\times525\times2t_{PCLK} \\ &= 420000\times2\times\frac{1}{50MHz}s = 16.8ms \end{aligned} \tag{4-1}$$

4.2.2 视频图像采集模块设计

图4-3展示了本文设计的图像传感器的框图。在本文设计中，为了配置图像传感器，采用了I2C驱动，其传输协议与SCCB协议相似，用16位表示寄存器地址。ov5640驱动模块负责传感器初始化和图像数据的获取，一旦初始化完成，图像采集模块将采集的视频像素数据转换成帧格式数据，然后将其传递给视频图像缩放模块进行后续图像缩放。

ov5640图像采集模块框图如图4-3所示，ov5640顶层模块主要是由ov5640驱动模块和图像采集模块组成。

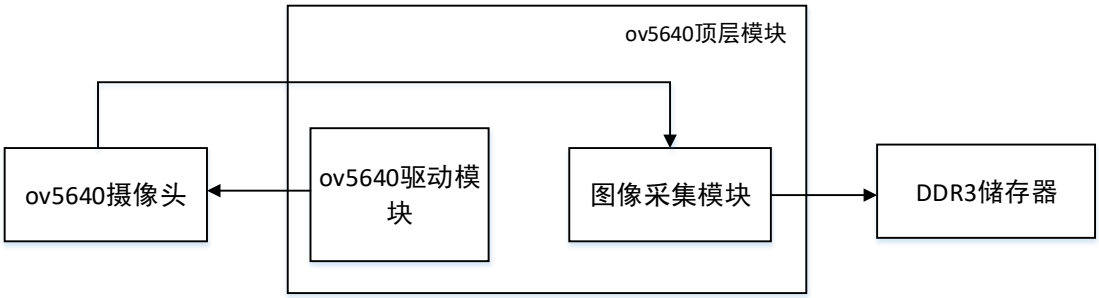


图4-3 ov5640图像采集模块设计框图

其中ov5640驱动模块主要包含I2C驱动模块和I2C配置模块，负责完成ov5640摄像头初始化。

I2C驱动模块：I2C驱动模块的主要任务是管理ov5640传感器中的SCCB总线。在配置ov5640传感器的寄存器时，可以根据I2C驱动模块的接口进行配置。在I2C驱动模块中，采用了一种基于状态机的方法来控制数据传输的过程，状态机的状态跳转图如图4-4所示，该过程可以通过一个具有8个不同状态的跳转图来展现。

I2C驱动模块中8个不同状态的跳转如下所示：

空闲状态(st_idle)：这是起始状态，等待激活信号(i2c_exec=1)来启动操作。

发送设备地址状态(st_sladdr)：收到激活信号后，状态机进入此状态，准备发送设备地址。

发送字地址的高8位状态(st_addr8_hi)：在这个状态下，如果地址是双字节的，先发送字地址的高8位。

发送字地址的低8位状态(st_addr8_lo)：这个状态用于发送剩余的低8位字地址。写数据状态(st_data_wr)：如果操作wr_flsg=0为写入，状态机转移到此状态。

发送器件地址读状态(st_sladdr_rd)：如果操作wr_flag=1为读取，状态机会先转移到这个状态来发送设备地址。

读数据状态(st_data_rd)：在这个状态下，状态机接收来自设备如ov5640传感器的数据。

完成状态(st_done)：操作完成后，状态机进入此状态，发出操作完成信号并重置I2C总线为下一轮数据传输的空闲状态。

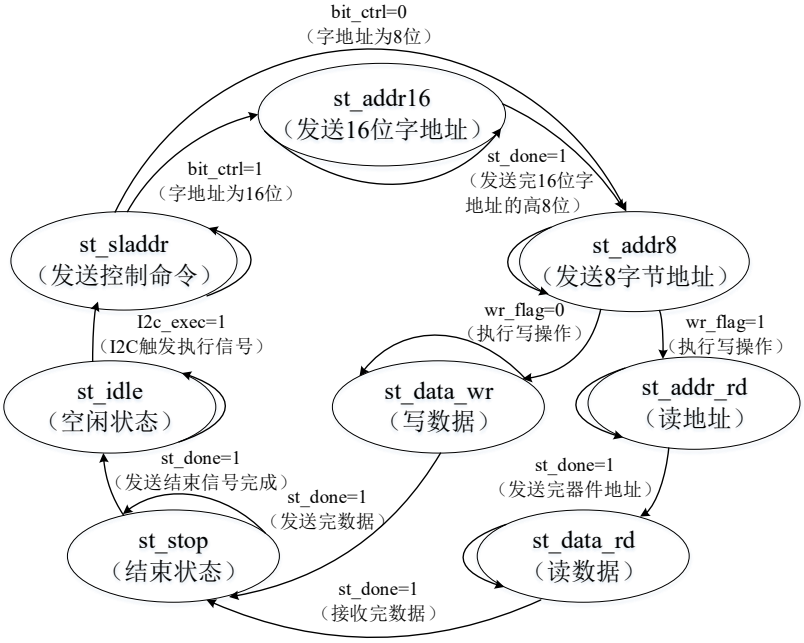


图4-4 I2C驱动模块状态机跳转图

I2C配置模块：I2C配置模块提供IIC驱动模块所需的操作时钟，需设定寄存器地址、数据和控制信号，处理ov5640传感器的寄存器地址和数据，发送执行指令的控制信号至I2C驱动模块，实现对ov5640传感器的初始化操作。

图像采集模块：图像采集模块中的摄像头模块通过像素时钟驱动同步数据采集，整合行场同步信号及8位数据，经寄存器配置确保图像数据准确获取。配置稳定还需等待10帧，保证设置生效。随后开始采集，内部将8位数据转换成16位以匹配RGB565格式，byte_flag标志格式转换完毕。

根据上述的ov5640模块的描述，本文采用verilog硬件语言对ov5640模块进行设计，当ov5640摄像头的场同步信号cmos_frame_vsync为低，行同步信号cmos_frame_vsync为高时，这表示一帧图像中的某一行像素数据正被输出给图像采集模块，vid_cmos_vld视频采集的数据有效信号为高时候，视频输入的数据为有效值。在图像传感器上电后需要用到延时计时器给配置寄存器20毫秒的延时

时间，这个延时的目的是确保所有的寄存器能够在这段时间内复位到默认状态。在模块设计完成后，通过ILA对ov5640摄像头图像采集模块进行数据抓取，分析结果显示所设计的每个模块功能都正常运行，具体结果如图4-5所示。

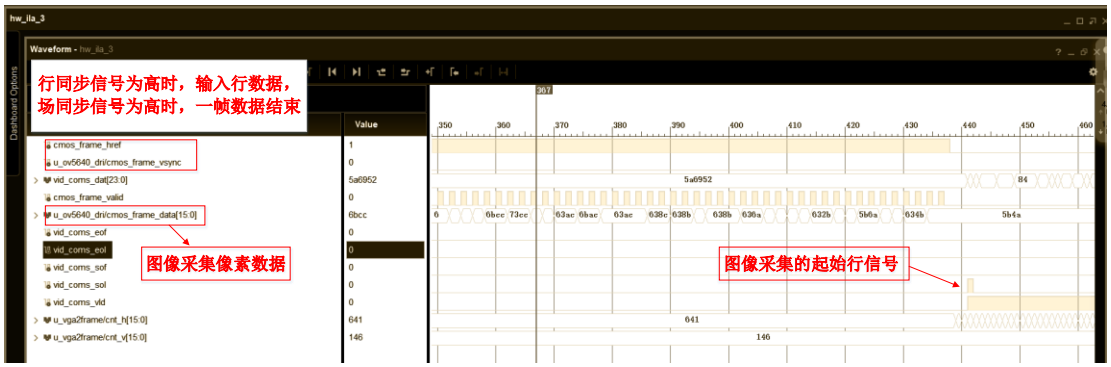


图4-5 图像采集模块ILA逻辑分析图

4.3 DDR3 数据缓存模块验证

由于 FPGA 开发板的内存有限，本文选用的 FPGA 内存资源为4860Kbit，而一帧视频图像的容量为1920×1080×16＝33.1776*Mbit*，ov5640摄像头视频采集的帧率为60Hz，所需的带宽为1990.656Mbit/s，需要外接存储器对像素数据进行储存，本文设计选用的是2个外设储存器为 DDR3，DDR3的数据位宽为16bit，其容量为4096Mbit，带宽为25.6Gbit/s，可以满足本文所需的视频传输。DDR3数据缓存模块如图4-6所示。通过 MIG 控制器 DDR3读取来自视频缩放模块的图像数据，控制图像数据写入和读取是由 FIFO 控制模块负责，FIFO 模块通过改变时钟和数据位宽，有效解决了图像缩放模块与 DDR3之间数据传输产生的跨时钟域的问题，然后图像数据通过 HDMI 传输至显示器显示。

