武汉大学[电工电子实验教学示范中心](http://eelab.whu.edu.cn/)

教学实验报告

《信号与系统实验2》

课程实验报告

|  |  |
| --- | --- |
| 院（系）名 称 ： | 电子信息学院 |
| 专 业 名 称 ： | 通信工程 |
| 学 生 姓 名 ： | 曹继东 2017301200179 |
|  | 方 建 2017301200173 |
|  | 王兴明 2017301200174 |

二○二零年七月

表1.学生信息表\*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名\* | 曹继东 | 学号\* | 2017301200179 | 报名照片 |
| 专业\* | 通信工程 | 手机\* | 17354422237 |
| 微信 | cao15939688596 | QQ | 2282319687 |
| Email\* | 2017301200179@whu.edu.cn | | |
| 姓名\* | 王兴明 | 学号\* | 2017301200174 |  |
| 专业\* | 通信工程 | 手机\* | 18487520200 |
| 微信 | ximwhite | QQ | 2975443325 |
| Email\* | ximwhite@qq.com | | |
| 姓名\* | 方建 | 学号\* | 2017301200173 |  |
| 专业\* | 通信工程 | 手机\* | 17354422237 |
| 微信 | cao15939688596 | QQ | 2282319687 |
| Email\* | 2017301200179@whu.edu.cn | | |

表2.报告报告评阅

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | **内容** | 存在问题 | **修改确认** | 评阅时间 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| **3** |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 综合评阅 | 评阅成绩**：**  评阅时间**：** | | | |

# 目 录

[目 录](#_Toc16751)

[1实验介绍](#_Toc7540)

[(实验名称，实验目的，实验内容简介等)](#_Toc27892)

[2 实验设计](#_Toc20409)

[2.1 实验原理](#_Toc32714)

[2.2系统设计](#_Toc16871)

[2.2.1 实验设备与测试环境](#_Toc12882)

[2.2.2顶层设计](#_Toc29628)

[2.2.3 组成结构](#_Toc4776)

[3 实验实施](#_Toc31449)

[3.1系统搭建记录](#_Toc26645)

[3.2 调试过程记录](#_Toc4323)

[3.2.1 硬件调试](#_Toc24359)

[3.2.2 软件调试](#_Toc3009)

[3.3 实验结果记录](#_Toc11118)

[(只记录讲义里要求结果)3.4面临的问题总结](#_Toc22759)

[4 实验结果与结论](#_Toc29739)

[4.1实验结果分析](#_Toc7110)

[.4.2实验结论](#_Toc19318)

[5 任务分工](#_Toc20688)

[5 致谢](#_Toc11181)

[6 参考文献](#_Toc8646)

[7 附录](#_Toc5407)

[7.1 项目代码清单](#_Toc15638)

# 1实验介绍

(实验名称，实验目的，实验内容简介等)

## 实验名称

FPGA实现基于双线性插值的图像缩放

## 实验目的

1. 学习verilog语言描述算法。
2. 掌握基于双线性插值法的图像缩放的FPGA实现。

## 实验内容简介

数字图像处理的对象因其涉及到社会的各个领域,正受到越来越多的关注,而图像缩放作为数字图像处理中的基本操作尤为重要,在社会的很多领域都需要对图像进行放大和缩小。实现图像的放大和缩小很容易,但是当放大或缩小的比率比较大时就很容易出现失真现象,因此必须进行改进。

通过对几种常用的图像缩放算法进行了比较 , 在权衡了算法复杂度、缩放效果和 FPGA 逻辑资源等3大因素后, 本文提出了一种双线性插值算法,用以改进图像的缩放质量，选择了双线性插值算法来实现图像缩放。 重点介绍了双线性插值算法和该方法的 FPGA 实现方法 , 包括图像数据缓冲单元、插值系数生成单元以及插值计算单元等。应用结果表明, 双线性插值算法达到了较好的效果。

# 2 实验设计

## 实验原理

* + 1. **算法选择**

图像缩放算法种类较多, 几乎都是通过插值算法实现的 ,传统的插值算法有:最近邻插值 、双线性插值和立方卷积插值等。

最简单的插值方法是最近邻插值, 与其他两种插值算法相比,最近邻插值具有简单快速的特点 ,但是对于近邻像素点的灰度值有较大改变, 细微结构是粗糙的 。双线性插值方法具有低通滤波器性质, 使高频信息受损,图像轮廓模糊,但边缘处的过渡比较自然。与最近邻插值相比 ,它的计算量较大 ,但缩放后的图像质量高 ,不会出现像素值不连续的情况,在对图像边缘质量要求不是非常高的情况下 , 这种方法是完全可以接受的。立方卷积法计算量要比双线性插值大很多, 但精度高 ,能保持较好的图像边缘细节 。

