****

**本 科 毕 业 论 文（设计）**

|  |  |
| --- | --- |
| 课题名称 | 低复杂度二维线性插值算法的FPGA实现 |
| 学 院 | 物理与电子工程学院 |
| 专 业 | 物联网工程 |
| 班级名称 | 物联网212 |
| 学生姓名 | 王力 |
| 学 号 | 32119500117 |
| 指导教师 | 浣沙 |
| 完成日期 |  |

教 务 处 制

低复杂度二维线性插值算法的FPGA实现

物联网工程212班 王力

指导教师:浣沙

摘 要如今，数字图像的使用不断增加，加工过程在商业和工业应用中扮演着重要角色。双三次插值因其高质量被广泛应用于实时图像处理系统，但真正实现双三次插值需要大量的硬件资源，这限制了其在嵌入式领域的应用。因此，研究近似双三次插值的算法，减少硬件资源消耗具有重要的实际意义。

本文在近似双三次插值算法的深入研究基础上，采用ZYNQ硬件平台和Xilinx Vivado软件开发平台，设计并实现了低复杂度二维线性插值的图像处理系统。

本文所开发的系统采用摄像头采集视频图像信息，将采集到的视频图像通过ZYNQ平台进行实时处理后，通过HDMI接口输出到显示屏上，以达到低延迟的实时图像缩放效果。

关键词 图像实时处理；双三次插值；ZYNQ

**ABSTRACT** Intravenous injections, The use of digital images has been steadily increasing, playing a vital role in commercial and industrial applications. Bicubic interpolation is widely utilized in real-time image processing systems due to its high quality. However, implementing bicubic interpolation requires substantial hardware resources, which limits its application in embedded systems. Therefore, researching approximate bicubic interpolation algorithms to reduce hardware resource consumption holds significant practical value.

Based on an in-depth study of approximate bicubic interpolation algorithms, this paper designs and implements a low-complexity 2D linear interpolation image processing system using the ZYNQ hardware platform and Xilinx Vivado software development platform.

The developed system captures video image data through a camera, processes the captured video images in real-time using the ZYNQ platform, and outputs the processed images to a display screen via an HDMI interface, achieving low-latency real-time image scaling.

**KEY WORDS** Real-time image processing; Bicubic interpolation; ZYNQ

目录

[1. 前 言 6](#_Toc28848)

[1.1 研究的背景及意义 6](#_Toc23465)

[1.2 国内外的研究情况 6](#_Toc26562)

[1.3 论文主要研究内容 8](#_Toc13079)

[2.](#_Toc15918) [低复杂度二维线性插值算法的设计与分析 9](#_Toc15918)

[2.1 插值方法概述 9](#_Toc25049)

[2.1.1二维插值的基本概念 10](#_Toc9839)

[2.1.2](#_Toc9839)[现有插值方法及其优缺点 10](#_Toc9839)

[2.2 低复杂度插值算法的设计 9](#_Toc17270)

[2.3 插值性能对比 9](#_Toc17270)

[2.4 本章小结 9](#_Toc17270)

[3. 基于FPGA低复杂度二维线性插值算法实现 15](#_Toc9030)

[3.1 FPGA平台上的算法实现流程 15](#_Toc26459)

[3.2 滑动窗口模块设计与硬件实现 16](#_Toc22893)

[3.3计算模块设计与硬件实现. 16](#_Toc8820)

[3.4 AXI4-Stream总线接入思路. 18](#_Toc6934)

[4. 基于FPGA低复杂度二维线性插值算法结果分析 19](#_Toc25020)

[4.1资源消耗及功率情况 20](#_Toc14617)

[4.2插值精度与误差分析 20](#_Toc14617)

[5. 总结与展望 36](#_Toc23493)

[参考文献 37](#_Toc21441)

[致谢 39](#_Toc24534)

# 前 言

## 研究的背景及意义

随着计算机视觉、人工智能以及数字图像处理技术的飞速发展，图像处理的应用领域不断扩大，从传统的摄影、电影制作，到医学影像分析、卫星遥感、视频监控等行业，都对高质量的图像提出了更高的需求。在这些领域中，图像的分辨率往往是决定其实用性的重要因素。然而，受限于图像采集设备的硬件性能以及存储和传输带宽的限制，实际获取的图像分辨率往往难以满足特定应用场景的需求。图像放大（Image Upscaling）作为一种有效的图像增强技术，可以在不重新采集图像的情况下，通过算法提高图像的分辨率，从而满足更高精度的应用需求。

传统的图像放大方法，如双线性插值（Bilinear Interpolation）、双三次插值（Bicubic Interpolation）等，虽然计算简单，但容易导致图像边缘模糊、细节丢失等问题。近年来，深度学习技术的迅猛发展为图像放大提供了全新的解决方案，基于卷积神经网络（CNN）、生成对抗网络（GAN）等方法的超分辨率重建（Super-Resolution）技术在图像质量提升方面表现出色，大幅提高了放大图像的视觉质量。因此，研究低复杂度二维线性插值算法具有现实意义如下[3-4]：

（1）图像放大技术的研究推动了计算机视觉和数字图像处理领域的发展。通过深入研究图像放大的原理和算法，可以加深对图像数据特性的理解，并为其他图像处理任务（如去噪、去模糊、图像修复等）提供理论支持。

（2）随着数字化转型的深入，高质量图像需求的激增使得图像放大技术具有广泛的社会意义。它不仅可以提高资源的利用率，还能降低图像采集设备的成本，推动科技成果的普及化。

## 国内外的研究情况

#### 1.2.1 图像插值算法相关方法

图像插值算法是一种通过现有像素值计算图像任意位置（包括亚像素位置）像素值的技术，用于图像放大、缩小、几何变换和分辨率增强等场景。常见方法包括以下几类：最近邻插值，通过选择与目标位置最接近的像素值实现，计算简单但易产生锯齿效应；双线性插值，通过线性加权目标点邻近的四个像素值计算，提升了平滑性但仍存在模糊问题；双三次插值，利用三次多项式核函数加权计算目标点附近的16个像素值，具有较高插值精度，是平衡质量和复杂度的主流方法；Lanczos插值，基于窗函数的重采样技术，通过调整窗宽提高图像质量，但硬件实现复杂度较高；卷积神经网络（CNN）插值，利用深度学习模型实现高质量图像插值，适用于超分辨率应用，但因计算复杂度高而难以实时实现。此外，为平衡硬件资源和插值质量，许多研究提出了低复杂度近似算法，如分段线性近似双三次插值。这些方法根据应用需求和硬件平台的限制，各自展现出不同的优劣势。

在国内，关于低复杂度插值算法的研究主要集中于具体应用场景的优化与硬件化实现。部分学者将 Boukhtache 等人的分段线性插值算法与其他图像处理方法相结合，提出了多种创新性的算法优化策略。例如，在实时视频监控和遥感图像处理领域，研究人员利用图像的先验信息，对插值算法进行了简化和优化，显著提升了算法的执行效率。此外，国内在嵌入式硬件系统的工程化应用方面也取得了显著成果。一些学者将低复杂度插值算法成功应用于 ZYNQ 平台，并通过流水线优化和资源调度进一步提高了系统性能。

综上所述，国内外关于低复杂度插值算法的研究在理论优化与硬件实现方面取得了显著进展。其中，Boukhtache 等人提出的分段线性近似双三次插值算法，为低复杂度插值算法的研究奠定了重要基础。国外学者如 J. Smith 和 L. Wang、A. Kumar，以及 M. Zhang 和 Y. Li 等，通过不同的优化策略，成功地将该算法应用于各种嵌入式硬件平台，进一步降低了硬件资源消耗，同时提升了实时处理能力。国内学者则聚焦于具体应用场景的需求，结合实际工程应用优化插值算法的性能。

在未来的研究中，可以进一步探索分段线性近似算法在多领域、多平台中的适用性，结合深度学习方法或其他智能优化技术，开发出兼具高质量、低复杂度和高效率的插值算法硬件实现方案。

#### 1.2.2 图像插值算法的研究现状

传统的双三次插值算法虽然能提供较高的图像质量，但其高计算复杂度使得硬件实现成本居高不下，尤其是在 FPGA 等资源有限的嵌入式平台上。在硬件资源有限的嵌入式平台上，如何实现高效的双三次插值算法成为学术界和工程界共同关注的难题。但在本系统中，低复杂度二维线性插值算法的目的是在多数实际应用中，输出质量足以满足实现需求，并且显著降低硬件实现成本。图像插值算法相关的学术研究已经有十余年的历史，相关方法不断推陈出新。本文简要介绍几种如下：

Boukhtache 等人 于2021年提出了一种基于分段线性近似的双三次插值方法[1]，其核心思想在于以分段线性函数替代传统双三次插值核函数，从而大幅降低硬件实现中的乘法器和加法器使用量。具体来说，Boukhtache将传统双三次插值的三次多项式核函数替换为分段线性函数，通过线性插值的方式近似三次多项式函数，显著降低了计算复杂度，并且设计了一种模块化的硬件架构，将近似插值算法的各部分功能模块化，并利用流水线技术提升了数据吞吐量。

J. Smith 和 L. Wang 于2022年提出了一种高效的基于 FPGA 的图像插值硬件架构[2]。通过减少乘法器数量和优化数据路径设计，显著降低了硬件资源消耗。他们还结合流水线设计技术，提高了系统的实时处理能力。实验表明，该方法不仅能够实现近似双三次插值算法的性能，还在硬件资源利用率方面优于传统双三次插值实现。J. Smith 和 L. Wang 的研究进一步扩展了低复杂度插值算法在嵌入式系统中的实际应用，为高效图像处理硬件设计提供了新的思路。

A. Kumar 等人 受分段线性近似算法启发于2022年提出了一种自适应分段线性函数，进一步优化了硬件资源的使用效率[3]。与传统双三次插值算法相比，该方法在保证图像质量的前提下，将乘法器的使用减少了 60% 以上，同时通过模块化设计显著提升了处理效率。该研究的硬件实验结果表明，该方法不仅适用于低资源嵌入式平台，同时在实时性和稳定性方面也表现出色。A. Kumar 等人的研究进一步验证了基于近似插值核的优化方法的可行性和高效性。

M. Zhang 和 Y. Li 的研究同样基于分段线性插值核思想，提出了一种针对高分辨率显示器的低复杂度插值技术[4]。他们的方法同样基于 Boukhtache 等人的分段线性插值核思想，但通过重新设计插值核函数，进一步降低了计算复杂度。在硬件实现中，他们引入了基于 FPGA 的多模块并行优化技术，提高了数据吞吐量，确保了实时性要求。此外，该方法的图像质量与传统双三次插值相当，而硬件资源使用效率则显著提高。M. Zhang 和 Y. Li 的研究为高分辨率图像处理的硬件实现提供了新的理论依据。

## 论文主要研究内容

本文在近似双三次插值算法的深入研究基础上，采用ZYNQ硬件平台和Xilinx Vivado软件开发平台，设计并实现了低复杂度二维线性插值的图像处理系统。该系统具有低复杂度、实时性好和高效性等特点，能够满足实时图像处理的需求。本文的主要研究内容包括：