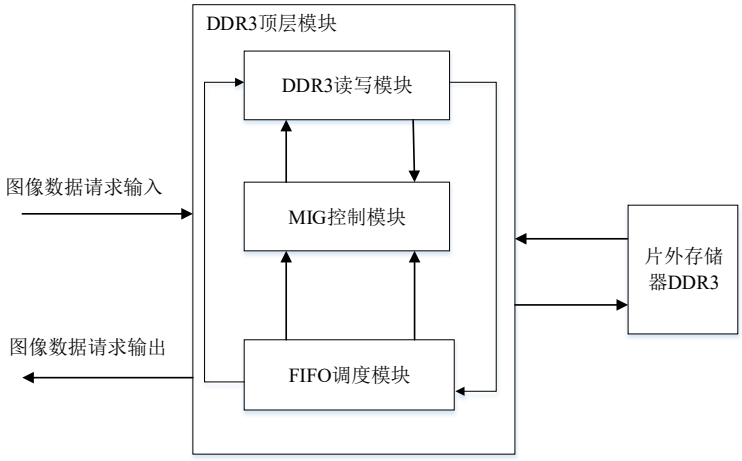


图4-6 DDR3顶层模块框图

4.3.1 FIFO 调度模块

整个DDR3数据缓存模块内含有数据输入、数据缓存和数据输出三个时钟域，数据在多个时钟域内传输容易出现数据丢失或传输错误的情况，为了解决这个问题，本文使用异步FIFO的方法，不仅可以切换成不同的时钟域，还可以进行位宽的改变。

FIFO调度模块如图4-7所示，FIFO调度模块包含了两个FIFO缓冲，一个用于读取（rd_fifo），另一个由于数据写入（wr_fifo），读取FIFO（rd_fifo）缓冲区暂时存储从图像采集模块获取的像素数据。写入FIFO（wr_fifo）缓冲区存放需要写入DDR3缓存模块的像素数据。MIG与rd_fifo和wr_fifo都有接口，管理着对读写操作必需的时序和控制信号。它确保数据能够按正确的顺序从缩放模块中读取数据和向DDR3中写入数据。

为了避免FIFO数据溢出和数据缺失，设计中FIFO调度模块将DDR3写请求给予优先处理。当FIFO内的数据超过设定阈值时，立刻发出DDR3写请求信号，同时更改写入地址，以便通过调整MIG的时序控制数据写入DDR3存储器。反过来，当FIFO内的数据量未达到预设阈值时，会在更新读取地址的同时发出读请求信号，并利用MIG的时序控制来从DDR3存储器读取数据。FIFO调度模块采用的是First Word-Fall_Through模式，这是一种常见的FIFO调度策略，其中最显著的特点是数据使能和读使能同时生效，这意味着当数据到达FIFO并准备好被读取时，读取操作可以立即进行，而无需等待额外的时钟周期。

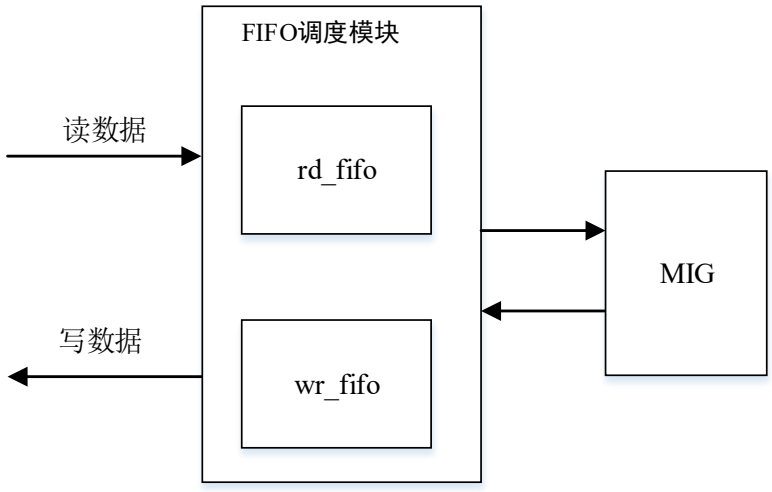


图4-7 FIFO调度模块架构

4.3.2 DDR3 储存空间管理

在本设计中，通过DDR3数据缓存模块来解决数据的存储和处理，采取了乒乓存储方法处理数据的传输如图4-8所示，这种方法通过将存储空间划分为独立的读写区域，允许数据在一个区域被处理时，另一个区域可以同时进行数据的读取或写入操作。这种流水线式的数据处理方式，提高了数据处理的效率和系统的响应速度，从而有效适应高度实时性的应用场景，如视频处理、实时图像分析等，都能够获得连续不断的数据支持。因此，本设计满足了图像像素数据处理的实时和连续性需求。

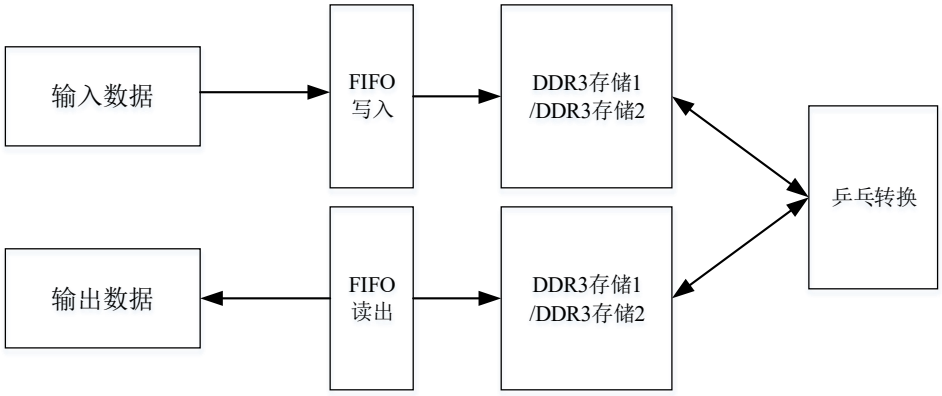


图4-8 DDR3数据储存流程图

DDR3数据缓存模块的流程图如图4-8所示，在本设计中，通过利用两片DDR3数据缓存模块并采用乒乓操作策略[55]，有效解决了在图像处理中常见的写入和读取数据速率不同步问题，从而避免了丢帧现象，确保了数据处理的高效性和流畅性。乒乓操作通过两个DDR3存储器交替缓存连续两帧图像的像素数据。这种方法结合了输入FIFO和输出FIFO，形成一个高效的流水线处理机制，不仅优化了数据的同步处理，还增强了系统处理图像数据的实时性和连续性。

具体来说，这一过程包括三个主要步骤的循环操作：首先，一个DDR3存储器被用来接收和缓存即将处理的图像数据；其次，当这个DDR3存储器中数据正在被处理时，另一个DDR3存储器则开始缓存下一帧图像数据；最后，处理完毕的数据通过输出FIFO被发送出去，为下一轮数据处理留出储存空间。这种循环操作保证了数据处理过程的高度同步，减少了处理延迟，从而满足了对实时性和连续性要求高的应用场景，如实时视频监控和动态图像分析等。