实际应用中, 因为 FPGA 内的逻辑资源有限,而算法越复杂占用的逻辑资源就越多, 对 FPGA完成图像的实时缩放有重要影响,所以实际应用中应选择合适的算法, 并非精度越高越好。综上所述, 本文选用双线性插值算法来实现图像缩放。

* + 1. **双线性插值算法**

双线性插值算法是一种比较好的缩放算法，它充分利用源图中虚拟点四周的四个像素点来共同决定目标图形中的一个像素值，因此缩放效果要比最临近插值算法好的多。

在双线性插值中，新创建的图像的像素值是由原图像位置在它附近的4个临近像素的值通过加权平均计算得出的，然后由新图像的某个像素值映射到原始图像处，对取整得到并得到和的值，利用双线性插值得到像素点的值并写回新图像。

对于目的图像中的某点坐标，通过乘以缩放倍数（srcwidth/dstwidth、srcheight/dstheight）得到一个浮点坐标（i+u，j+v）（其中i，j均为浮点坐标的整数部分；u，v为浮点坐标的小数部分），则这个浮点坐标（i+u，j+v）处的像素值f(i+u，j+v)可以由原图像中的坐标点（i，j）、（i+1，j）、（i，j+1）、（i+1，j+1）所对应的四个像素值来共同决定，即

图2.1为双线性插值的原理图。

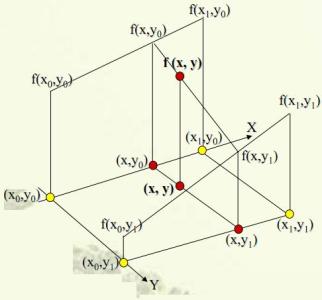


图2. 1 双线性插值原理图

设原始图像中相邻两行中相邻两列的4个像素点为，水平和垂直两个方向点距为单位长度。4点坐标分别为，目标像素点的坐标为分别是目标像素点与原始图像中临近点的水平和垂直坐标方向上的两个增量。那么，采用双线性插值法得到目标插值像素点A的灰度值为：



考虑以FPGA来实现插值运算时，共需要8个乘法器，2个减法器和4个加法器，占用了较多的逻辑资源，所以将其改写为：



我们已经可以同时求出四个像素点的坐标，但是如何通过这四个坐标同时求出对应的像素值仍需解决，由于待缩放的数据是先缓存进RAM的，如果待缩放的图像数据仅仅缓存在一个RAM里，是不可能通过四个像素点的坐标同时访问这个RAM，即不可能同时求出对应的四个像素点的值。所以，可以通过牺牲面积换取速度的方式，即将RAM复制4次，每个RAM都缓存整个待缩放的图像数据，这样四个像素点的坐标就可以通过访问不同的RAM来同时访问对应的四个像素值了。虽然牺牲了FPGA内部的存储资源，但是提升了双线性插值算法的速度。

* + 1. **双线性插值的FPGA实现**

我们发现使用FPGA实现双线性插值有以下难点：

1. 如何处理算法中出现的小数；
2. 如何同时求出相邻的四个点的坐标以及系数；
3. 如何使用这四个坐标同时输出相邻的四个像素值；
   * 1. **RGB图像存储格式**

真彩色图像中，每一个像素由RGB三个分量组成，每个分量各占8位，取值范围为0到255，每个像素24位，每个像素由8位表示，每个真彩色图像由3个矩阵表示，如下图：

|  |  |  |
| --- | --- | --- |
|  |  |  |



图2. 2 RGB图存储格式

## 2.2系统设计

### 2.2.1 实验设备与测试环境

* **软件**：
* **硬件：**未能实现硬件测试

### 2.2.2顶层设计

(TOP entity interface)

整个算法主要由图像数据缓冲单元，系数生成和逻辑控制模块，双线性插值计算模块等3大模块来实现，如图2.2所示：

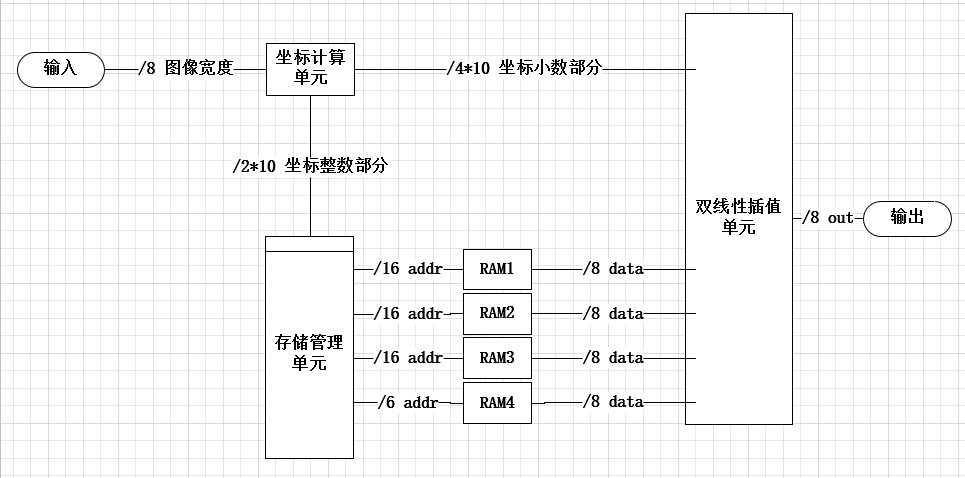


图2. 3 用FPGA实现双线性插值的系统框图

### 2.2.3 组成结构

(structure figure)