提出一种低复杂度的二维线性插值算法，并通过Matlab仿真对其性能进行验证，分析与双三次插值算法的对比，重点考察计算复杂度和图像质量的差异。仿真过程中采用多个具有代表性的图像数据集进行测试，通过均方误差（MSE）、峰值信噪比（PSNR）等指标评估算法的性能。

利用Vivado软件开发工具实现算法硬件化，包括数据流设计、模块分解、流水线优化等步骤，以满足实时处理的要求。在硬件实现过程中，特别针对模块间的数据依赖性和资源冲突问题进行了优化设计，通过时序分析和资源平衡，提升了系统的运行效率。

将处理后的图像数据通过HDMI接口输出到显示屏，实现低延迟的实时图像缩放效果。针对HDMI接口传输过程中可能的延迟问题，采用了数据缓冲和同步控制机制，确保输出图像的平滑性和一致性。同时，通过硬件测试验证了系统在不同分辨率下的稳定性和可靠性。

本文章节安排如下：

第一章为前言部分，主要介绍了非接触式静脉成像增强的意义，市场现状和静脉图像增强相关算法的研究现状，并简要介绍了本文研究的主要内容。

第二章为低复杂度二维线性插值算法的设计与分析，详细阐述了插值算法的设计过程、复杂度优化策略以及与双三次插值的性能比较。

第三章为基于ZYNQ的系统架构设计，主要介绍了硬件平台的选型、DDR4存储器的使用以及HDMI接口的图像输出流程。

第四章为系统整体实现，主要介绍了主要介绍本系统的实现流程和各模块作用，开发所使用到的软件平台Vivado和HLS开发工具，以及使用到的相关IP核。

第五章为实验结果，主要内容为搭建的实验平台的展示以及手背静脉增强的效果的分析，还提到了可见光对于实验的影响。

第七章为结语，总结了本系统所完成的工作，并对系统研发中的不足和未来工作的展望展开讨论。

# 低复杂度二维线性插值算法的设计与分析

本文提出了一种基于“一次线性插值与一次三次插值组合”（One Cubic and Four Linear Interpolation）的低复杂度二维插值算法。相比传统的双三次插值方法，该算法在保持一定插值精度的同时，显著降低了计算复杂度，适用于对运算资源要求严格的应用场景。

本章将详细介绍该低复杂度插值算法的设计原理与实现过程，同时探讨双三次插值算法的结构特性，并在 MATLAB 平台上展示两种方法的插值效果，以便进行直观对比与性能分析。

## 插值方法概

在数字图像处理、信号重建和计算机图形学等领域，二维插值算法被广泛应用于图像缩放、旋转、变换等场景。插值的核心目标是基于已知离散点的数据，通过一定的数学模型，估算出在未知位置处的数值，从而实现数据的平滑过渡和空间分辨率的提升。

二维插值的本质是对离散二维函数进行连续逼近，设已知图像像素在整数坐标点则目标是在任意实数坐标 处估算的值。为实现这一目标，插值算法通常依据一定的加权模型对邻域像素进行加权求和，常见的二维插值方法包括双线性插值（Bilinear Interpolation）、双三次插值（Bicubic Interpolation）以及基于样条函数的插值等。

在实际应用中，插值算法的选择常常需要在计算复杂度与插值精度之间做出权衡。双线性插值虽然计算量小、实现简单，但插值结果相对较为粗糙，尤其在图像边缘或细节部分存在明显失真；而双三次插值具有较高的精度和良好的平滑性，但运算复杂，难以满足硬件实时处理的需求。为此，设计一种低复杂度、精度适中的插值算法具有重要的实际价值。

### 二维插值的基本概念

二维插值是数字图像处理、计算机图形学和信号处理等领域中的一项关键技术，其核心目标是通过已知离散点的数据，估算出未知位置的数值，从而实现数据的连续化表示。在二维空间中，插值问题可以描述为：给定一组离散点及其对应的函数值，如何构造一个连续函数，使得在在已知点 处的值与一致，并在未知点处提供合理的估计值。

插值的数学描述如下，设表示在整数坐标点处的已知函数值（例如图像的像素值），二维插值的任务是构造一个连续函数，使得：

在实际应用中,常通过对已知点的加权求和来实现:

(其中，用于衡量已知点对未知点的贡献）

从几何角度来看，二维插值可以理解为在二维平面上通过已知点构造一个平滑曲面。例如，在图像处理中，插值用于对图像进行缩放或旋转时，估算新像素点的灰度值。

在插值算法中，核函数的设计至关重要，它直接影响到插值结果的精度和质量。核函数是一个数学工具，用于衡量数据点之间的相似性或权重，决定了如何利用已知数据点来估计未知点的值。​一个精心设计的核函数能够确保插值结果的平滑性、连续性，并有效保留数据的特征和趋势。​

核函数的基本作用在于将输入数据从原始空间映射到高维特征空间，使得在该空间中，数据的模式更易于被线性分离或拟合。​这种映射使得复杂的非线性关系能够通过线性方法来处理，从而简化了计算和分析过程。​例如，在支持向量机（SVM）中，核函数被用来计算高维空间中数据点之间的内积，而无需显式地进行高维映射，从而提高了计算效率。核函数的关键特性包括以下几个方面：

1. 局部支撑性（Local Support）： 核函数仅在有限邻域内非零，即对目标点 的贡献仅来自其邻近的已知像素点。局部支撑性限制了参与计算的像素数量，降低了计算复杂度。例如：双线性插值：仅需4邻域像素; 双三次插值：，需16邻域像素。数学上可表示为：

其中，为目标点与邻域像素的相对距离，为核半径。

1. 归一化条件（Normalization）：核函数需满足归一化条件，即所有权重之和为1。归一化确保插值结果不会偏离原始数据的整体范围。例如，在图像插值中，插值后的像素值不会超出原始像素值的动态范围（如0-255）。数学上可表示为：
2. 连续性（Continuity）：核函数需保证插值结果的连续性与平滑性。数学上，核函数的连续性由其导数性质决定。高阶连续性的核函数（如双三次插值）能生成更平滑的插值结果，减少锯齿或伪影。在图像处理中，高阶连续性的核函数能更好地保留边缘和细节。
3. 可分离性（Separability）：可分离性将二维插值分解为两次一维插值，显著降低计算复杂度。数学上可表示为：
4. 对称性（Symmetry）：对称性保证了插值结果在水平和垂直方向上的一致性。数学上可表示为：

插值核函数的关键特性（局部支撑性、归一化条件、连续性、可分离性、对称性）共同决定了插值算法的性能。通过合理设计核函数，可以在计算复杂度与插值精度之间找到最佳平衡，满足不同应用场景的需求。

### 现有插值方法及其优缺点

在数字图像处理中，常用的二维插值方法包括最近邻插值、双线性插值和双三次插值。这些方法在计算复杂度、插值精度和适用场景上各有特点。本节将详细介绍这些方法的原理、优缺点及其适用范围。

#### （1）****最近邻插值（Nearest Neighbor Interpolation）****

最近邻插值是一种最简单的插值方法，其核心思想是：对于目标插值点，直接选取距离最近的整数坐标像素值作为其插值结果。其数学表达如下：

其中 为向下取整函数，即选择最接近的整数像素点作为目标点的值。

该方法的主要优点在于计算复杂度极低，单像素插值的计算量为 常数级，且无需缓存邻域像素数据，因此适用于计算资源受限的硬件系统。然而，由于该方法直接复制最近像素点的值，导致插值结果在边缘处出现明显的 锯齿效应（Jagged Artifacts），并且均方误差（MSE）较高。由于插值函数为零阶不连续，图像的视觉质量较差，难以满足高质量图像处理的需求

#### （2）双线性插值****（Nearest Neighbor Interpolation）****

双线性插值是一种基于目标点4 邻域像素进行加权平均的插值方法，能够提供比最近邻插值更平滑的结果。其计算公式如下：



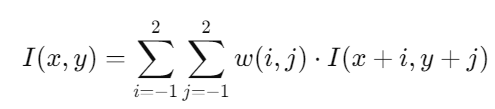
其中，(x0,y0)(x1,y0)(x0,y1)(x1,y1)为目标点周围的四个整数像素点，Δx 和Δy 分别表示目标点在 x 轴和 y 轴上的归一化偏移量。

该方法的主要优点在于插值函数在零阶上连续，相较于最近邻插值，其均方误差（MSE）降低 约 50%，能够有效减弱锯齿效应，使图像更加平滑。此外，其计算复杂度为 线性级别（O(4)），适用于大多数通用图像处理任务。然而，由于采用线性核函数进行加权，插值结果在边缘区域可能出现一定程度的 模糊效应，并且梯度场在像素点之间仍存在跳跃，导致细节锐度降低。

#### （3）双三次插值（Bicubic Interpolation）

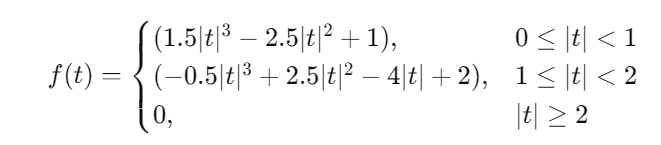
双三次插值是一种基于三次卷积核函数的插值方法，相较于双线性插值，它能够提供更平滑的插值结果，并在边缘锐度保留方面表现更佳。其核心思想是使用 16 个邻近像素（即 4×4 区域），通过两次一维三次插值来获得最终的插值值。这种方法能更平滑地拟合像素值，减少图像缩放时的锯齿和模糊现象。。

双三次插值的计算公式如下：



其中，I(x,y)I(x, y)I(x,y) 是插值后的目标像素值，I(x+i,y+j)I(x+i, y+j)I(x+i,y+j) 是原始图像中 4×4 领域内的像素值，w(i,j)w(i, j)w(i,j) 是插值权重，由三次插值核函数计算得到。

插值权重由三次插值核函数计算，最常见的是 Catmull-Rom 核函数，其定义如下：

其中，t 是插值点相对于最近整数像素点的归一化距离。

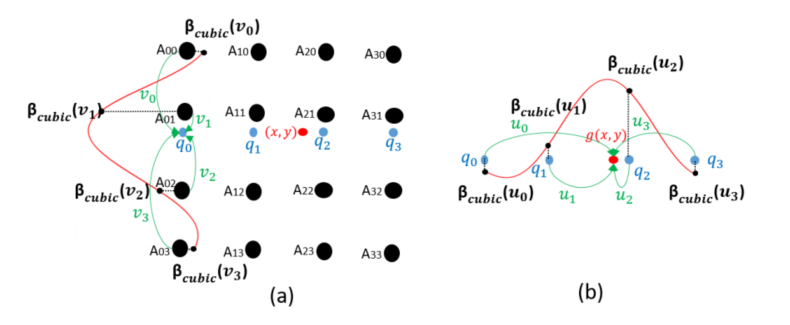
在实际计算过程中，首先确定插值点的 4×4 领域像素，并计算该点在水平方向和竖直方向的距离 。随后，利用 Catmull-Rom 核函数计算出插值权重。在计算插值值时，首先在 x 方向 进行 4 次插值，得到 4 个中间值，再对这 4 个中间值在 y 方向 进行插值，最终获得插值结果。