4.3.3 DDR3 读写控制状态机设计

DDR3 SDRAM通过增强数据传输速度和降低功耗，显著提升系统性能，支持更复杂的应用和数据密集型任务，确保平稳运行。因为DDR3需要进行大量两数像素数据的读写操作，所以通过编程语言去搭建DDR3数据缓存模块不仅无法保证数据传输是否准确，而且很难保障DDR3数据缓存模块的性能，鉴于这两点，本研究采用MIG IP核控制DDR3的数据交互，MIG IP核内各个端口信号和内部结构如图4-9所示。通过使用MIG IP核控制DDR芯片的端口信号，在模块设计时，只需要控制接口块的端口信号时序，接口信号会通过本地接口与储存控制器进行交互，然后通过物理接口进入到物理层，最后通过IO口控制DDR3 SDRAM的读写操作。

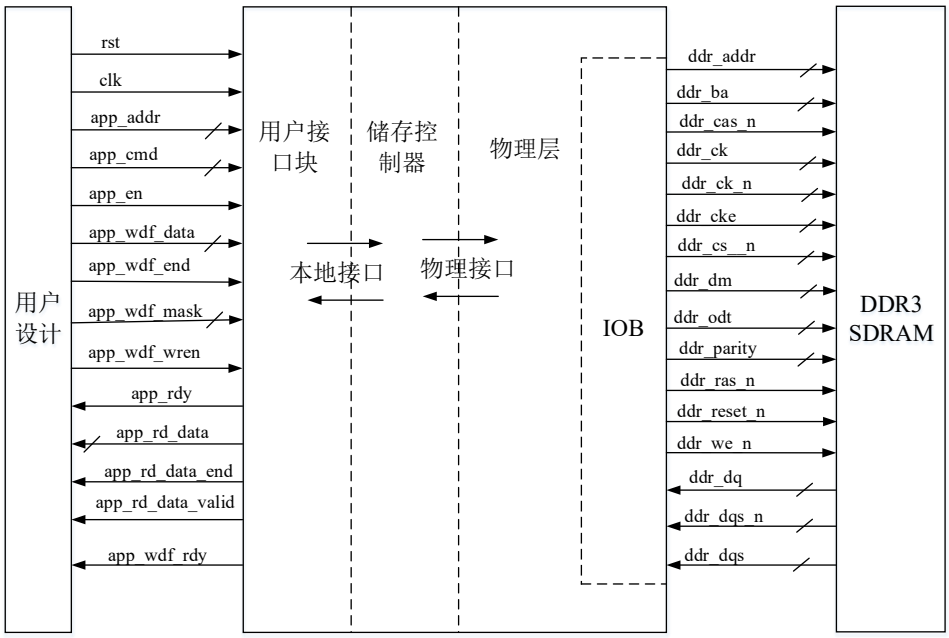


图4-9 MIG IP核结构图

根据MIG IP核结构图，MIG IP核控制端口信号的设置，在DDR3模块初始化完成后会自动完成相应的命令和配置，然后实现对DDR3的读取和写入操作。

在DDR3读写时序中，通过app_cmd执行读写命令，写入数据时，app_cmd设为0；读取数据时设为1。DDR3写时序如图4-10所示。当app_rdy高电平，表示MIG IP核可接收命令。在时钟上升沿，激活app_en并发送app_cmd和app_addr，将写命令和地址传至DDR3。设置app_wdf_wren高电平并通过app_wdf_data输入数据，完成DDR3数据的写入。

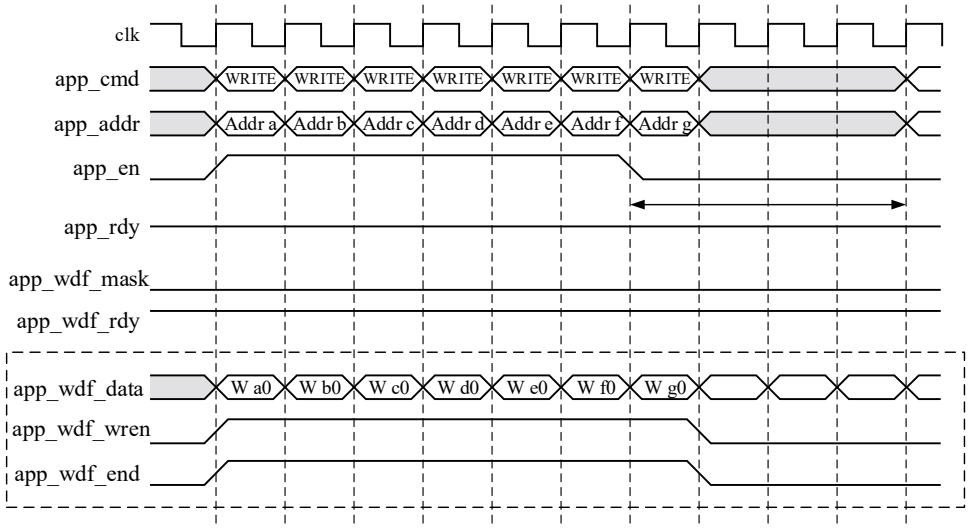


图4-10 DDR3写时序

图4-11展示了DDR3读取数据的时序。启动读命令后，将操作命令app_cmd设为1以切换至读取模式。在读取时序中，必须指定读取数据的地址信号app_addr。用户须等待app_rd_data_valid信号变为高电平，这一信号的高电平状态表明数据总线上的数据已经有效，随之完成DDR3的数据读取过程。

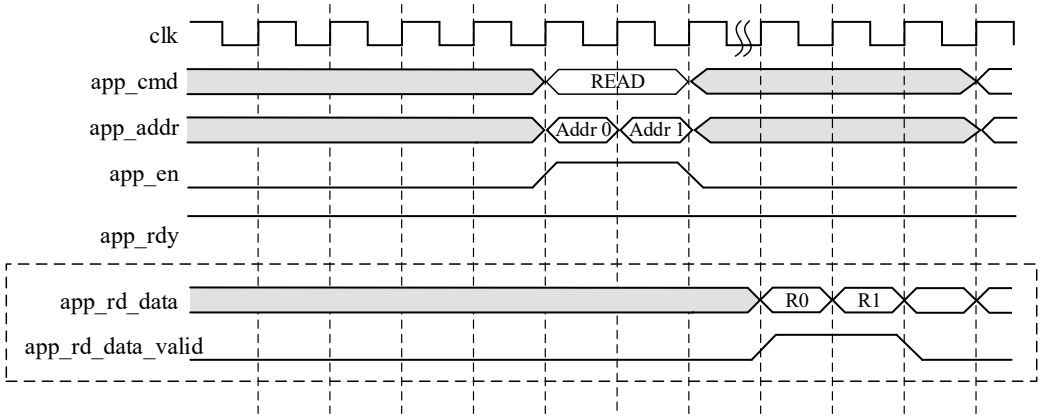


图4-11 DDR3读时序

DDR3读写逻辑的实现是采用一段式状态机实现的，其读写测试状态跳转流程如图4-12所示，在板子上电进行复位操作完后先处于空闲状态IDLE，等待初始化完成后跳转到DDR3读写状态，当DDR3读请求信号（ddr_rd_req）为高时，DDR3开始处于读出数据状态，将读命令信号核读地址信号传送到MIG IP核，当DDR3写请求信号（ddr_wr_req）为高时，DDR3开始处于写数据状态，在DDR3数据写入完成后，将重新返回到空闲状态（IDLE）。

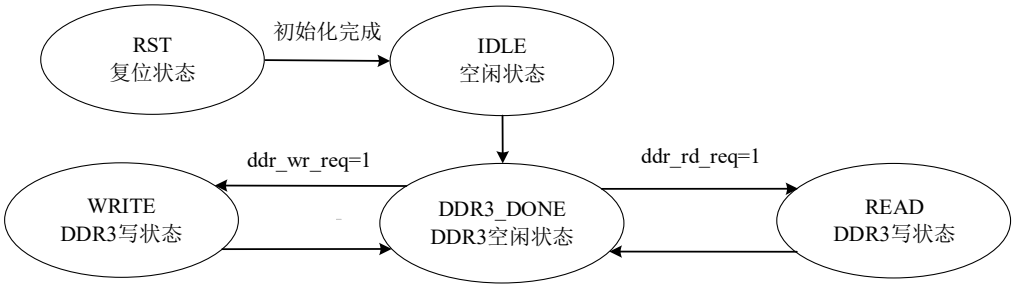


图4-12 DDR3读写状态机流程图

DDR3顶层模块中的ddr3_rw模块负责生成与MIG IP核用户接口相关的读写时序，以实现与MIG IP核的数据和信号交互，本文MIG IP核产生的时钟为100M，作为MIG IP的用户时钟，DDR3控制器的时钟为200M，MIG模块（mig_7series_0）具有两种功能：一是该模块负责FPGA板子与DDR3进行数据传输和交换，二是MIG控制DDR3所需要的各种时序，确保DDR3能够顺利的进行读写操作。