**2.2.3.1图像数据缓冲**

数字视频信号的数据量很大 , 传输带宽特别高。在用硬件系统对数字视频图像进行实时处理时,需要建立一个图像数据的存储缓冲机制 ,为插值运算及时准确地提供图像数据 。一般情况下,需要采用片外大容量高速存储器存储至少两帧的图像数据, 利用“Ping-Po ng”轮换机制实现图像数据缓冲。当需要使用一帧内像素点数据时 ,可以及时从一个存储器中读取 ;与此同时,连续的图像数据被存至另一个存储器中。但是, 这样的帧存储器不仅增加了成本 ,还要在 FPGA 内利用逻辑模块设计帧存控制逻辑。为了降低成本 ,提高系统集成度 ,本文利用 FPGA 的内部逻辑资源为每一彩色通道设计了一组由3个双端口 RAM(DPRAM)组成的图像数据缓冲阵列 , 并将这样一个缓冲阵列称为行缓冲器。

**2.2.3.2 插值系数生成**

根据双线性插值运算公式及二维图像插值的FPGA 实现方法, 两个运算系数 d x 和 dy 需要准确及时地求出, 提供给插值运算单元,并且只有当生成的系数在时序节拍上与图像数据存储 、插值计算单元的时序配合一致时, 才能有效地完成整幅图像的插值运算。因此 ,在生成系数的同时 ,必须能够给出系数在原图像中对应邻近像素点的坐标信息,即在 DPRAM 中的存储地址 , 以便图像数据缓冲模块可以同步地取出参与运算的 4 个图像数据。

首先需要确定缩放因子 S 。对于以单位长度“1”来表示相邻两个像素点的距离的模型进行缩 放时 ,以 S 为间隔形成一个新采样格栅, 重新对原图像在水平和垂直两个方向上进行重采样, 新图像的像素点距离即为 S 。如果把分辨率为 640 × 480 的图像转化为 800 ×600 的图像 ,就需要放大1.25倍，则缩放因子：



如果把如果把分辨率为 1 024 ×768 的图像转化为800×600 ,将图像缩小0.78125倍时,缩放因子为:



上述两种情况下 ,目标图像在水平和垂直两个方向的缩放比例是一样的。若在水平和垂直两个方向上的缩放比例不一致, 则缩放因子也就不相同 ,分别定义为。那么，对应于新图像中坐标为的像素点，系数可以由下列公式得到：



“”表示向下取整。

**2.2.3.3 双线性插值运算单元**

在获取插值像素点相邻4点的图像数据和插值系数后 ,就可以进行双线性插值运算了 。依据插值公式 设计了一种具有流水线特性的运算逻辑结构 ,如图2.3所示：

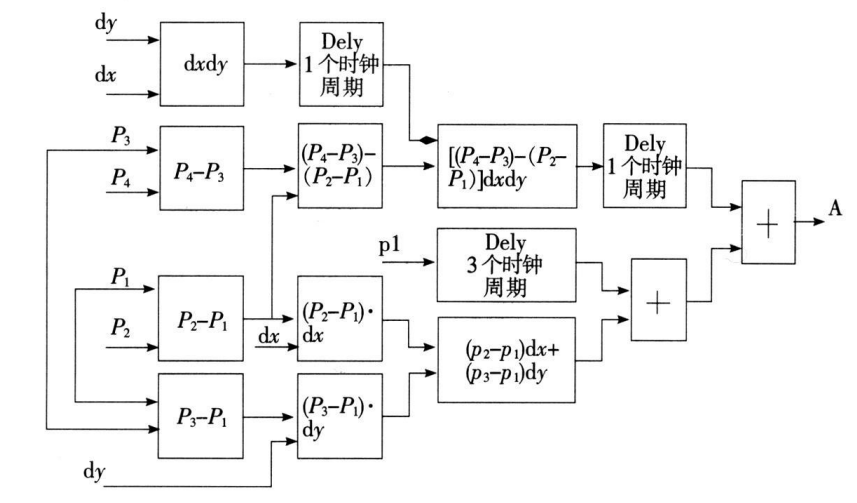


图2. 4 双线性插值运算单元

图中,由 DPRAM 实时输出,  则由系数生成单元提供 。逻辑实现过程中 ,要特别注意时序节拍的控制 ,力求时序同步 。当某个运算步骤输出数据与其他运算的输出值在某个时钟不能同步到达下一个运算器时, 在数据线上插入延时单元 , 使它们能达到同步, 并且适当增加乘法器和加法器等的流水线延时。