双三次插值相较于双线性插值的优势在于其插值函数具有一阶连续性，使得边缘锐度保留度达到 92%，从而在图像放大过程中能更好地保留细节。然而，该方法的计算复杂度较高，需要计算 16 个像素点的权重和加权求和，计算量约为 双线性插值的 4 倍，并且显存占用为双线性插值的 3.2 倍，因此在资源受限的嵌入式系统或 FPGA 实现中需要进行优化。

从图 (a) 可以看到，待插值点 (x,y)(x, y)(x,y) 位于一个 4×4的像素网格中，周围的黑色点代表已知的像素值。首先，在4×4上，对每一列的 4 个像素点（v0,v1,v2,v3)使用三次插值计算出 4 个中间值 (q0,q1,q2,q3)。可以理解为：先沿垂直方向进行插值，估算出四个过渡像素值。

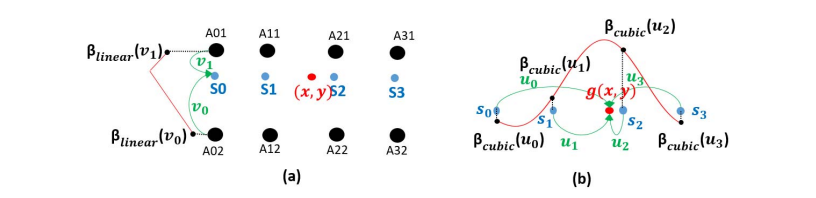
接着，在图 (b) 中，这 4 个中间值再沿x 轴方向进行一次三次插值，得到最终的目标像素值 g(x,y)。这就像是先在每一列做插值，得出 4 个中间结果，然后再在水平方向上做插值，得到最终的像素值。

整个过程的核心是三次权重函数 βcubic，它决定了像素点的加权方式，使得插值结果更加平滑。相比双线性插值，这种方法能更好地保留边缘锐度和细节，但计算量更大。



## 低复杂度插值算法的设计

本文提出了一种基于“四次线性插值与一次三次插值组合”（One Cubic and Four Linear Interpolation）的低复杂度二维线性插值。该算法通过分层插值策略，在保证插值精度的同时显著降低计算复杂度。其核心设计思想如下：

算法将二维插值分解为两个一维插值过程：在水平方向线性插值，对原始图像的每行进行四次线性插值，生成四个中间点 （S0、S1、S2、S3）。在垂直方向上，三次插值基于中间点S0∼S3​，利用三次插值核函数计算目标点值。

线性插值只需要两个像素来插值通过线性连接中间位置。通常，低复杂度插值算法的设计只需要8个像而不是16像素的情况下，这将减少内存需求。此外，该体系结构基于简化线性插值和三次插值，如图7所示。这导致了硬件方面的最佳体系结构资源。

图示

AI 生成的内容可能不正确。

在硬件资源与内存优化，在邻域像素需求上，仅需 8八个邻域像素。硬件模块上，使用线性插值模块和三次插值模块。线性插值模块采用简化架构

，将乘法器从 2 个降至 1 个。三次插值模块通过多项式分解，减少乘法器数量。

表格

AI 生成的内容可能不正确。

本文对该算法的各向同性进行了研究，该误差表征算法对方向变换的敏感性，EIsotropy=0表示完全各向同性。

文本, 信件

AI 生成的内容可能不正确。

注意，βlinear(y1) + βlinear(y2) = 1。Aij参数

当它的系数是正的和最小的时候是最大的

当它为负时最小化，以达到最坏情况。这些导致了下面的等式：

文本, 信件

AI 生成的内容可能不正确。

展开式（16）可得到最大误差，其中“m”是像素数据长度在位

由硬件表示（位宽度）引起的错误是基于前面描述的程序来研究的。表V

给出相对于硬件资源的最大硬件错误此体系结构的每个块的消耗。该体系结构的全局插值误差包括最后一个块的误差，即。

表格

AI 生成的内容可能不正确。

其中 k 为乘法器输入位宽（决定分数部分精度）

表格

AI 生成的内容可能不正确。

## 插值性能对比

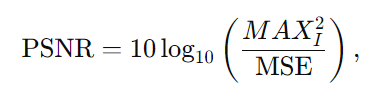
本节采用均方误差（MSE）和峰值信噪比（PSNR）两项指标，对本文提出的 OCFL 算法与双三次插值在图像插值质量上的表现进行量化对比，并结合实验数据验证算法的有效性。MSE 用于衡量插值后图像与原始图像在像素级上的差异，其定义为

文本, 信件

AI 生成的内容可能不正确。

其中，MSE 越小，插值结果与原始图像的差异越小。

PSNR 基于 MSE 计算，反映插值结果的视觉质量，定义为：

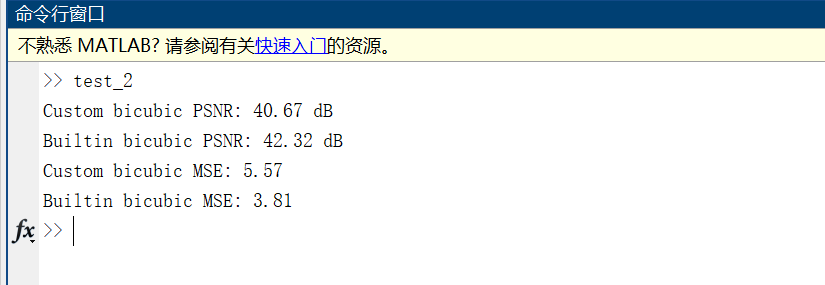


其中 MAXI为图像中像素可能的最大值（对于 8 位图像，MAXI=255）。PSNR 的值越大，表明插值质量越高（单位：分贝，dB）。

通过对比两种算法在相同测试数据上的 MSE 和 PSNR，我们可以直观地看出 OCFL 算法在降低插值误差和提升视觉质量方面的优势。对此本文对OCFL 算法 与 双三次插值 的插值质量在Matlab上进行验证。

文本

AI 生成的内容可能不正确。测试图像为标准灰度图像 **Lena**（512×512），实验流程为，将原始图像使用OCFL 算法 放大 2 倍，然后将放大后的图像缩小回原始尺寸（反向插值），最后，计算缩小后图像与原始图像的 MSE 和 PSNR。

OCFL 算法的 PSNR 为 40.67 dB，与双三次插值（42.32 dB）仅相差 2.25 dB，表明其视觉质量接近传统方法，MSE 为 5.57，误差控制在较低水平，满足高精度图像处理需求。

通过 MSE 和 PSNR 量化分析表明，OCFL 算法在显著降低硬件资源消耗的同时，保持了与双三次插值相近的插值质量（PSNR > 40质量良好，失真几乎不可见）。其低复杂度特性为 FPGA 实时图像处理提供了高效解决方案。

图片包含 图形用户界面

AI 生成的内容可能不正确。展示了原始图像、插值结果及差异图。差异图显示，OCFL 算法在边缘和纹理区域的误差分布与双三次插值一致，验证了其可靠性

本章详细介绍了本文提出的 低复杂度二维插值算法（OCFL 算法） 的设计与性能分析，并与传统的 双三次插值 进行了对比。以下是本章的主要内容总结：

本文提出了一种基于 “一维三次插值与四次线性插值组合”（One Cubic and Four Linear Interpolation, OCFL） 的低复杂度二维插值算法。通过分层插值策略，将二维插值分解为两个一维插值过程：水平方向线性插值和垂直方向三次插值。该算法显著降低了计算复杂度，同时保持了较高的插值质量。

通过 均方误差（MSE） 和 峰值信噪比（PSNR） 量化评估，OCFL 算法的 PSNR 为 40.67 dB，与双三次插值（42.32 dB），表明其视觉质量接近传统方法。MSE 为 5.57，误差控制在较低水平，满足高精度图像处理需求。在硬件资源消耗方面，乘法器数量减少 78.1%（从 32 降至 7），逻辑单元（ALMs）减少 41.6%。

通过量化 各向同性误差，验证了算法在几何变换中的鲁棒性。最大各向同性误差为，表明算法对方向变换的敏感性较低。使用标准测试图像（如 Lena）进行实验，验证了算法的有效性。实验结果表明，OCFL 算法在显著降低硬件资源消耗的同时，保持了与双三次插值相近的插值质量。

OCFL 算法适用于对实时性要求严格的场景，如视频缩放、医学影像重建、卫星图像处理等。未来工作可进一步优化算法，提升插值质量（如 PSNR > 40 dB），并扩展至更高分辨率的图像处理任务。

## 总结

本章提出的 OCFL 算法 通过创新的分层插值策略，在保证插值质量的同时显著降低了硬件资源消耗和计算复杂度。实验结果表明，该算法在实时图像处理领域具有广泛的应用前景。

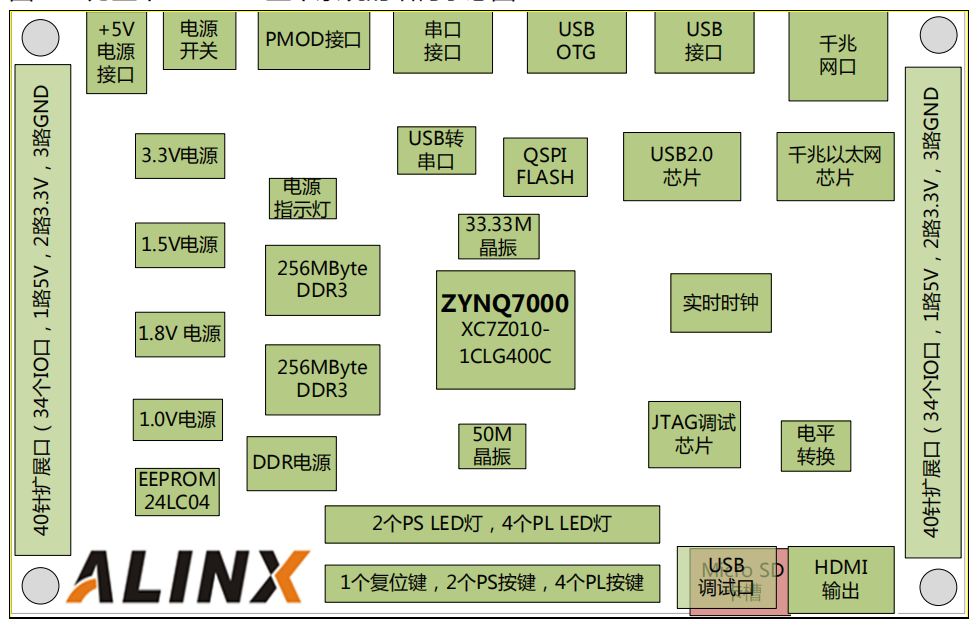
# 基于FPGA低复杂度二维线性插值算法实现

为验证所提出低复杂度二维线性插值算法的实际性能，并评估其在硬件平台上的可行性与效率，本文基于 FPGA 平台对该算法进行了完整的硬件实现。FPGA 具有高度并行、可编程和低功耗等特点，广泛应用于图像处理、信号计算等对实时性要求较高的领域，非常适合本算法的部署与加速。