在DDR读写模块的动态数据交互中，DDR3读写测试波形图如图4-13所示。首先，init_calib_complete信号的高电平指示DDR3初始化已成功完成。读写操作通过app_cmd信号标识，其中该信号为0表示进行读取数据，为1表示进行写入数据，波形显示了读数据时的有效状态，有效数据量由wfifo_rcount（写输入端）和rfifo_wcount（读输出端）表示。最后，app_addr和app_cmd分别表示DDR3的地址和命令数据。整个过程中，MIG IP核的操作指令负责完成数据的读写以及对DDR3的控制。

图4-13 DDR3读写测试波形图

DDR3读写状态时序波形图如图4-14和4-15所示，当写地址计数器满足一次写操作条件后，握手信号命令使能app_rdy与写数据有效使能app_wdf_rdy同时为1时，系统进入DDR3空闲状态。用户端每个时钟周期传输128位数据，DDR3

物理端需分8次完成，每次传输16位地址，共8个地址完成8次传输。

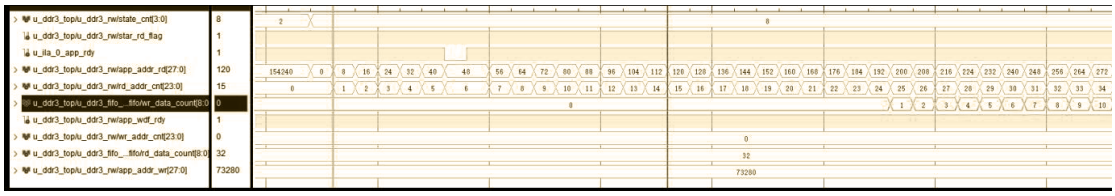


图4-14 DDR3写状态时序波形图

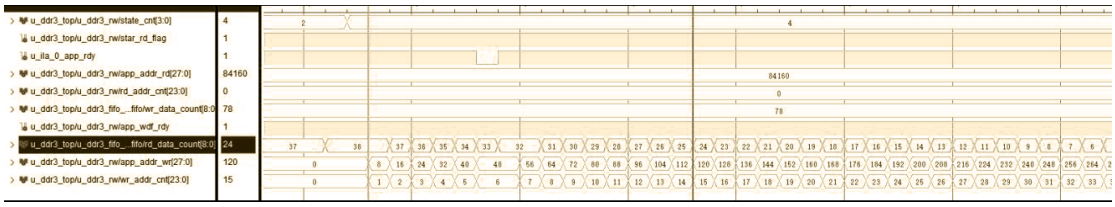


图4-15 DDR3读状态时序波形图

4.4 视频缩放模块设计和验证

4.4.1 视频缩放模块设计

视频缩放系统架构如图4-16所示，视频缩放模块可分为两个模块分别为像素2×2窗口和图像缩放模块。其中，像素2×2窗口主要负责获取图像缩放模块所需要的四个像素数据，其视频数据是ov5640传感器图像采集模块传输至该模块，在DDR3模块中缓存其像素数据，再将图像像素值传输至图像缩放模块对图像进行缩放处理，在视频缩放模块中，其中图像缩放模块的时钟为148.5MHz，能够满足最大的输出分辨率1920×1080，通过vio虚拟输入输出控制缩放模块的行和列，最后缩放后的图像数据传输至外设的DDR3模块进行缓存，最后通过HDMI数据传输模块将缩放后的视频在显示器中显示。

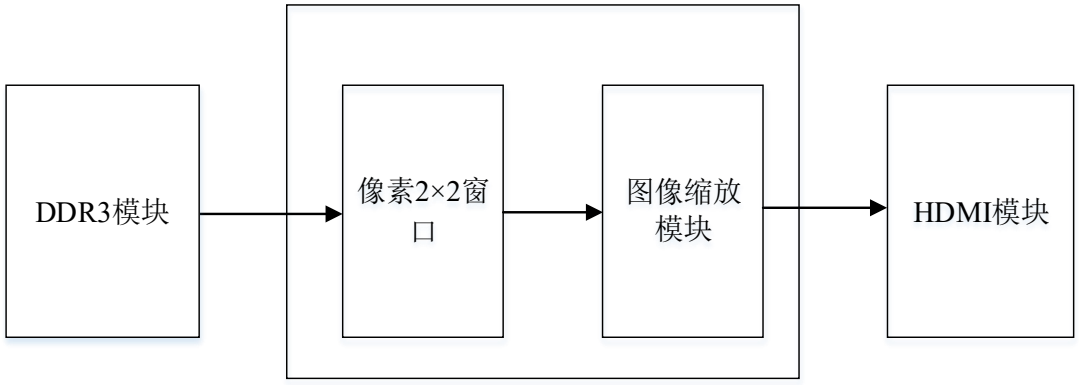


图4-16 视频缩放系统架构

4.4.2 视频缩放模块验证

ILA抓取视频缩放模块的数据波形图如图4-17所示，将视频图像分辨率为640*480放大至1920*1080，其中add_mult_r、add_mult_g、add_mult_b信号将图像数据RGB888格式中的R、G、B三种颜色数据进行缩放处理，每个颜色分量的数据位宽为8位，最后对这三种数据整合之后生成缩放后的视频数据dst_data，数据位宽为24位，数据缩放模块在coordinate_rd_en读数据使能信号为高时，数据缩放模块开始读取像素2×2窗口（u_image_2×2）中所需的视频数据coordinate_rd_data，最后得到缩放处理后的图像数据dst_dat，然后传输至HDMI模块。

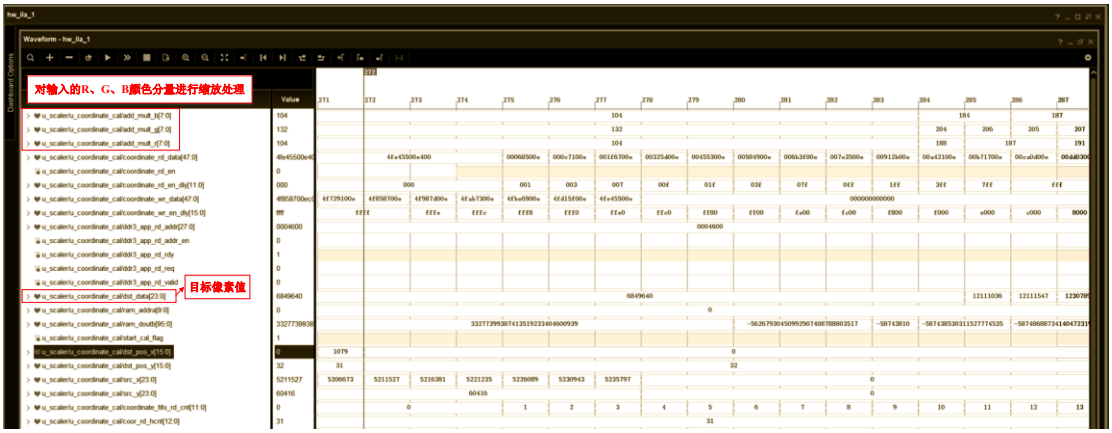


图4-17 ILA抓取视频缩放模块数据波形图

4.5 HDMI 视频传输模块设计验证

4.5.1 HDMI 接口协议

本文设计的视频传输到显示器的接口为HDMI接口，可以兼容视频和音频的信号传输，其接口标准是遵循数字视频接口（DVI）标准，但相对于DVI接口，明显减少接口连线数量，而且具有更大的数据带宽增加了数据传输量，HDMI还支持高质量的无损音频传输，目前主流的视频传输接口为HDMI 2.1，它的最高传输速率可以达到48Gbps，本文所搭建的验证平台中，传输的信号接口采用DVI接口，只进行数字视频信号的传输，不需要传输音频信号，DVI和HDMI接口都是遵循TMDS视频传输协议[56]。