**2.3 安全注意事项**

# 3 实验实施

## 3.1系统搭建记录

该实验需要的软件有matlab，python，quartus II，matlab用于图像数据处理，即将RGB彩图分离出R，G，B三种颜色，并存储为可供quartus生成RAM的mif文件，quartus II 用于实现双线性插值处理图像，python用于处理后图像数据的分析和验证。双线性插值实现图像缩放的FPGA项目系统搭建步骤如下：

步骤1：项目创建

在quartus II 中创建“new project”，命名为“bilinear”,并且创建文件“bilinear.v”（顶层模块），“coordinate\_trans.v”（坐标变换模块），“mem\_control.v”（图像数据存取模块），“bilinear\_cal.v”（双线程插值单元）。并且，将matlab处理后的mif文件放入该项目，供后续IP核创建使用。

步骤2：RAM创建

在“tools”里面选择“megaWizard Plus in Manager”,新建RAM，如图3.1所示，并命名为“ram\_control”。

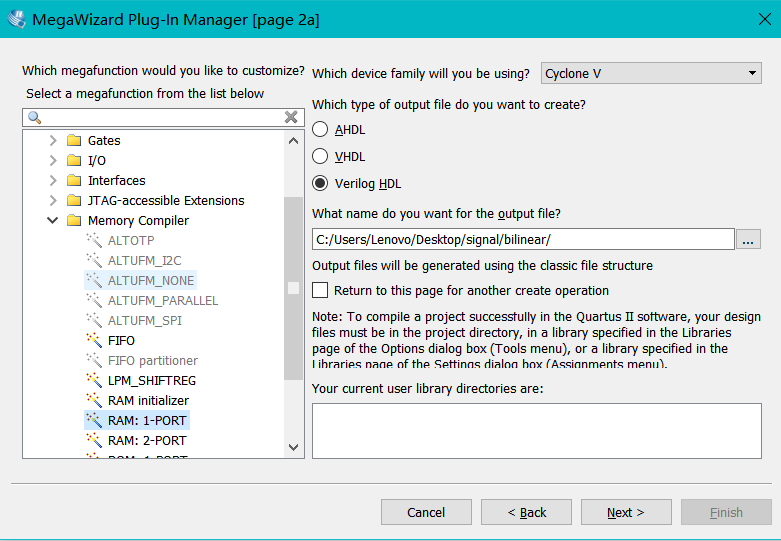


图3. 1 quartus II 新建RAM

输出位宽选择8位，存储容量选择16384字节，如图3.2所示。