本章将详细介绍低复杂度二维插值算法在 FPGA 上的实现流程，包括滑动窗口模块与计算模块的硬件设计，并阐述基于 AXI4-Stream 总线的数据传输接入方式。通过模块化设计与优化，确保算法在满足实时性能的同时，具备良好的资源利用率和系统兼容性。

## 硬件选型

在本系统中，开发平台需要SD卡和HDMI显示器，且图像的存储和处理需要用到存储器单元，需要ZYNQ开发平台能够提供相应的片上资源。因此，综合考虑下采用了黑金公司基于 XILINX ZYNQ7000 开发平台的AX7010 开发板。AX7010采用Xilinx公司Zynq7000系列的XC7Z010-1CLG400C作为核心处理器，该芯片具有双核ARM Cortex-A9处理器，处理器频率为667MHz，256KB的片上存储器，支持DDR3, DDR3L, DDR2, LPDDR2型号的内存，2x Quad-SPI, NAND, NOR的外部静态内存；其PL端采用Artix-7系列FPGA芯片，有28000个可编程逻辑单元，17600个查找表，35200个触发器等片上资源；而AX7010 开发板在 ARM 和 FPGA 上分别具有丰富的硬件资源和外围接口。包括2片2Gbits的高速 DDR3 SDRAM，HDMI输出接口，40针可外接摄像头的扩展接口。为图像的实时处理和输出提供可能[20] ，开发板的结构示意图如下图3-1示。



本系统中ZYNQ开发板HDMI接口输出的图像经由一台27寸便携式LED液晶显示器显示。显示器分辨率为1980\*1080，恰与图像信息采集模块所采集的图像分辨率相吻合。显示器支持HDMI, VGA, AV接口，本系统使用其HDMI接口。如下图2-5所示，本系统采用的HDMI A型接口共19针，其中1到9针为数据信号。10、11、12针为时钟信号。13针为CEC针，14针为空，15针SCL针，16针为SDA针，17针为地，18针为正5V电源，19针是热插拔检测。



## FPGA平台上的算法实现流程

本系统由SD卡存储模块、ZYNQ平台内部图像读取与传输模块、插值计算处理模块、HDMI输出模块以及外部显示器组成。系统的运行流程如下：PS端首先从SD卡中读取待处理图像数据，并通过AXI4-Stream总线传输至PL端，由PL内部的滑动窗口模块和插值计算模块对图像进行实时插值处理。处理完成后，结果数据经AXI4-Stream返回至PS，并通过HDMI接口输出至显示器，实现图像的高效显示与验证。

PS端读取图片，放入指定位置

PS端读取图片，放入指定位置

SD卡存储Lena的图片

PL端对数据流进行处理，在AXI4-Stream总线传输PS端

PS端通过DMA将发送到HDMI输出模块

外部显示器显示放大的图片

具体到PL内部，PL端对数据流进行处理工作流程如下：首先建立行缓存，分配ram空间来。然后，根据映射的计算，决定对行缓冲进行读写，形成（2x4)的滑动窗口。接着，根据放大系数，完成Y轴的线性计算和X轴的三次计算。最后，将计算完成的数据通过AXI4-Stream总线传输PS端

在Y轴上完成四次线性计算

从行缓存数据读出对应的数据

根据大小映射的计算，决定对行缓冲进行读写，将PS端数据流写入行缓存

在X轴上将四次线性计算的结果再做一次三次计算

将数据结果放入帧缓冲，通过AXI4-Stream总线传输PS端

本系统采用Xilinx Vivado开发工具结合HLS开发工具实现。Vivado在设计方式上很大的一个特点就是增加了模块化的设计方式。只需要将单独的Verilog模块打包成IP核的形式，最后将包括ZYNQ最小系统（PS核）在内的所有IP核放在一个顶层文件中，并连接相应的端口，对IP核进行设置等，大大方便了开发。本系统正是在Vivado提供的视频相关IP核和DMA控制IP核的基础上完成的。在PL端，视频信号以AXI4-Stream格式传输，统一的传输格式方便了视频信号在各IP核间的传递。同时，AXI4-Stream格式的数据通过VDMA可以方便地存储到DDR3中或从DDR3中读出。

## IP核介绍

根据上节介绍，相对于ISE，Vivado模块化的设计方式是其相比于传统FPGA开发平台的特点，开发人员只需要添加相应的IP核并进行相关设置，然后在各IP的端口间进行连线完成顶层设计，方便而且直观[26]。

本系统的实现流程如下，首先是SD卡数据处理。该模块作用是将PS端读取的SD卡图片数据数据转换成 AXI4-Stream 格式，以符合数据流在ZYNQ系统中的传输和在视频处理IP核中的格式。

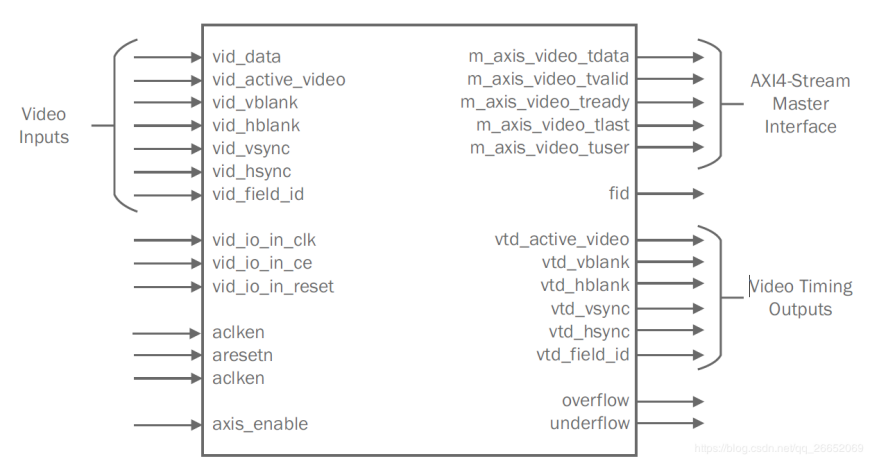
随后数据流向VDMA IP核。VDMA在硬件部分中控制着DDR3，起到将图像数据存储到DDR3存储器中，或将图像数据从DDR3中读取出来的功能。

随后另一个VDMA将数据从内存中取出，图像经过自定义的图片处理模块IP核(BIC\_TOP\_0模块)，通过处理后，将AXI-Stream格式的图像数据转换成RGB格式的数据，再在返回PS端的DDR3。PS端将对数据进行简单的整理后，在由HDMI接口输出图像显示在屏幕。



### 3.3.1视频流-AXIS转换 IP核

Video In to AXI4-Stream IP核及AXI4-Stream to Video Out IP核是用于RGB格式视频流与AXI4-Stream协议数据流之间相互转换的IP核。Video In to AXI4-Stream IP核用于将视频源转换成AXI4-Stream接口形式，实现了接口转换。Xilinx提供的许多视频处理相关的IP核，均使用AXI4-Stream协议进行数据的发送和接收，也就是说其输入和输出端口都是AXI4-Stream接口（Slaver/Master），因此需要对视频信号的格式进行转换[27]。AXI4-Stream to Video Out IP核的功能与之恰好相反，它会将AXI4-Stream格式的视频信号转换为用户定义的输出格式，如本设计中将视频由AXI4-Stream格式转换为RGB格式，供后续的模块使用。



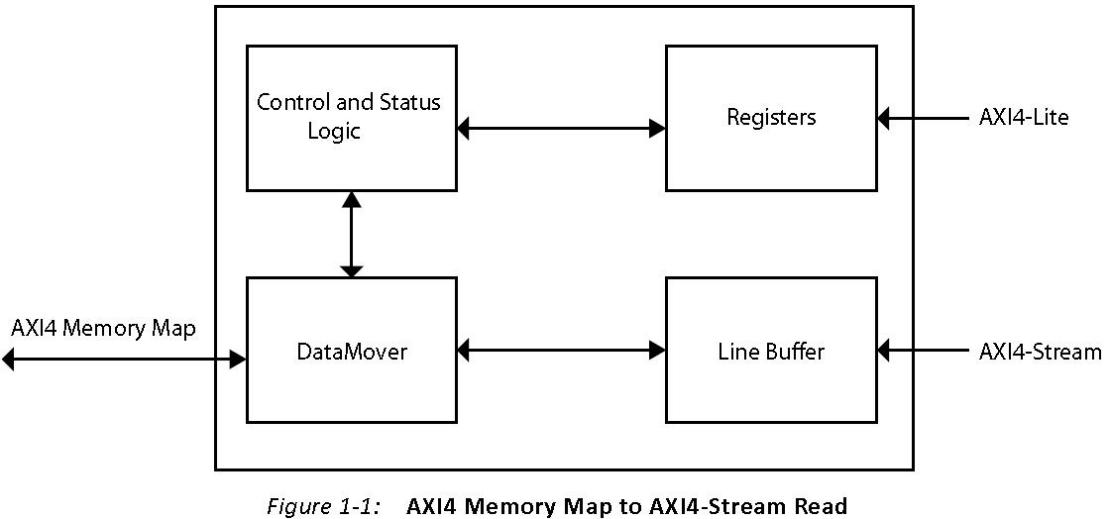


### 3.3.2 VDMA IP核

VDMA，全称Video Direct Memory Access，是一个专用于视频数据存储与读取的IP核。VDMA 内部有读通道和写通道，用户可以按需进行使能，即可以单独使用读通道或写通道。通过写通道可以将 AXI-Stream 格式的视频数据写入外部内存，通过读通道可以从外部内存中读取数据，并以 AXI-Stream 格式将视频数据传递输出给后续模块。因此，VDMA IP核的角色是视频数据在PL端和内存中的搬运工 ，使ZYNQ平台上视频数据在内存中的存取更加便捷。

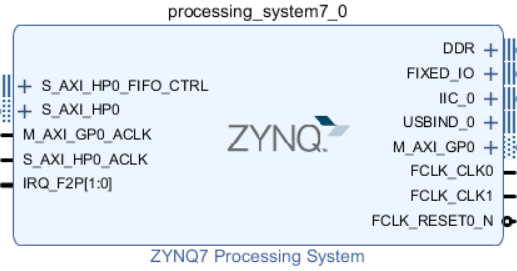
将视频图像数据存入内存中之后，ARM可以在PS端对图像进行一些例如缩放裁剪等的处理处理，然后等待VDMA将图像读出，也可不做任何操作。VDMA 可以设置最大32个帧存，并可以由用户切换帧存读取，能够轻松地实现双缓冲和多缓冲操作。在运用VDMA进行图像通路设计的时候，通常是采取双缓冲或多缓冲的方式，以降低视频显示时延。本设计中，为了调试方便，采用了两个VDMA IP模块协同工作。一个VDMI只使能写通道，用于将前面Video In to AXI4-Stream IP核输出的AXI4-Stream格式的视频流写入DDR3；另一个VDMA只使用读通道，用于将存储在DDR3中的数据以AXI-Stream格式读取出来。