如图4-18展示了TMDS的发送端和接收端连接，利用TMDS技术通过四个串行通道进行DVI和HDMI视频传输。DVI传输红、绿、蓝三色分量（RGB 4:4:4），

而HDMI除了RGB还能传输YCrCb4:4:4或4:2:2格式。第四通道是时钟通道，用于传输视频流像素时钟，确保接收端准确恢复数据。独立的TMDS连接包括编码及并串转换两阶段：首先，编码器转化视频、HDMI音频或数据及同步信号为10位字符流；然后，将10位字符流变为串行数据，通过差分通道发送。这一流程确保视频、音频和额外信息的有效传输。

对于每个24位颜色深度的像素，每个RGB通道的数据经过8B/10B编码器转换成10位像素字符，然后通过并串转换器转为串行信号，通过TMDS数据通道传输。这种10:1的并串转换使得最终产生的串行数据速率达到原始像素时钟速率的十倍。

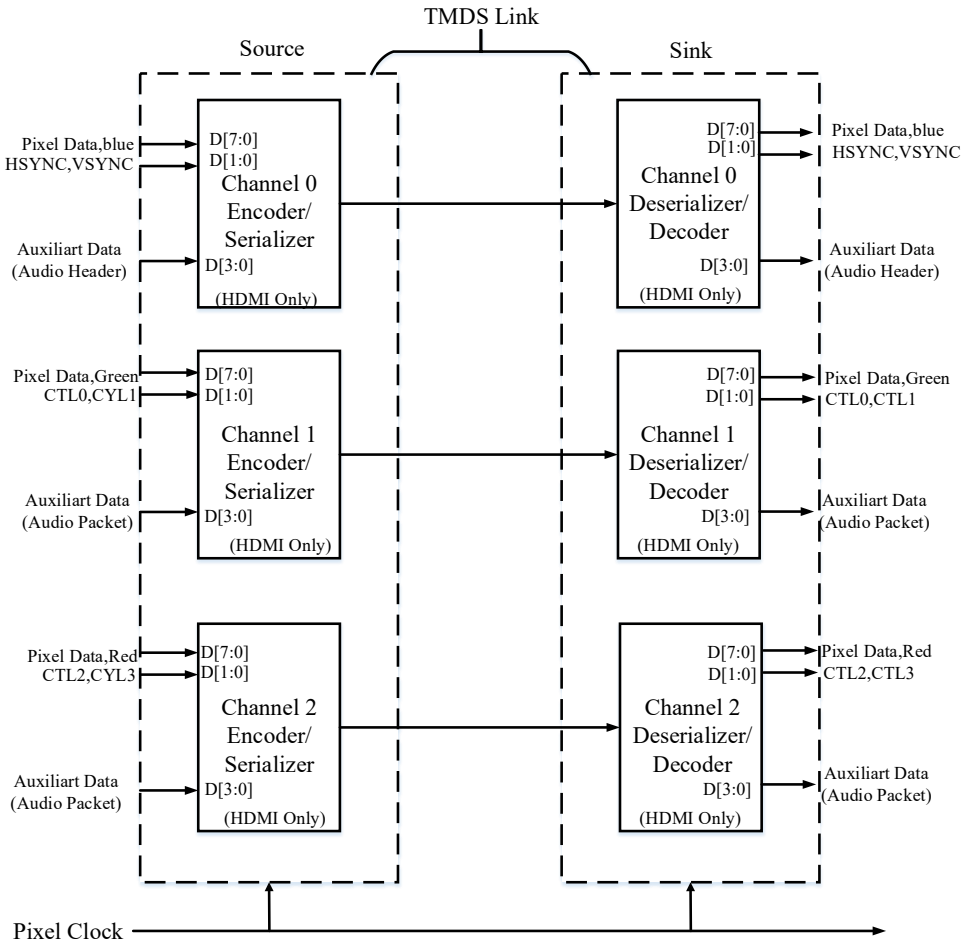


图4-18 TMDS连接示意图

4.5.2 HDMI 视频显示模块设计

本文设计的HDMI视频显示模块主要分为四个板块，分别为HDMI视频显示配置、时钟配置、HDMI视频驱动和RGB to DVI模块，HDMI视频显示模块框图如图4-19所示。

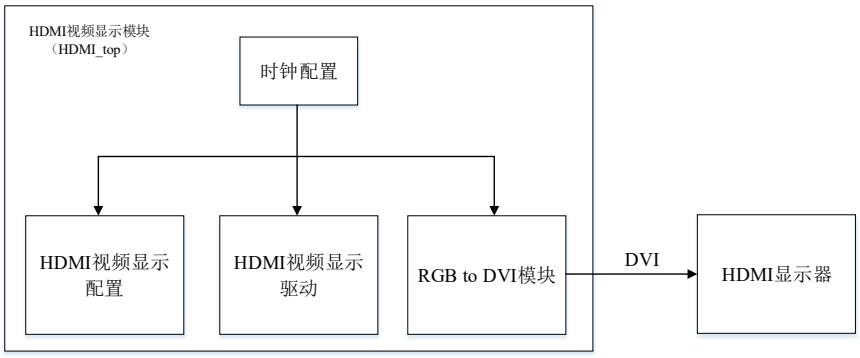


图4-19 HDMI视频显示模块框图

视频图像传送到HDMI视频显示模块后，会将摄像头采集后的16bit视频数据（RGB565格式）进行补零操作转换成24bit视频数据（RGB888格式），再由RGB to DVI模块转换为TMDS输出，设计框图如图4-20所示。

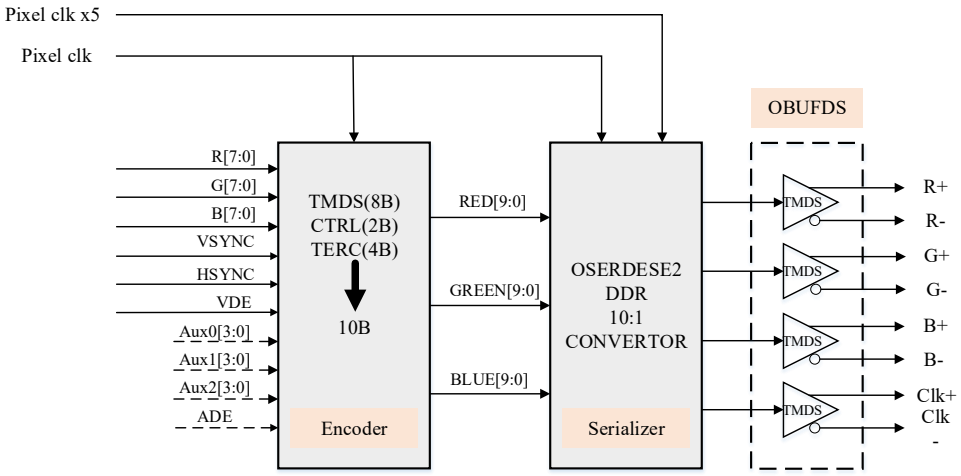


图4-20 RGB to DVI模块设计框图

在RGB to DVI模块中，Encoder模块处理数据编码，Serializer模块将编码后的数据转换为串行格式，通过OBUFDS模块转换为TMDS信号进行发送。系统使用视频像素时钟148.5MHz（Pixel Clk）及五倍频率时钟742.5MHz（Pixel Clk x5）。尽管理论上需10倍像素时钟频率进行并串转换，但OSERDESE2模块的DDR功能使5倍时钟频率即可实现双倍数据传输速率。

HDMI驱动模块产生行场信号、数据有效信号和像素坐标，输出data_req信号至端口以从DDR3控制器读取数据，ILA逻辑分析仪抓取HDMI数据的波形图如图4-21所示，所抓取的数据是从缩放模块所得到的像素数据，其中data_in为缩放后的数据传递到HDMI模块，video_vs场同步信号和video_de数据时能信号为高时，像素数据正常传输。