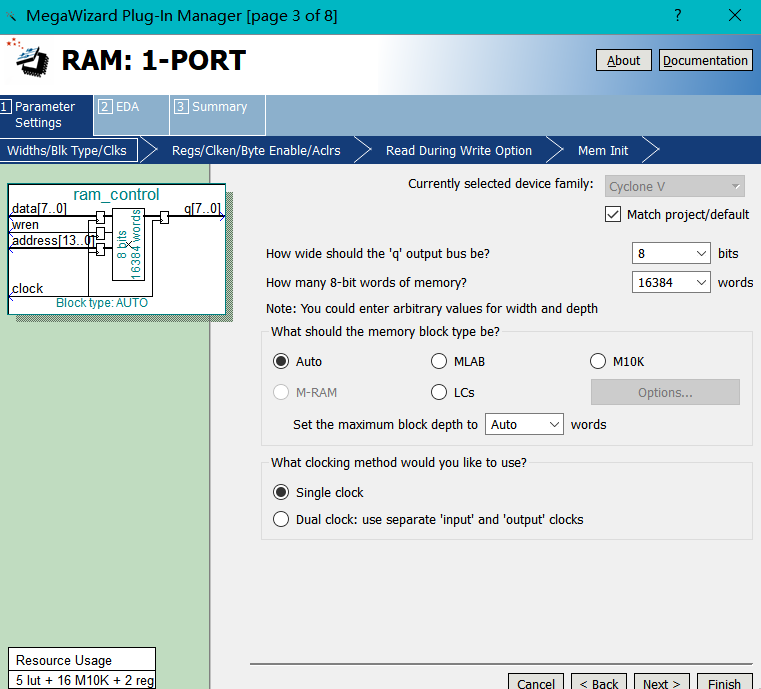


图3. 2 RAM 字长和字节选择

内存初始化选择前面步骤添加进来的mif文件。

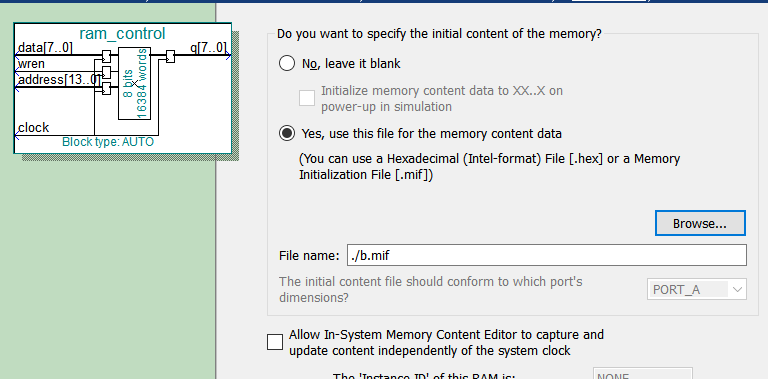


图3. 3 RAM初始化文件选择

步骤3：testbench生成

启动 testbench writer 直接生成testbench模板“bilinear.vt”：如图3.4所示

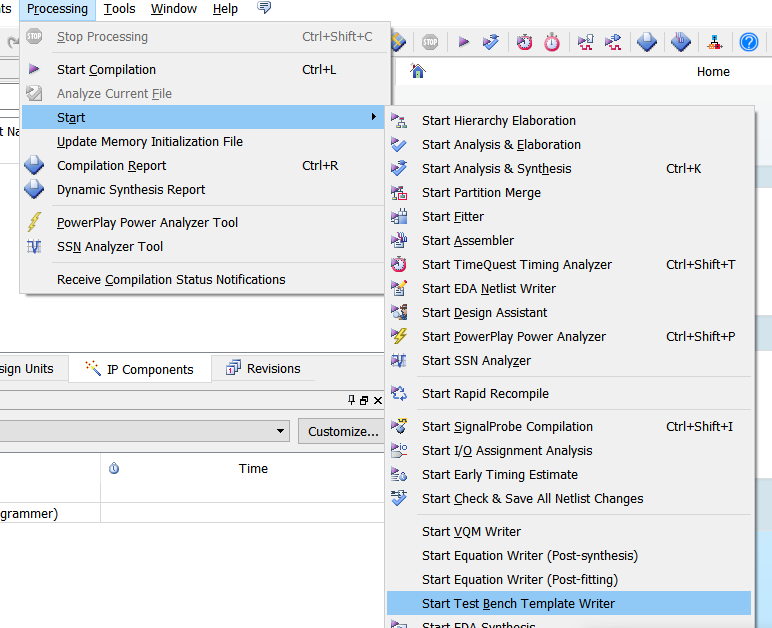


图3. 4 生成testbench模板

之后添加仿真文件到simulation，如图3.5所示

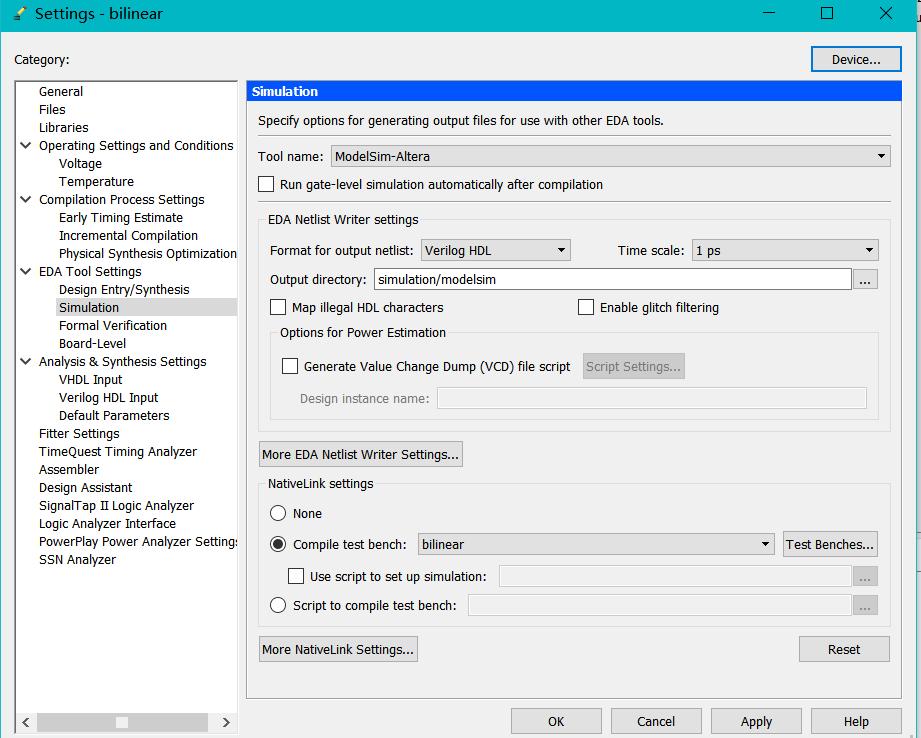


图3. 5 添加testbench文件到simulation

至此，FPGA项目搭建完毕。

## 3.2 调试过程记录

### 3.2.1 硬件调试

（硬件不做记录）

### 3.2.2 软件调试

问题1:modelsim仿真无任何波形显示

解决过程：按理说，仿真波形中至少会出现clk时钟信号，但仿真时无任何波形显示，并且“objects“里面无任何变量显示，初步怀疑是仿真文件编写错误，于是改为用quartus II的模板生成testbench模板，之后问题解决，波形出现。