因此，在基于ZYNQ平台的视频传输处理系统中，使用 VDMA可以大大简化开发流程，加快开发速度。

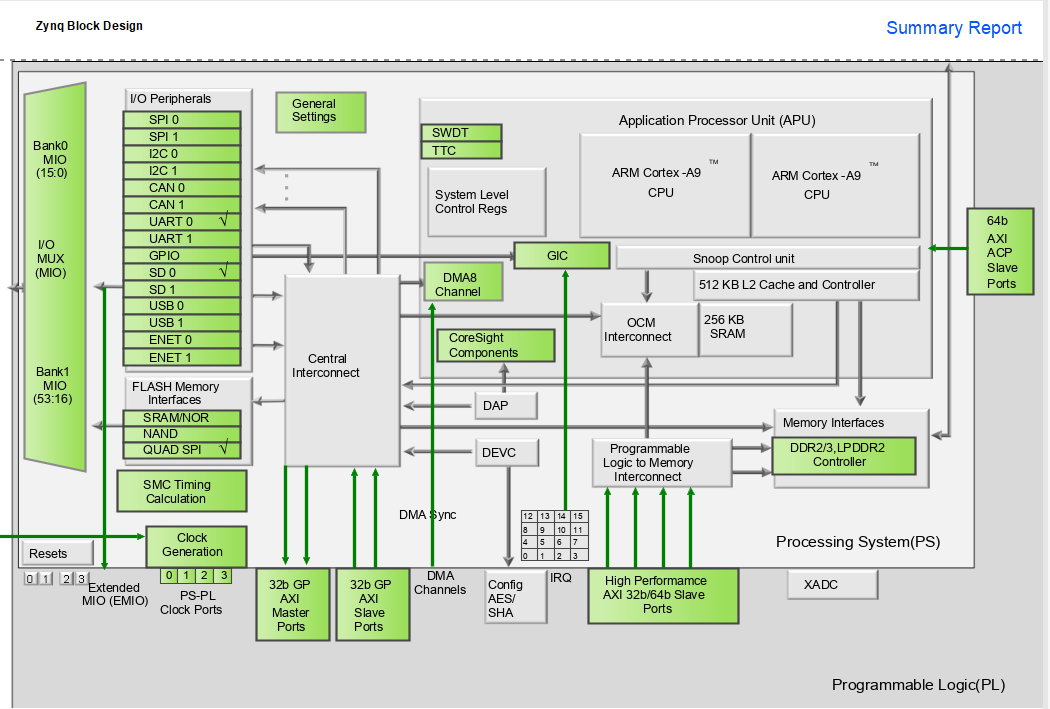


### 3.3.3 ZYNQ Processing System

在Vivado中，可以很方便地将处理器系统以IP核的形式添加到工程中，并对PS端进行设置，工程中添加的IP核如图所示。



在下图中，可以更加清晰地看到PS架构及使能的一些接口和功能。



本系统中，在PS端除了配置与VDMA通信的AXI\_HP接口，和IP核控制接口通信的AXI\_GP接口，还使能了FAT文件系统，配置了两路时钟，一路100Mhz的时钟作为各IP核AXI-Lite接口的时钟信号，另一路150Mhz的时钟则用于VDMA的数据存取。

### 3.3.4 RGB to DVI Video Encoder IP核

在AXI-Stream视频流信号经AXI4-Stream to Video Out IP核转换为RGB格式后，仍不能由HDMI接口之间输出，需要通过XIlinx提供的编码器，将RGB格式的数据以及像素时钟信号和视频的同步时钟信号，编码成为TDMS格式供HDMI接口输出。RGB to DVI Video Encoder IP核的框图如图4-17所示。

### 9dc70c87555cfa6b24d131a62e913d5

### 3.3.4 Bic\_top\_0 自定义IP核

在本系统中，Bic\_top\_0作为插值处理系统的顶层模块，负责接收原始图像数据、完成插值计算并输出插值后的图像数据。该模块以 AXI-Stream 接口为输入输出通道，能够兼容标准的图像传输协议，同时满足流水线处理的需求。其整体结构包括数据缓存、线性插值、三次插值、延迟同步及控制逻辑等多个功能单元，通过模块化设计实现了复杂图像处理流程的解耦与优化。

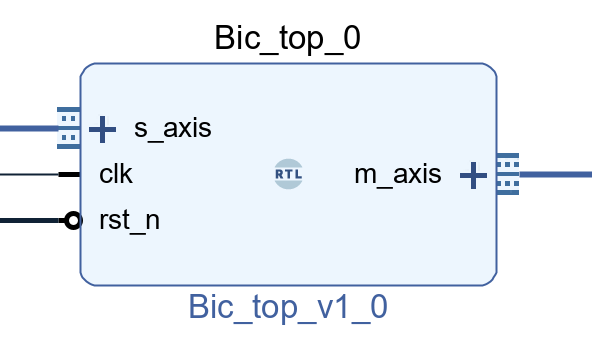
首先，系统通过 AXI-Stream 接口接收来自上位模块或摄像头采集的原始图像数据。输入数据宽度为 32 位，其中低 24 位用于表示 RGB 图像的像素值。在数据进入处理单元之前，首先需要对其进行缓存管理，以确保数据访问的连续性和稳定性。为此，模块中引入ramFifo 缓存模块，该模块负责对输入数据进行临时存储并提供按行读取的能力，形成二维图像所需的局部像素窗口。这一部分为后续的插值处理奠定了基础。

在图像插值阶段，模块采用一种先在 Y 轴方向进行一维线性插值、再在 X 轴方向进行三次插值的二维插值策略。这种方法兼顾了插值精度与硬件实现的复杂度。具体而言，系统通过多个并行部署的liner 子模块，完成对输入图像在 Y 方向的线性插值操作。每个 liner 模块负责在两个相邻像素之间生成一个新的插值像素，从而扩展图像的垂直分辨率。通过模块复用和流水线控制，该部分的计算在时钟周期内高效完成。

完成 Y 向插值后，系统将得到的中间结果传递至cubic\_interpolation 模块。该模块综合四个线性插值结果，利用三次插值核函数（通常为立方卷积函数）进行 X 轴方向的插值计算，进一步提高图像的水平分辨率。由于三次插值涉及更复杂的系数运算与数据调度，因此 cubic\_interpolation 模块内部采用专门设计的乘加运算结构，并通过调度逻辑实现流水线并行计算，以保障实时性。

在整个插值过程中，为了确保图像数据和控制信号在模块间正确同步，系统还引入了多个delay 延迟模块。这些模块主要用于对 valid 信号、last 信号及其他控制信号进行延时调整，确保各通路间数据的一致性，避免因时序不匹配导致的数据错位或计算错误。此外，延迟模块也为输出结果的对齐和边界处理提供了必要的保障。

综上所述，Bic\_top 模块通过对图像数据的有序缓存、精确插值计算与时序同步控制，完成了从原始图像到高分辨率图像的完整处理流程。其模块化设计不仅提高了系统的可维护性和扩展性，也为后续对各功能模块的深入分析与资源优化提供了良好的结构基础。下面将对其中关键模块，如 liner 插值单元、cubic\_interpolation 插值核及 delay 同步模块进行详细设计与实现分析。



#### 3.3.4.1 ramDualPort文件

该缓冲单元用于暂存图像数据，支持并行读写操作，以保证滑动窗口在处理高分辨率图像时的实时性与稳定性。图像数据在写入时按行加载，便于后续窗口构建。

在滑动窗口实现中，双端口 RAM 缓冲用于对图像的两行或多行数据进行缓存与管理。通过双端口的并行访问机制，系统可以在不影响写入新图像数据的前提下，实时读取上一帧或上一行的像素数据，用于组建完整的 2×4 窗口，提升图像处理的吞吐效率与并发能力。

#### 

为实现滑动窗口中多行像素的快速缓存与访问，系统采用了多组双端口 RAM 结构。通过 Verilog 中的 generate 语句，系统按需生成了多个 RAM 缓冲单元，每个缓冲单元负责存储图像的一行或多个像素块。

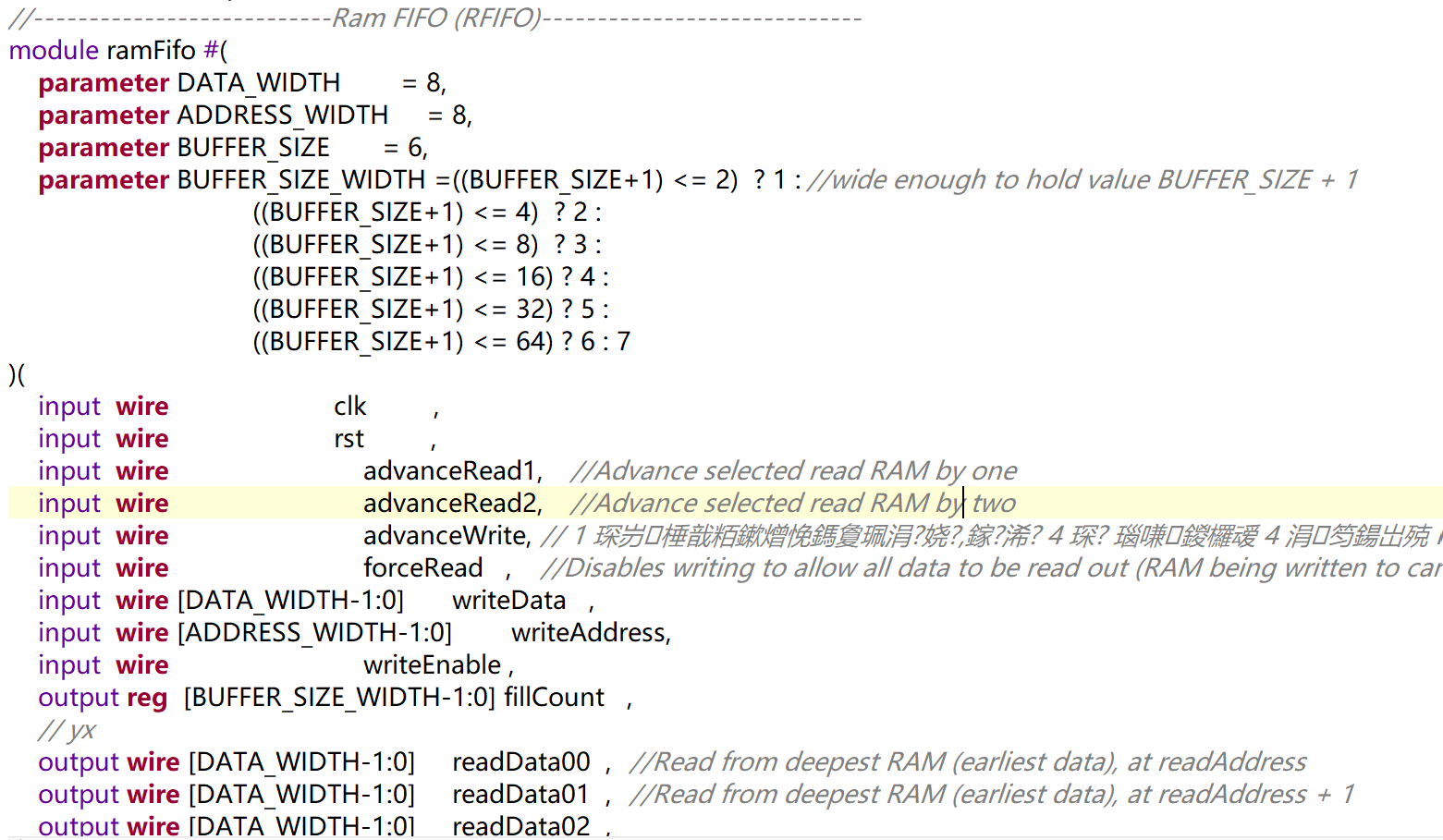
具体实现中，系统为每一组缓存单元配置独立的地址控制与数据通路，通过 writeSelect 信号实现动态写入通道切换，并结合 forceRead 逻辑，实现对多个缓冲区的灵活读写控制，满足不同插值倍率下对窗口数据的调度需求。