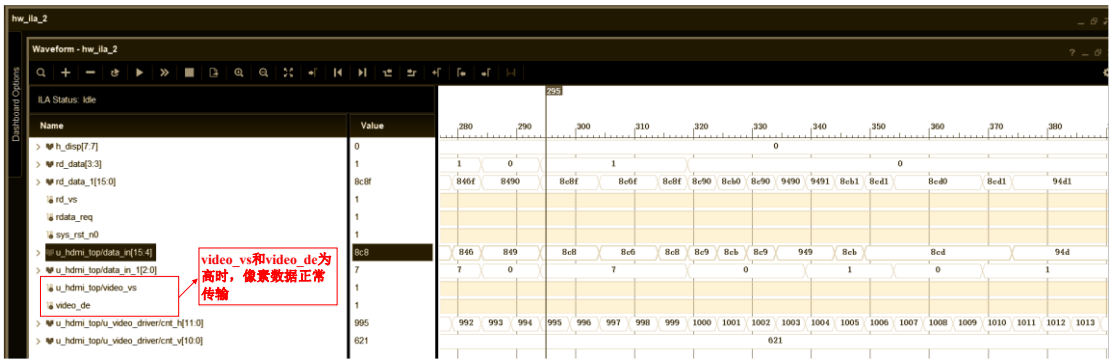


图4-21 HDMI模块ILA逻辑分析图

4.6 视频缩放系统测试

对系统各个模块的测试完成后，本节将会对整体的系统进行结果分析。本系统的视频来源是通过ov5640摄像头采集所得，传输到FPGA开发板中，本文设计选用的FPGA板子的型号为xc7a35tfgg484-2，在FPGA中通过Verilog硬件描述语言对系统中ov5640模块、图像缩放模块、DDR3数据缓存模块、HDMI数据传模块进行描述，通过vio（虚拟输入输出）控制输出图像的分辨率完成缩放操作，在对整个系统进行行为及仿真和功能性仿真后，通过JTAG下载接口将程序下载到FPGA板中，最后通过HDMI传输在显示器上实现视频的实时显示。

在进行板级验证的时候，首先是输出ov5640采集分辨率640×480视频图像如图4-22所示，通过vio虚拟输入输出控制缩放模块中视频图像的行信号和列信号，将视频缩小到分辨率为320×240，如图4-23所示。再将原视频图像放大到分辨率1920×1080，如图4-24所示，时钟模块给视频缩放模块的最大时钟频率为148.5MHz，此时缩放模块的处理帧率为60fps，能够满足实时处理视频图像缩放。将缩放后的视频图像转化成VGA视频格式，在HDMI显示器显示缩放后的视频。

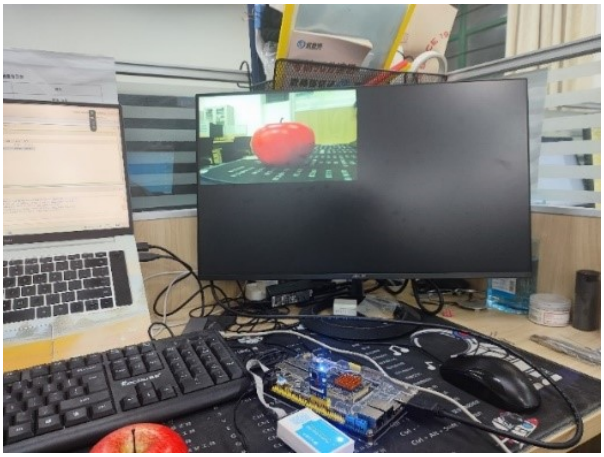


图4-22 视频图像初始分辨率640×480

图4-23 视频缩小后图像分辨率320×240

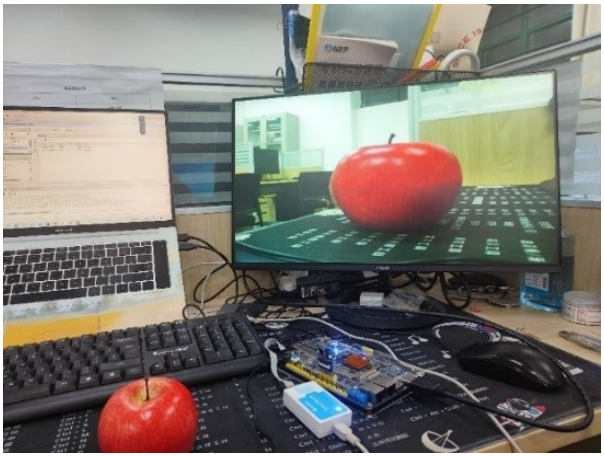


图4-24 视频放大后图像分辨率1920×1080

4.7 视频缩放系统性能测试

4.7.1 视频缩放系统资源消耗

在资源有限的FPGA板子中，资源占用率和系统功耗是衡量视频缩放系统的重要指标。在vivado软件中，搭建完视频缩放系统工程并且实现综合后，Project Summary可以观察整个系统的资源占有情况，视频缩放系统资源消耗如图4-25所示，整个视频缩放系统所占用的资源在可接受的范围内，此外，由于外接的DDR3缓存器作为外部存储像素数据的介质，所以对FPGA开发板中BRAM等存储资源的消耗不高。如表4-2所示，本文设计与其他文献所提视频缩放系统的资源消耗的对比，可以明显看出，本文设计的视频缩放系统消耗更少的硬件资源，缩放模块的的处理视频帧率为60fps，能够实时对视频进行缩放处理。

Resource	Utilization	Available	Utilization %
LUT	12757	63400	20.12
LUTRAM	1403	19000	7.38
FF	10584	126800	8.35
BRAM	21.50	135	15.93
DSP	5	240	2.08
IO	93	285	32.63
MMCM	3	6	50.00
PLL	1	6	16.67

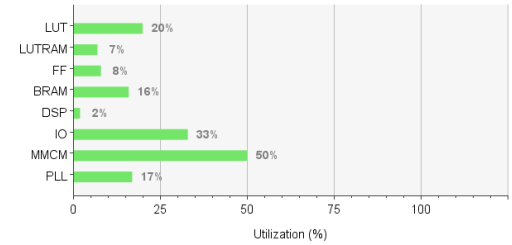


图4-25 视频缩放系统资源消耗

表4-2 本文设计与其他文献的资源对比

资源	文献[57]	文献[58]	文献[59]	本文设计
LUT	44317	43112	19335	12757
LUTRAM	14470	2850	1964	1403
FF	37370	31700	11635	10584
DSP	145	14	33	5
IO	108	140	156	93
MMCM	2	3	3	3

4.7.2 视频缩放系统功耗

在视频缩放系统工程综合完成之后生成整个系统的功耗报告，在summary界面可以查看具体的功耗信息如图4-26所示。系统总功耗为1.52W，系统结温为29.3℃，在系统的可控范围内，动态功耗占比率（dynamic）为95%，静态功耗占比率（static）为5%。

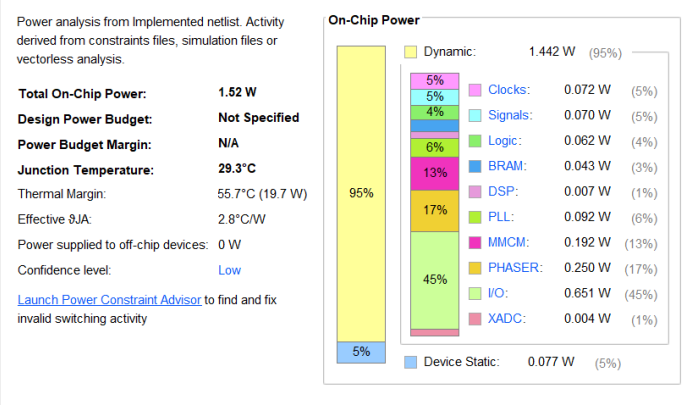


图4-26 视频缩放系统功耗信息

4.8 本章小结

本章对系统的主要模块进行了代码的设计和实现，并完成了整个系统的测试与验证，并且利用vivado软件自带的ILA逻辑分析仪对系统各个模块抓取关键信号的信息，构建FPGA视频缩放系统验证平台，实现对640×480缩放至1920×1080的视频信号进行实时处理，HDMI输出像素时钟为148.5MHz，HDMI输出1920×1080图像的帧率为60fps，其中行和场的总像素点的数量分别为2200个和1125行，符合VESA标准分辨率视频格式。最后，分析整个系统各个模块的功能能够正常运行，实现视频最后对整个系统进行板级验证和性能分析，证明了系统性能满足设计需求。