问题2：仿真时输出图片处理结果全为0

解决过程：查看送入“ram\_control”内存处理单元的数据，wren=0，clk正常，说明可以正常读取数据，再查看送入的地址（address）发现地址已经超出定义的RAM范围，说明代码编写有误。检查代码，发现地址截取有误，更改后输出数据正常。

问题3：pyhton验证仿真结果发现显示图片颜色异常

解决过程：查阅资料后发现是因为opencv的imread读取RGB图像为[B,G,R]格式，而matplotlib.pyplot显示图像的格式为[R,G,B]，改正后图片显示正常。

问题4：modelsim仿真后存储的图像数据中多了一个数据点

解决过程：

## 3.3 实验结果记录

Python验证图像缩放的双线性插值算法：

图3.6为原始图像提取出R，G，B，后分别显示的图像，图左为双线性插值放大后的图像，图右为原始图像。图3.7为python实现双线性插值放大后图像和原始图像对比，可以看到，python实现的双线性插值算法具有较好的实现效果。

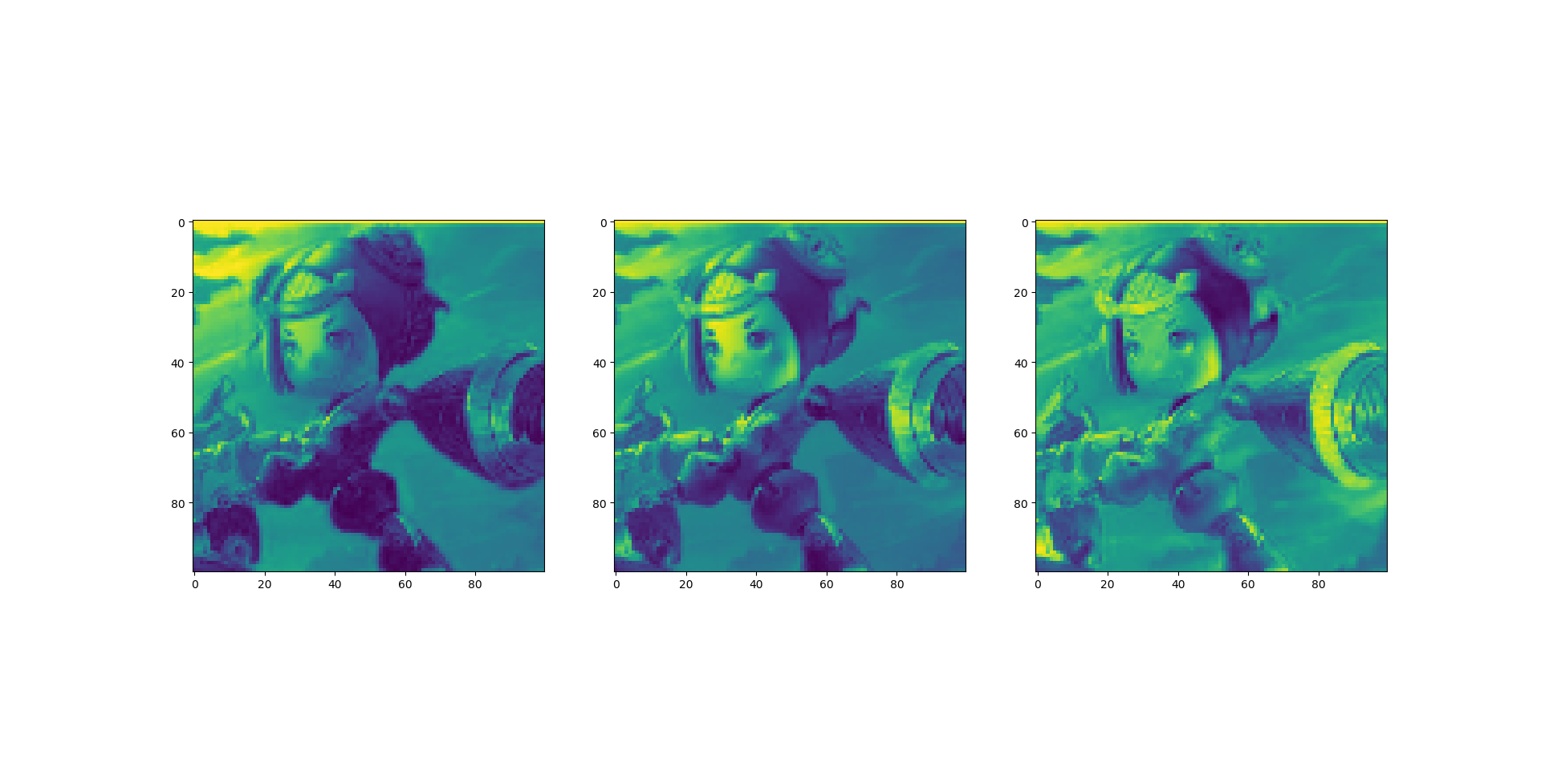


图3. 6 RGB分离后的像

|  |  |
| --- | --- |
|  |  |

图3. 7 pyhton实现双线性插值放大后图像和原始图像对比

FPGA验证图像缩放的双线性插值算法：

图3.8为仿真输出波形。

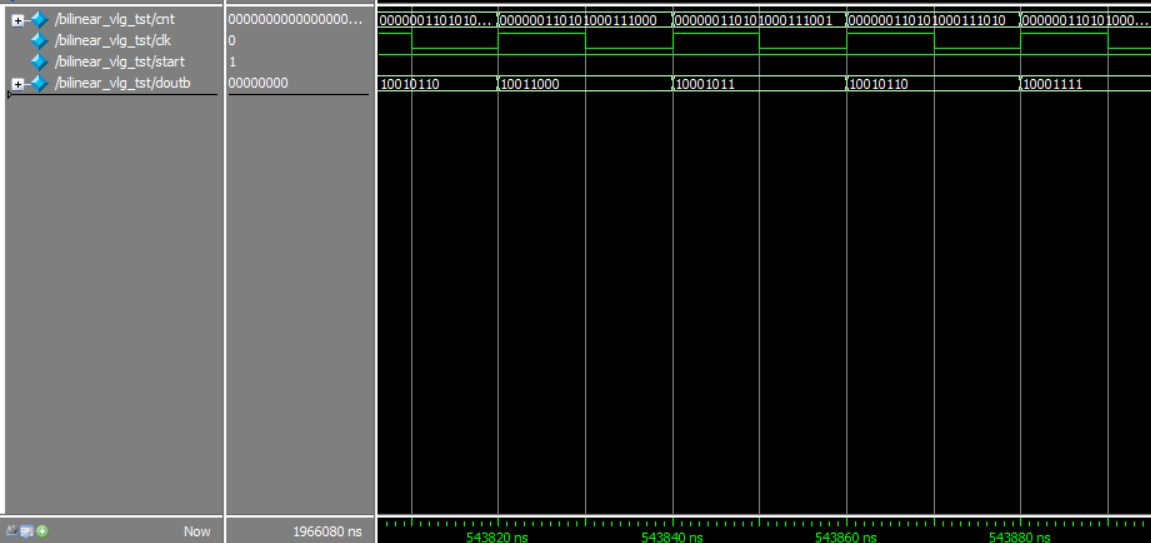


图3. 8 仿真输出波形

图3.9为pyhton加载仿真后的数据，data\_R为双线性变换后R通道数据，data\_G为G通道书上，data\_B为B通道数据，data\_RGB为通道混合后数据。