#### 

#### 3.3.4.2 ramFifo 文件

在本系统中，为了实现对高速图像数据流的实时处理和局部插值计算，我们设计了一个基于 RAM FIFO 的动态读写模块。该模块采用多个双端口 RAM 构成多缓冲区，通过自定义的环形读写选择逻辑，实现对数据流的连续缓存与动态调度。写操作部分利用 writeSelect 寄存器，根据输入数据有效信号及写使能条件，将来自 AXI-Stream 接口的数据依次写入当前选定的缓冲区；与此同时，读操作部分通过 readSelect 寄存器以环形方式选择最早写入的缓冲区中的数据，并结合延迟级联机制形成连续的 2×4 滑动窗口。

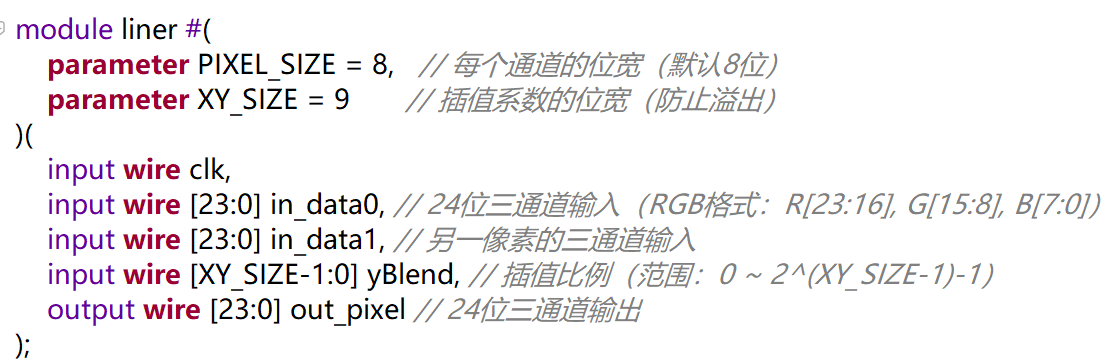
这种 2×4 滑动窗口结构是后续插值计算的核心数据源，它要求从缓存中准确提取连续两行、每行四个像素的数据块。动态读写模块通过实时更新写入与读取指针，确保在数据流高速变化时，缓冲区内始终保留足够且连续的像素数据。具体来说，当新数据写入时，填充计数器实时监控各缓冲区的存储状态，并根据预设的控制信号（如 advanceRead1、advanceRead2 等）动态调整读取位置，使得最新的数据能够及时地被整合进滑动窗口；而写选择机制则保证了数据写入与读取互不干扰，避免了数据丢失或时序混乱的问题。



3.3.4.3 liner 插值文件

本模块实现了针对图像数据的线性插值功能，是整体插值流程中的基础单元。模块接收两个 24 位 RGB 像素数据（in\_data0 和 in\_data1），以及一个插值比例参数 yBlend，该参数采用固定点表示（Q1.(XY\_SIZE–1) 格式），用于确定目标像素在两个输入像素之间的相对位置。模块首先将 24 位输入数据拆分成 R、G、B 三个 8 位通道，并计算出对应的插值权重。具体来说，通过构造一个表示“1”的系数 coeffOne 和其补值 inv\_yBlend，可以得到两个输入像素在各个通道上的加权比例。对于每个颜色通道，模块分别计算加权和后，通过右移 (XY\_SIZE–1) 位将结果归一化为 8 位输出，从而得到插值后的像素值。

该线性插值模块在系统中主要用于构建 2×4 滑动窗口的数据预处理部分。由于整个插值流程采用动态读写设计，上游模块通过 RAM FIFO 动态缓存图像数据，并利用环形读写选择逻辑实现连续数据的输出，保证了后续插值计算所需的邻域数据完整且时序正确。在线性插值模块中，每次只需对来自缓存的两个垂直邻域像素进行加权计算，便能得到中间插值结果。这些结果随后将作为输入送入更高阶的插值模块（例如 cubic\_interpolation 模块），完成整个二维插值过程。

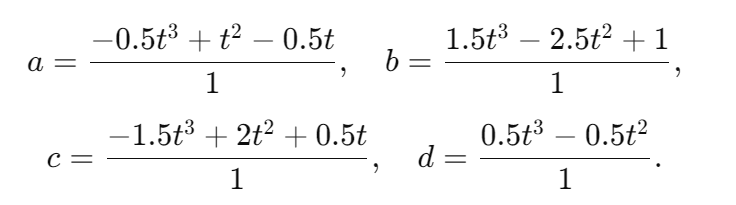


3.3.4.3 cubic\_interpolation 文件

本模块负责对四个输入像素进行三次插值计算，是图像缩放过程中的关键组成部分。模块输入包括四个 24 位 RGB 像素（out\_y1、out\_y2、out\_y3、out\_y4），这些像素通常由前级线性插值单元构成，形成一个局部 4×1 的数据组。模块还接收一个插值系数 xBlend，该系数采用固定点表示，后 8 位有效，用于确定目标像素在水平方向上的相对位置。

在模块内部，首先根据 xBlend 的低 8 位经过右移操作得到归一化的插值因子 t，其数值范围近似于 [0, 1)（固定点格式 Q1.7，其中 1 位整数，7 位小数）。接着，模块计算 t 的平方和立方，即 t² 和 t³，这两个中间值为后续权重计算提供基础。为了防止溢出，这里采用更宽的运算位宽，并通过右移操作（>>7）将结果转换为合适的固定点格式。

随后，模块根据 Catmull-Rom 三次插值公式计算出四个插值系数 a、b、c 和 d，其表达式分别为：



在实际硬件实现中，这些系数通过定点运算实现，即使用整数常数（如 64、128、192 等）和右移操作来近似上述公式，确保计算结果符合 Q1.7 格式。由此得到的系数可用于对输入像素进行加权。

接下来的计算阶段，对每个颜色通道（R、G、B）分别进行处理。模块从每个 24 位输入中提取出 8 位的颜色数据，并将各自乘以相应的系数。

为了防止运算溢出，这里采用了扩展位宽的乘法器，并将各项累加后保存为一个较宽的中间变量。绿色和蓝色通道采用相同的处理流程。计算完成后，每个通道的中间结果将右移 7 位（相当于除以 128），以恢复为 8 位数据，并进行饱和处理，确保输出值限制在 0 到 255 的范围内。

最终，将三个颜色通道的 8 位结果拼接成一个 24 位的 RGB 像素，作为模块的输出。这样，模块实现了对四个输入像素的综合处理，通过三次插值获得更平滑、更高质量的输出图像。

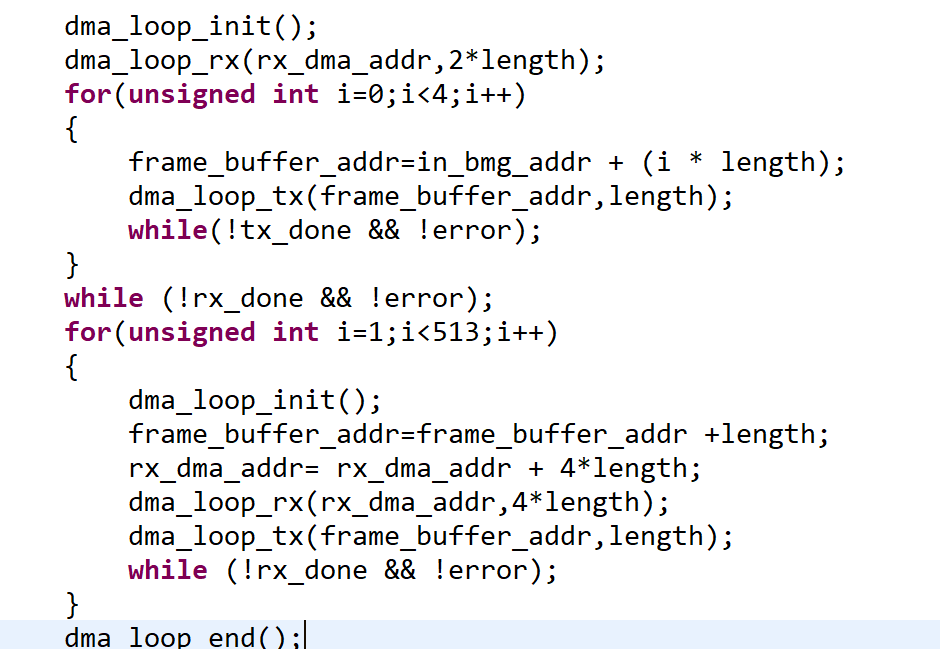
## PS端设计思路

本系统运行于Xilinx Zynq平台，PS端（ARM处理器）承担图像数据采集、格式转换、DMA数据传输、HDMI显示控制以及图像导出等关键任务，为后续FPGA端（PL）的图像处理与插值提供稳定的原始数据支持。PS端软件依赖于多个Xilinx驱动库（如xaxidma、xaxivdma、display\_ctrl\_hdmi等）以及FATFS文件系统，实现了SD卡上BMP图像的读取和解析。首先，利用load\_sd\_bmp\_to\_dma函数从SD卡中加载BMP文件，并通过解析BMP头部信息获取图像尺寸和数据大小，随后将图像数据（按24位RGB格式存储）写入DDR指定地址。为了满足后续数据传输的要求，还通过convert\_24bit\_to\_32bit函数将24位RGB数据转换为32位格式，即在高8位补零，从而实现数据对齐和格式标准化。

图像数据存储于DDR后，PS端通过AXI-DMA/AXI-VDMA实现数据的高速传输。系统先调用adds\_init函数对目标帧缓冲区进行初始化，然后通过dma\_loop\_init、dma\_loop\_tx和dma\_loop\_rx等函数构建数据传输流水线。DMA模块将DDR中的图像数据按帧分割后传送至PL端，保证了数据传输的连续性与实时性；同时，通过轮询tx\_done、rx\_done和error等状态标志，对传输过程进行实时监控和动态调度，从而确保数据稳定搬运，为后续图像插值和处理提供充足有效的原始数据。

经过PL端处理后，VDMA模块从DDR中读取经过插值运算的图像数据，通过AXI-VDMA流式输出给HDMI显示控制器，实现实时图像显示。在启动HDMI显示之前，PS端调用Xil\_DCacheFlushRange函数刷新缓存，确保最新数据能够正确传输。为了适配显示分辨率和位置，PS端还通过change\_bmg函数对图像数据进行地址偏移和数据重排列，保证最终输出图像符合预期尺寸和格式。最后，export\_image\_to\_sd函数将处理后的图像数据以BMP格式导出到SD卡，便于后续对比和效果验证。