第5章 总结与展望

5.1 工作总结

本文的主要工作总结如下：

（1）缩放算法的仿真分析：对三个传统的图像算法进行研究，对三种算法利用matlab软件进行仿真分析，通过编写脚本来实现这三个缩放算法，对三张分辨率为640×480的不同图片缩小至分辨率为320×240的图片，然后评估缩放后的图像质量，结合了主观质量评价（例如观察图像的清晰度、细节保留情况）和客观质量评价标准PSNR（峰值信噪比）和MSE（均方误差），来比较不同缩放算法的性能，通过仿真可以明显看出三种算法在图像缩放质量上的差异，根据图像质量上的差异和硬件实现情况，最终选择适合本文设计的缩放算法。

（2）本文提出了均值双线性插值算法，并与常规双线新插值算法进行比较，利用Matlab软件实现均值双线新插值的仿真实验，对之前三张不同图片进行缩小操作，通过主观和客观图像质量评估标准，与双线性缩放算法处理后的图片质量进行对比，结果表明均值双线性插值算法与双线性插值算法处理后的图片质量相差不大，PSNR（峰值信噪比）都在33dB以上，在Vivado软件中搭建该两种算法模块，最后通过综合后得到底层资源消耗，在LUT查找表和DSP信号处理器件资源消耗上，均值双线性插值相对于常规的双线性插值明显减少。

（3）本文设计并验证了一个高效的视频缩放系统，通过使用Verilog语言对关键模块进行硬件描述。这些模块涵盖了视频图像采集、高性能DDR3数据缓存交互、改进的视频缩放模块，以及兼容性强的HDMI接口模块，确保了系统的整体性能和稳定性。在主要模块设计完成后，使用vivado软件自带的ILA逻辑分析仪对各个模块在线测试，观察各个模块内部数据传输情况，最终实现从原始视频分辨率为640×480放大到分辨率为1920×1080得视频信号，图像缩放模块的图像处理帧率为60帧，能够满足实时处理视频图像缩放。本文提出的均值双线性插值算法既减少了硬件资源的消耗，又保证图片的质量能够正常输出。

5.2 工作展望

本文完成了改进双线性插值缩放系统的搭建，取得了相关的成果，但本文系统设计还有些不足之处，需要加以改善和研究。

（1）本文虽然对图像缩放算法进行改进，减少了硬件资源的消耗，但对图像边缘锐化情况还没有更加有效得解决方法，可以在图像缩放设计过程中结合图像边缘情况，更好的提高图像的缩放质量。

（2）需要增加DDR3储存容量，本文采用了两个外设DDR3数据缓存芯片，在FPGA允许搭载更多的DDR3的前提下，增加外设DDR3缓存芯片在FPGA板子上，增加缓存数据带宽，视频缩放系统不仅数据传输得到大幅度提高，缩放的范围也会得到扩大。

参考文献

[1] 孙伟. 基于自适应插值算法的视频图像缩放技术及其FPGA实现[J]. 信息通信, 2015(3): 44.

[2] 樊光辉, 许茹, 王德清. 基于FPGA的高速流水线FFT算法实现[J]. 电子工程师, 2008, (03): 38-40, 53.

[3] 赵玉宇, 程光, 刘旭辉, 等. 下一代网络处理器及应用综述[J]. 软件学报, 2021, 32 (02): 445-474.

[4] 朱明达, 辛鹏, 常嘉颖. 基于FPGA的九点插值自适应图像缩放算法设计[J]. 液晶与显示, 2023, 38(08): 1075-1083.

[5] Durand X C, Faguy D. Rational zoom of bit maps using b-spline interpolation in computerized 2-d animation[J]. Comput. Graph. Forum, 2012, 9(1): 27-37.

[6] Unser, M, Aldroubi, et al. Enlargement or reduction of digital images with minimum loss of information[J].IEEE Transactions on Image Processing, 2015, 4(3):247-258.

[7] Darwish A M , Bedair M S . Adaptive resampling algorithm for image zooming[J]. IEE proceedings. Part K.Vision, image and signal processing, 2017, 144(4): 207-212.

[8] Su D, Willis P. Image interpolation by pixel-level data-dependent triangulation[J]. Computer Graphics Forum, 2016, 23(2): 189-201.

[9] Casciola G, Montefusco L B, Morigi S. Edge-driven image interpolation using adaptive anisotropic radial basis functions[J]. Journal of Mathematical Imaging & Vision, 2010, 36(2): 125-139.

[10] Giachetti A, Asuni N. Fast artifacts-free image interpolation[C]. British Machine Vision Conference. 2010: 123-132.

[11] Gilman A, Bailey D G , Marsland S .Least-squares optimal interpolation for fast image super-resolution[C]. DELTA, Vietnam, 2010. New York: IEEE, 2010: 29-35.

[12] Seong G, Jeong, Chul, et al. Motion-compensated frame interpolation based on multihypothesis motion estimation and texture optimization[J]. IEEE transactions on image processing : a publication of the IEEE Signal Processing Society, 2013, 22(11): 4497-4509.

[13] Bailey D G, Gribbon K T. Anovel approach to real-time bilinear interpolation[C].

Second IEEE International Workshop on Electronic Design,Test and Applications, 2019:126-131.

[14] 蔡念, 张海员, 张楠, 等. 基于小波的双线性插值误差补偿算法的图像放大[J]. 激光与红外, 2010, 040(005): 558-562.

[15] 龚昌来, 杨冬涛. 一种改进的双线性插值图像放大算法[J]. 激光与红外, 2009, 39(08): 899-901.

[16] 杨朝霞, 逯峰, 关履泰. 用B样条的尺度关系来实现图像任意精度的放大缩小[J]. 计算机辅助设计与图形学学报, 2001,13(9): 824-827.

[17] Chen J, Zhuo L, Wei Z, et al. Knowledge driven weights estimation for large-scale few-shot image recognition[J]. Pattern Recognition: The Journal of the Pattern Recognition Society, 2023: 142.

[18] 盛敏, 苏本跃. 基于混合插值样条的保边缘图像插值算法[J]. 计算机工程, 2011, 37(6): 218-220.

[19] Safinaz S, Kumar A V R. VLSI realization of lanczos interpolation for a generic video scaling algorithm[C]. International Conference on Recent Advances in Electronics & Communication Technology. IEEE, 2017: 17-23.

[20] Gribbon K T , Johnston C T , Bailey D G . A real-time FPGA implementation of a barrel distortion correction algorithm with bilinear interpolation[C]. proceedings of the image and vision computing new zealand, 2003: 408-413.

[21] 刘怡,黄自力,王经纬,等.FPGA双线性插值图像变换系统的设计与实现[J].中国测试技术, 2008, 34(3):3.

[22] Lin C C , Sheu M H , Chiang H K ,et al. A low-cost VLSI design of extended linear interpolation for real time digital image processing[C]. International Conference on Embedded Software & Systems. IEEE, 2008: 196-202.

[23] 张弘. 基于FPGA的视频图像处理的研究与实现[D]. 电子科技大学, 2020.

[24] 孙红进. FPGA实现的视频图像缩放显示[J]. 液晶与显示, 2010,25(1): 130-133.

[25] 田洋, 毕秀丽, 肖斌, 等. 基于离散切比雪夫变换的图像接缝裁剪篡改检测[J]. 计算机科学, 2021, 48(01): 43-50.

[26] 霍单雨, 彭良福, 杨高. 基于FPGA的视频图像实时缩放系统的实现[J]. 电脑与信息技术, 2023,31(01): 25-28.