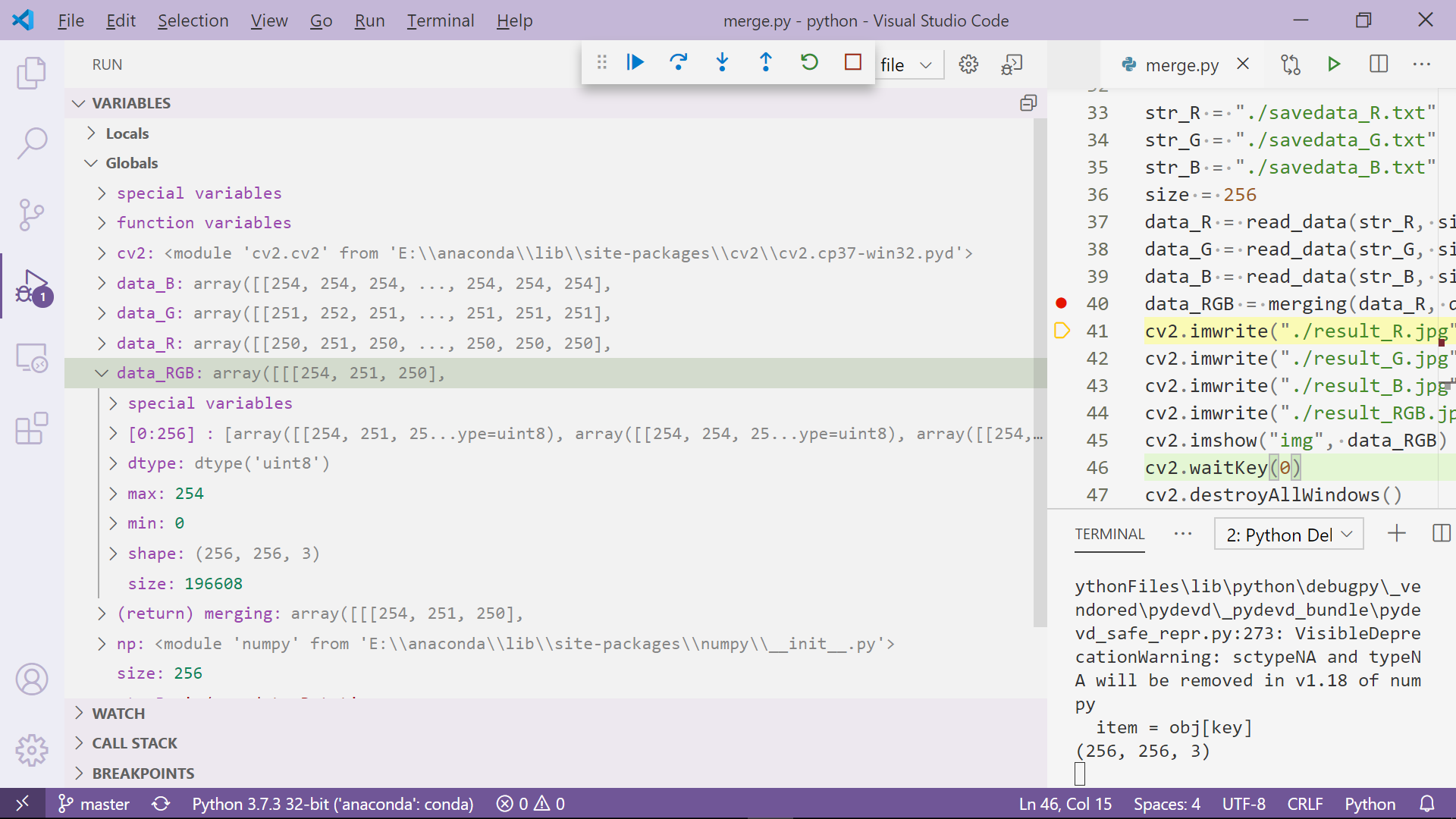


图3. 9 仿真输出数据

图3.10为经过FPGA处理后还原的图像，与原图对比，发现经过双线性变换后，图像有的像素点出现了模糊的现象。

|  |  |
| --- | --- |
|  |  |
|  |  |

图3. 10 FPGA实现双线性插值算法处理后的图像

## (只记录讲义里要求结果)3.4面临的问题总结

该项目利用双双线性插值算法实现了图像的缩放，然而由于设计水平有限，目前，该项目还存在以下问题：

一是内存资源占用过大：由于我们在设计时采取了“以空间换时间”的设计思路，在读取每进行一次双线性插值运算需要的四个数据点时实例化了四个RAM，即内存占用大了四倍，之所以这样做是因为它能够简化时序设计。如果设计的目标是要尽可能地节约内存资源，那么可以考虑用有限状态机来控制，即每进行一次双线性插值计算需要4个时钟周期。

二是图像的RGB分离是提前进行的：理想情况下，应该是输入一幅RGB图像，然后输出一幅缩放后的RGB图像，然而我们的设计是预先将图像的RGB分离，存储为mif文件，再处理单通道图像。如果将RGB图像所有数据都存储于一个mif文件中，那么利用verilog实现RGB的分离将会变得困难，基于这样的考虑我们设计的系统一次只处理一个通道数据，如果处理一幅RGB图像需要3次处理过程。

# 4 实验结果与结论

## 4.1实验结果分析

## .4.2实验结论

（实验是否成功）

(实验效果评价等)

# 5 任务分工

|  |  |
| --- | --- |
| **任务** | **执行者：** |
| 背景理论与研究 | 方建 |
| 基于matlab数据预处理 | 方建 |
| 双线性插值算法的verilog实现 | 王兴明 |
| 基于modelsim的双线性插值算法仿真 | 王兴明 |
| 基于python双线性插值实现效果的验证 | 曹继东 |
| 报告撰写 | 曹继东 |

# 5 致谢

# 6 参考文献

1. TOYOSHIMA M. Trends in satellite communications and the role of optical free-space communications Invited [J]. J Opt Netw, 2005, 4(6): 300-11
2. Smutny B. In-orbit verification of optical inter-satellite communivation links based on homodyne BPSK[R].Bellingham: SPIE,2008
3. Smutny B. In-orbit verification of optical inter-satellite

# 7 附录

## 7.1 项目代码清单

（代码文件名称及简介）

（代码下载地址,如网盘，github地址等）