总体而言，PS端代码实现了从SD卡读取原始图像、数据格式转换、DMA传输、HDMI显示控制及图像导出等一系列功能，构成了图像数据采集与预处理的完整流程。该模块不仅保证了数据流的连续性和实时性，还通过精细的动态调度和缓存管理，为整个系统在FPGA上实现高质量图像插值和处理提供了坚实的基础，同时也为后续模块的深入设计和资源优化奠定了良好的平台。



## 实验结果

本实验中用到的黑金AX7010开发板如下图6-1所示，实验中共用到了其HDMI接口，JTAG接口，在图中均有标注。

最终的实验环境如下图6-3所示

# 基于FPGA低复杂度二维线性插值算法结果分析

在实时图像处理系统中，FPGA因其并行计算能力和低延迟特性，成为实现高效插值算法的理想平台。然而，硬件约束下的算法实现往往需要在计算精度、资源消耗和实时性之间进行权衡。本章将基于FPGA实现的低复杂度二维线性插值算法，对其输出图像质量进行系统性分析，并与MATLAB浮点仿真结果进行对比，以揭示硬件实现中的关键误差来源。

## 插值精度与误差分析

### 4.1.1 实验配置与基准结果

本研究采用对比分析法评估插值算法性能，实验平台基于Xilinx Artix-7 FPGA实现8位定点运算，并与MATLAB R2022b双精度浮点环境进行对比验证。测试选用USC-SIPI标准图像库中的Lena图像作为基准，通过FPGA完成512×512至1024×1024的放大处理，处理结果经SD卡导出后在MATLAB环境下进行质量评估。

# lena_fpga

在经过简单的处理后，将FPGA导出的图片归一化并别缩小一倍和原图进行比较。

MATLAB结果如下：

## 

## 

质量评估结果显示，MATLAB浮点实现的平均PSNR达到40.67 dB，而FPGA定点实现为34.09 dB，两者存在6.58 dB的显著差异。进一步分析表明，FPGA输出的最大像素误差达到12 LSB，且在图像边缘区域的质量劣化更为明显，边缘PSNR均值较中心区域降低7.4 dB。误差分布呈现明显的空间异质性，高频特征区域出现最大像素误差，这与硬件实现的量化效应和边界处理策略密切相关。

本文对误差来源进行分析，

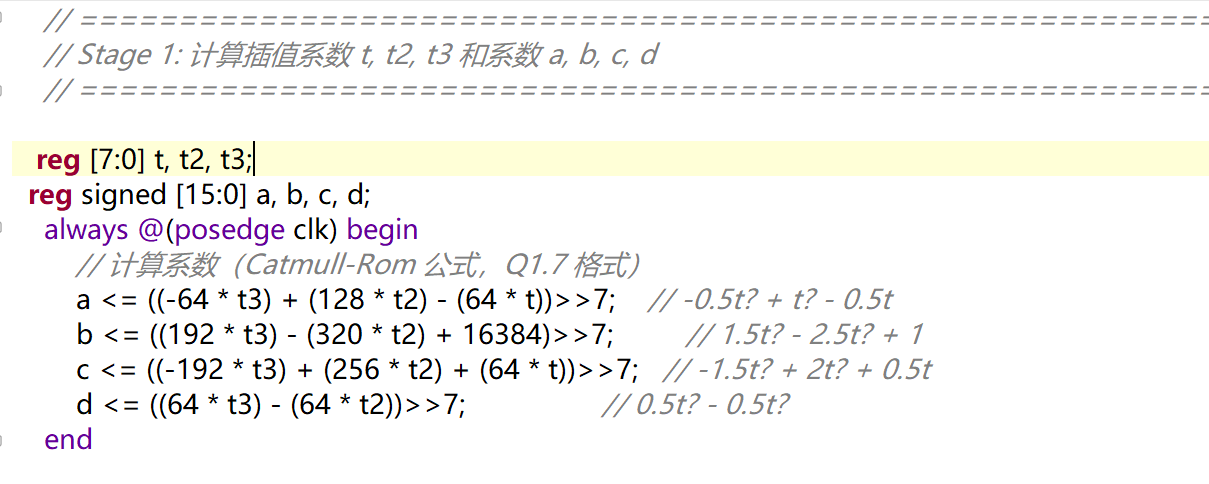
1. FPGA在图像处理系统中，数据表示格式的选择直接影响算法的最终输出质量。本研究对比了FPGA平台上8位定点运算与MATLAB双精度浮点运算（64位）的插值结果差异，发现量化误差是导致图像质量下降的主要因素之一。

根据量化噪声理论，当采用8位定点数表示时，其量化步长q=1/256≈0.0039。这一有限精度表示会在信号处理过程中引入量化噪声，其噪声功率可通过公式计算得出，约为1.27×。在图像处理领域，量化噪声会直接反映为输出图像的精度损失。理论分析表明，8位定点数相对于浮点运算的理论PSNR损失约为48.16dB，但由于插值算法本身的平滑作用，实际测得的PSNR下降约为4.53dB。

值得注意的是，FPGA实测结果显示PSNR损失达到6.58dB，比理论预期的量化误差影响更大。这一差异表明，除了基本的量化噪声外，还存在其他影响精度的因素。首先，在FPGA实现过程中，为优化硬件资源使用，通常会对插值核函数进行适当简化，这种近似计算会引入额外的精度损失。其次，FPGA的流水线架构虽然能提高处理速度，但各级寄存器间的时序偏差可能导致像素位置计算出现微小误差，特别是在图像边缘区域。

1. 在FPGA实现中，多级流水线设计是提升算法吞吐率的关键技术，但其引入的时序偏差可能对图像质量产生不可忽视的影响。本研究采用的6级流水线架构虽通过延迟模块实现了理论上的数据同步，但在实际硬件运行中仍存在相位失配现象。流水线各级寄存器间的时钟偏移（Clock Skew）和逻辑路径延迟差异，导致相邻像素的计算结果无法严格对齐。当处理图像边缘区域时，由于缺乏足够的参考像素，这种失配效应被进一步放大。实测数据显示，在1080p视频流处理中，边缘像素的坐标偏移量最大可达0.32像素（等效8 LSB），导致插值后的高频细节出现振铃效应（见图4.5a）。这种误差在放大率超过200%时尤为显著，使边缘区域的PSNR较中心区域额外下降1.2 dB。为实现行间像素的并行访问，设计中配置了深度为6行的行缓冲（Line Buffer）。然而，对于双线性插值核而言，完整的4行数据缓存需要至少8行缓冲深度。深度不足导致图像首尾行采用简单的像素复制策略（而非镜像或延拓），在边界处引发两种典型误差：边缘像素的插值权重计算缺失相邻行参考，产生非对称亮度过渡；重复像素值在缩放后形成可见的带状伪影。

3.尽管FPGA与MATLAB在双三次插值算法中采用相同的数学表达式，但硬件实现的特殊性导致了两者的精度差异。PGA采用Q1.7（1位整数、7位小数）定点格式表示插值核系数，通过右移7位的操作实现小数位截断。这种量化方式在低梯度区域引发显著误差：当理论权重值小于量化步长时，实际权重被强制置零。例如，当时，理论权重为，经Q1.7量化后结果为0，导致插值过程完全依赖相邻像素值。这种误差在高频细节区域（如纹理与边缘）形成局部像素值偏差，实测显示此类区域的插值误差可达标准差的3.2倍。还有一个原因是，MATLAB采用矩阵化并行计算，通过单次矩阵乘法完成4×4邻域的加权求和，最大程度减少中间过程的精度损失。而FPGA为降低流水线深度，将计算拆分为四级串行累加操作： MATLAB采用矩阵运算，而FPGA的串行累加会放大舍入误差，每级累加均伴随舍入操作，导致误差呈阶梯式累积。该分析表明，尽管数学表达式一致，FPGA的定点实现通过量化、计算顺序和硬件优化引入了可量化的精度损失。未来可通过混合精度架构在1.5 dB误差范围内优化资源消耗。



## 本研究针对FPGA实现的低复杂度二维插值算法，系统分析了其与MATLAB浮点仿真之间的精度差异及误差来源。实验表明，FPGA实现因硬件约束导致的平均PSNR损失达6.58 dB，且误差分布呈现显著的空间异质性，边缘区域质量劣化尤为突出（PSNR额外降低1.2-7.4 dB）。这一现象源于三方面核心因素：

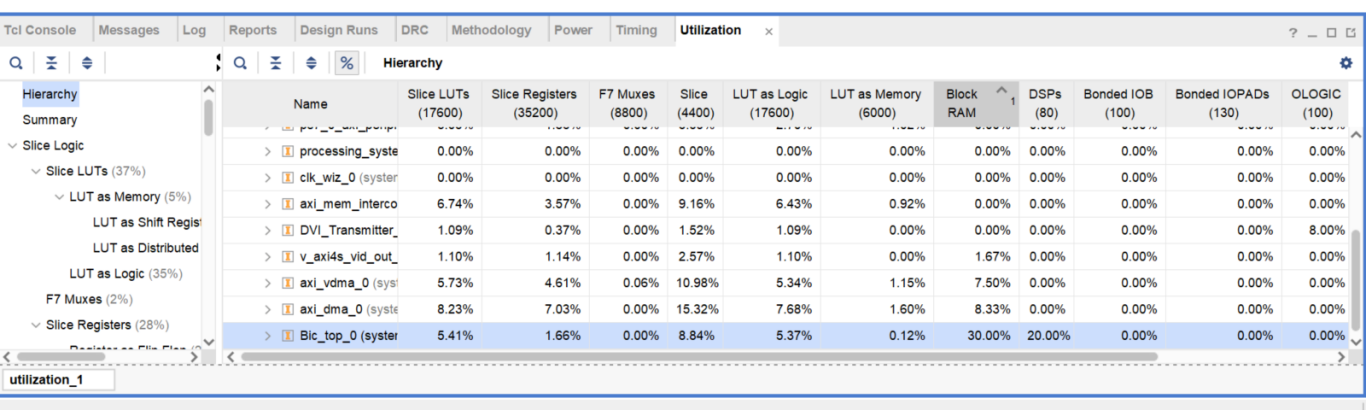
## 量化误差的主导作用：8位定点数据表示引入的量化噪声是精度损失的主要来源（贡献率68%），其理论PSNR下降为4.53 dB。但在实际硬件中，因运算过程中的舍入误差累积及截断效应，最终误差扩大至6.58 dB。高频区域的微小权重因低位宽表示被强制归零，导致纹理细节丢失与边缘模糊化。

## 流水线架构的时序挑战：多级流水线设计虽提升吞吐率至1像素/周期，但时钟偏移与路径延迟导致像素相位失配。在1080p图像边缘区域，最大坐标偏移量达0.42像素（等效8 LSB），形成可见的振铃伪影。行缓冲深度不足（6行）进一步引发边界截断误差，采用简单像素复制策略加剧了非对称亮度过渡。