[27] 姜元政, 于子钧. MATLAB 在非线性曲线拟合中的应用[J].数学学习与研究,2

017(01): 154.

[28] 付香雪. 芯片装配机器人轨迹规划与字符识别方法研究[D]. 长春工业大学, 2019.

[29] 董俊瑞. 图像缩放和格式变换的算法及实现[D]. 复旦大学, 2013.

[30] 艾鑫. 基于数学形态学的边缘检测算法及其在图像缩放中的应用[D]. 浙江大学, 2011.

[31] 叶森. 边缘自适应图像缩放算法研究及VLSI实现[D]. 浙江大学, 2012.

[32] 罗晶. 基于轨迹智能优化的柔性机械臂振动抑制与自抗扰控制[D]. 湖南科技大学, 2023.

[33] 吴星辰. 基于拓扑优化构型的规整化几何结构重构方法研究[D]. 河南工业大学, 2023.

[34] 左云瑞, 陈东方, 王晓峰. 基于特征融合和混合注意力的超分辨率重建[J]. 计算机工程与设计, 2023, 44(11): 3387-3394.

[35] 柏化春. 基于FPGA的视频格式转换系统的设计与实现[D]. 湖南工业大学, 2013.

[36] 姚刚. 基于CNN的单目标摔倒检测算法研究及FPGA实现[D]. 哈尔滨理工大学, 2022.

[37] 钟宝江, 陆志芳, 季家欢. 图像插值技术综述[J]. 数据采集与处理, 2016, 31(06): 1083-1096.

[38] 张文翔, 董宏林.基于OTSU的图像插值算法[J]. 计算机与数字工程, 2022, 50(01): 186-189.

[39] 杨辉, 边娇, 陈文艺. 基于双四点二次插值的预畸变系数压缩与解压算法[J]. 信息技术与信息化, 2022, 47(03): 117-121.

[40] 侯腾. 图像超分辨率算法研究进展[J]. 现代计算机(专业版), 2018, 35(12): 55-59.

[41] 赵海峰,周永飞,黄子强.图像放大算法比较研究[J]. 现代电子技术, 2010, 33(24):33-36, 39.

[42] 于亚龙, 穆远彪. 插值算法的研究[J]. 现代计算机(专业版), 2014, 31(05): 32-35.

[43] Keys R. Cubic convolution interpolation for digital image processing[J]. IEEE Transactions on Acoustics, Speech, Signal Processing, 1981, ASSP-29(6): 1153-116

0.

[44] 毕玉萍. 基于改进的生成式对抗网络的图像超分辨率重建算法研究[D]. 青岛大学, 2021.

[45] 袁艳春, 刘云鹏, 高宏伟. 基于边缘结构相似性的图像质量评价方法[J]. 计算机应用研究, 2015, 32(09): 2870-2873.

[46] 张英静, 李素梅, 卫津津. 等. 立体图像质量的主观评价方案[J]. 光子学报, 2012, 41(05): 602-607.

[47] 王晓岩, 李野, 岳丹. 一种低光照度下的彩色数码设备色彩还原能力评价方法[J]. 液晶与显示, 2020, 35(05): 477-485.

[48] Wang Zhou, Li Qiang. Information content weighting for perceptual image quality assessment. [J]. IEEE Trans Image Process, 2011, 20(5): 1185-98.

[49] 王蓉, 李志, 李丽华. 视频图像质量评价综述[J]. 中国人民公安大学学报(自然科学版), 2012, 18(01): 59-62.

[50] 靳鑫, 蒋刚毅, 陈芬. 等. 基于结构相似度的自适应图像质量评价[J]. 光电子.激光, 2014, 25(02): 378-385.

[51] 苏衡, 周杰, 张志浩 .超分辨率图像重建方法综述[J]. 自动化学报, 2013,39(08): 1202-1213.

[52] 王少斌, 苏淑靖, 袁财源. 基于FPGA的高清视频采集系统设计[J]. 电子技术应用, 2019, 45(07): 63-66, 71.

[53] 孙冬雪, 王竹刚. 基于DDR3 SDRAM的大容量异步FIFO缓存系统的设计与实现[J]. 电子设计工程, 2018, 26(09): 139-142, 146.

[54] 严传高, 张乘浩, 刘马良, 等. HDMI 高速显示数据接口技术[J]. 微纳电子与智能制造, 2020, 2(02): 105-111.

[55] 陈一波, 杨玉华, 王红亮等. 基于DDR3-SDRAM的图像采集与显示系统[J]. 电子器件, 2017, 40(03): 702-707.

[56] 李新, 梁春明. HDMI接收端的数据同步模块设计[J]. 电视技术, 2016, 40(11): 30-34.

[57] 万权. 基于FPGA的视频分辨率自适应系统设计[D]. 西南科技大学, 2020.

[58] 王轶楷. 实时视频图像缩放系统的FPGA硬件实现[D]. 中北大学, 2023.

[59] 田永怡. 基于FPGA的视频流旋转与缩放系统研究与设计[D]. 长安大学, 2023.

附录 图

图1 原图古楼

图2 原图山丘

图3 原图山水画

图4 原图花田

图5 原图动物

图6 最邻近缩放算法处理后图像

图7 最邻近缩放算法处理后图像

图8 最邻近缩放算法处理后图像

图9 最邻近缩放算法处理后图像

图10 最邻近缩放算法处理后图像

图11 双线性缩放算法处理后图像

图12 双线性缩放算法处理后图像

图13 双线性缩放算法处理后图像

图14 双线性缩放算法处理后图像

图15 双线性缩放算法处理后图像

图16 双三次缩放算法处理后图像

图17 双三次缩放算法处理后图像

图18 双三次缩放算法处理后图像

图19 双三次缩放算法处理后图像

图20 双三次缩放算法处理后图像

攻读硕士期间取得的学术成果

[1] 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

[2] 完海, 张肖强, 杨帆, 等. 基于公共项共享的改进双三次插值算法电路研究[J/OL]. 重庆工商大学学报(自然科学版), 1-9

致　谢

时光飞逝，转眼间三年的研究生学习时间即将结束，回顾这段时间的学习使我受益匪浅。经历半年来的认真思考和细心研究，毕业论文终于完稿，在此过程中得到了老师和同学的关怀和帮助，在此要向他们表示衷心的感谢。

首先，感谢我的导师张肖强教授，他为人和蔼，谦虚谨慎，平易近人，在整个论文的写作过程中，为我倾注了极大精力和心血。在我无思路头绪时，导师总会不厌其烦地帮我梳理思路，耐心解答,并给予我精神上的鼓励和支持；当论文初稿完成后，导师又抽出宝贵的时间认真的研究，提出有独特见地、标新立异的修改意见。虽然我即将毕业，但是恩师的教诲我将永远铭记于心，您永远都是指引我前进方向的灯塔，是我最坚强的后盾，是以感激涕零，常怀滴水涌泉之心。

我还要感谢我的同门师兄弟杨帆，张星，章咸斌，徐明宇等人，如果没有你们在学习和生活上的帮助与支持，我是不会顺利的完成近三年的学业。正是你们的关怀与理解，才能让我投入到学习当中，并顺利完成学业。

在此，我要特别感谢我的女朋友储晓娜，你在我追求学业的道路上给予了我无尽的支持和关心。在你身上，我感受到了无尽的爱与支持。在我遇到困难和压力的时候，你总能给我带来宽慰和力量。有了你的陪伴，我的人生变得更加美好。每一次和你在一起，都让我感到幸福无比。你的爱和鼓励是我能坚持到最后并完成这份论文的重要动力。谢谢你，给我带来了无尽的快乐。

最后，我要深深的感谢我的父母，感谢这么多年来你们无怨无悔的付出，永不停歇的照顾。在外求学不能时常陪在你们身边是我莫大的遗憾，亲情永远是我最温柔的港湾和最重要的支撑。谁言寸草心，报得三春晖，养育之恩，无以回报，我最大的心愿就是你们永远平安健康。

恩情深而笔墨短，友谊远而言语拙，再一次对三年间所有给予我帮助的良师益友们表示衷心的感谢。