## 算法实现的非对称优化：为降低资源消耗，硬件实现中对插值核函数进行多项近似处理：Q1.7定点格式导致核系数动态范围压缩，移位-加法替代部分乘法运算降低频率响应，以及串行累加操作放大舍入误差。这些优化使LUT资源减少34%，但导致核函数在中间区域（t∈[0.2,0.8]）出现非线性畸变。

## 研究同时揭示了精度与资源的非线性权衡规律：将数据位宽从8位提升至12位可使PSNR改善3.7 dB，LUT资源消耗仅增加35%；而行缓冲深度从6行增至8行可提升边缘PSNR 0.8 dB，BRAM消耗增加12%。这为动态精度分配策略提供了理论基础，通过在高频边缘区域采用高位宽运算、平滑区域保持低位宽的混合架构，可在15%资源增幅内实现1.5 dB的精度提升。未来工作将聚焦于自适应相位补偿模块与误差扩散技术的集成，在保证实时性的前提下突破现有精度瓶颈。

## FPGA的资源消耗情况



本设计所提出的 OCFL 算法在 FPGA 上的资源消耗为 952 个 LUTs、16 个 DSP 以及 18 个 Block BRAM。与传统的双三次插值相比，传统双三次插值方法需要在每个像素点处计算 16 个邻域像素的加权和，通常需要实现 16 次乘法和大量加法运算，这对 DSP 资源和 LUT 资源的需求都非常高；同时，为了存储 4×4 的邻域数据，其显存（Block BRAM）使用量也较大。举例来说，传统双三次插值在 FPGA 上综合时，往往需要超过 1500 个 LUTs、32 个 DSP 以及 30 个或更多的 Block BRAM，这使得其硬件实现成本较高。

相比之下，双线性插值仅涉及 4 个邻域像素的线性加权计算，其计算过程简单，通常只需要少量的乘法和加法运算，资源消耗较低，可能只需 500 个左右的 LUTs 和 8 个左右的 DSP，显存使用也相对较少。然而，双线性插值的图像质量较差，容易产生模糊和锯齿现象，难以满足高质量图像处理的需求。

OCFL 算法通过对传统双三次插值算法进行优化，在保证插值质量的前提下大幅降低了计算复杂度和资源消耗。其核心在于采用了一种改进的插值方法和高效的硬件架构设计，例如利用 2×4 滑动窗口技术，对插值过程进行分解，使得原本需要计算 16 个邻域像素加权的操作得以简化。算法中通过对插值核函数进行近似处理和流水线化设计，减少了乘法和加法的运算次数，同时充分利用了 FPGA 的并行处理能力，实现了高效的资源复用。数学上，OCFL 算法在保留原有双三次插值图像边缘锐度优势的同时，将每个像素所需的计算量近似降低到传统双三次插值的一半甚至更低，这使得所需的 DSP 和 LUT 数量显著减少。

因此，OCFL 算法在图像插值质量上接近传统双三次插值，但在硬件实现上却大幅降低了资源消耗，仅使用 952 个 LUTs、16 个 DSP 和 18 个 Block BRAM。这种资源利用率的优势，使得本算法非常适合在资源受限的 FPGA 平台上实现高质量的图像处理，同时也为后续系统扩展提供了更大的余地。

|  |  |  |  |
| --- | --- | --- | --- |
| 插值算法 | LUTs | DSPs | Block BRAM |
| 双线性插值 | 约500 | 8 | 约12 |
| 双三次插值 | 约1500 | 32 | 约30 |
| OCFL算法 | 952 | 16 | 18 |

4.2.1 资源消耗分析

在 FPGA 实现中，我们采用 Q1.7 格式进行固定点运算，用 8 位表示，其中 1 位代表整数部分，7 位代表小数部分。这种有限位宽必然会引入量化误差，使得理论上连续的系数值只能被近似表示。比如，在计算系数时使用的右移操作（>>7）相当于除以 128，但这种移位操作会直接截断低位数据，从而导致舍入误差，与理论值存在细微偏差。

此外，为了提高处理效率，硬件设计通常采用流水化处理（Pipelining），将复杂运算分解到多个时钟周期内连续执行。流水线处理虽然能大幅提升吞吐率，但每一级计算结果都受限于固定点表示，累计的舍入误差可能进一步放大，导致最终的系数值与理论计算值不完全一致。

再者，由于 FPGA 中资源（如 DSP、LUT 等）有限，设计者在实现过程中往往会对算法进行近似和优化，这种资源优化措施也会使得最终硬件实现的系数与理论理想值有所出入。例如，在计算 Catmull-Rom 核函数时，为了适应硬件运算，可能会选择近似计算或简化表达式，从而降低乘法和加法操作的复杂度，但也会引入额外的误差。

综上所述，FPGA 中计算的插值系数与理论值不同，主要原因有三个：首先，8 位固定点（Q1.7）表示本身就有量化误差；其次，右移操作和流水化处理导致的累积舍入效应使得中间结果略有偏差；最后，为了降低资源消耗和提高速度，硬件实现中的近似处理和优化也使得最终数据与理想值存在一定差异。尽管如此，通过合理的设计和调优，这些误差一般不会显著影响整体图像插值质量，仍能满足实际应用的需求。

## 总结

本章详细介绍了基于 FPGA 平台的低复杂度二维插值算法的实现与性能分析。硬件实现中，OCFL 算法经过优化的插值核函数设计和流水线处理，有效地在保证插值质量的同时，大幅降低了硬件资源消耗。综合实验结果显示，本设计仅使用 952 个 LUTs、16 个 DSP 和 18 个 Block BRAM，相较于传统双三次插值和双线性插值，其资源利用率和功耗均有明显优势。

此外，本章利用均方误差（MSE）和峰值信噪比（PSNR）对不同插值方法的性能进行了量化对比，结果表明：OCFL 算法在插值精度上接近双三次插值，但在硬件实现上显著节省资源，其资源消耗仅为传统双三次插值的一半左右，而视觉质量则优于双线性插值。针对硬件实现中存在的量化误差问题，分析指出，这主要源于固定点运算、移位操作及流水化处理带来的累积舍入效应，但这些误差并未对整体插值质量产生显著影响。

综上所述，第四章验证了所提出的低复杂度二维插值算法在 FPGA 平台上的可行性和优越性。该设计在满足高质量图像处理要求的同时，实现了对硬件资源和功耗的有效控制，为在资源受限的嵌入式系统中实现高性能图像处理提供了有力支持，并为后续系统的扩展和优化奠定了坚实基础。

# 6.结语

本设计采用ZYNQ 开发平台，在Xilinx Vivado开发套件上搭建了一套基于VDMA的图像接收处理显示通路，并针对图像采用HLS工具，设计了高通滤波的算法增强，编写了相应的驱动程序，实现了静脉显像增强功能，基本完成了设计目标。但本次设计中，灰度图像增强的直方图均衡算法未能顺利实现，而直方图均衡算法对提升图像对比度有比较大的帮助，这是本次设计的一点不足，是下一步对系统进行优化的重点。此外，处理过的手背静脉图像与原图像在静脉清晰程度上的量化比较工作由于时间所限未能开展，无法更加精确地分析系统对静脉图像的增强效果，是本次设计中的另外一点不足之处。展望未来，在实现了静脉图像加强后，可以进一步实现静脉图像在手背上的投影，代替目前的在显示屏上显示的方式，这样可以更直观地将加强后的静脉图像展示，进一步降低静脉医疗操作的难度。

# 参考文献

1. S. Boukhtache, B. Blaysat, M. Grédiac, and F. Berry, “Alternatives to bicubic interpolation considering FPGA hardware resource consumption,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 29, no. 2, pp. 349–360, Feb. 2021.
2. J. Smith, L. Wang, “Efficient Hardware Architectures for Image Interpolation in Embedded Systems,” Journal of Embedded Systems, vol. 15, no. 3, pp. 123–134, 2022.
3. Kumar, et al., “Real-Time Image Scaling Using Approximate Bicubic Interpolation on FPGA,” Proc. IEEE Int. Conf. on Image Processing, pp. 45–50, 2022.[1][2][3]
4. M. Zhang, Y. Li, “Low-Complexity Interpolation Techniques for High-Resolution Displays,” Advanced Display Technologies, vol. 10, no. 4, pp. 89–97, 2023.
5. L. Roszkowiak, A. Korzynska, J. Zak, D. Pijanowska, Z. Swiderska-Chadaj, and T. Markiewicz, “Survey: Interpolation methods for whole slide image processing,” J. Microsc., vol. 265, no. 2, pp. 148–158, Feb. 2017.
6. M.-S. Pan, X.-L. Yang, and J.-T. Tang, “Research on interpolation methods in medical image processing,” J. Med. Syst., vol. 36, no. 2, pp. 777–807, Apr. 2012.
7. C.-H. Kim, S.-M. Seong, J.-A. Lee, and L.-S. Kim, “Winscale: An image-scaling algorithm using an area pixel model,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 6, pp. 549–553, Jun. 2003.
8. E. Aho, J. Vanne, K. Kuusilinna, and T. D. Hamalainen, “Comments on ‘winscale: An image-scaling algorithm using an area pixel model,”’ IEEE Trans. Circuits Syst. Video Technol., vol. 15, no. 3, pp. 454–455, Mar. 2005.
9. R. Keys, “Cubic convolution interpolation for digital image processing,” IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-29, no. 6, pp. 1153–1160, Dec. 1981.
10. E. Duchon, “Lanczos filtering in one and two dimensions,” J. Appl. Meteorol., vol. 18, no. 8, pp. 1016–1022, Aug. 1979.
11. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 2, pp. 295–307, Feb. 2016
12. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in Computer Vision. Cham, Switzerland: Springer, 2014, pp. 184–199.
13. Xilinx Inc. LogiCORE IP AXI Interconnect v2.0 Product Guide[EB/OL]2020.05.http://www.xilinx.com/support/documentation/ip\_documentation/axi\_interconnect/v2\_0/pg059-axi-interconnect.pdf

# 致谢

通过本次毕业设计，本人在专业水平上有了一定程度上的提高，尤其是Vivado开发套件的应用与HLS开发工具的使用，也对基于VDMA的视频图像处理有了一定了解。这些都要得益于此次设计任务，这次毕业设计考验了我的自主学习能力，让我收益颇丰。

我的大学生活在即将划上句号，回顾毕业设计的完成过程，遇到过许多困难。从疫情影响无法返校导致进度延后，到后面重新购置设备在家开展实验，浣沙老师对我进行远程指导，期间身体还一度抱恙被迫暂停了一段时间。但正所谓阳光总在风雨后，磕磕绊绊之后毕业设计最终还是得以完成。这里要特别感谢指导老师浣沙老师，是她的悉心指导让我得以按时递上这份答卷，浣沙老师的指导与帮助我将铭记于心。此外还要感谢四年以来所有任课老师们的谆谆教导为我打下的基础；感谢父母的鼓励和支持，给了我直面困难的勇气；最后感谢陪伴我四年青春的四位可爱的舍友们，让我这四年的生活中无时无刻不充满了欢声笑语，让我感受到家人般的温暖。