景嘉微杯

一种应用于图形显示的Upsampling IP

队伍编号：CICC1837

团队名称：啊对对对对对对

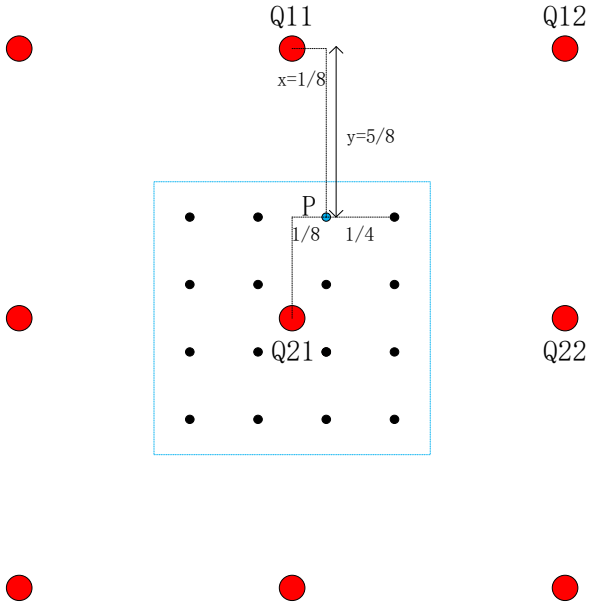
团队成员：朱涛、徐振华、李子倞

目录

CONTENT

软件算法

新型双线性插值



新型双线性插值原理图

$$P = (1-x)(1-y)Q_{11} + x(1-y)Q_{12} + (1-x)yQ_{21} + xyQ_{22}$$

传统双线性插值其目标图像会含有原图像的所有像素点，放大4倍会导致整体图像的偏移。

求解新型双线性插值点P像素值：
P点相对于Q11的偏移为x=1/8,y=5/8 ， 带入传统的双线性插值公式，即可得P点的像素值

均值调整

1k图像经均值下采样获得，原4k图像16点平均值等于相应位置1k图像的像素值，在还原的4k图像中，我们也希望仍然保持这一性质。

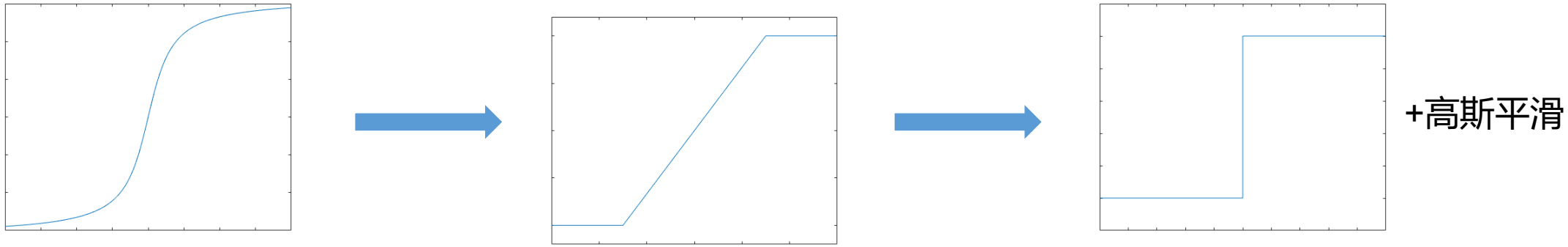
设所求的1k像素为average，16个4k像素点为dot_16_4k，经均值调整过后新的16个点为adjust_dot_16_4k，则有如下等式成立：

$$adjust\_dot\_16\_4k=dot\_16\_4k-(fix(\frac{sum(sum(dot\_16\_4k))}{16})-average)$$

sum为求和函数，双重sum表示对16点的求和，fix为向0取整函数。
对于新的16点，若其值小于0，则赋值为0，若其值大于255，则赋值为255。

边缘插值

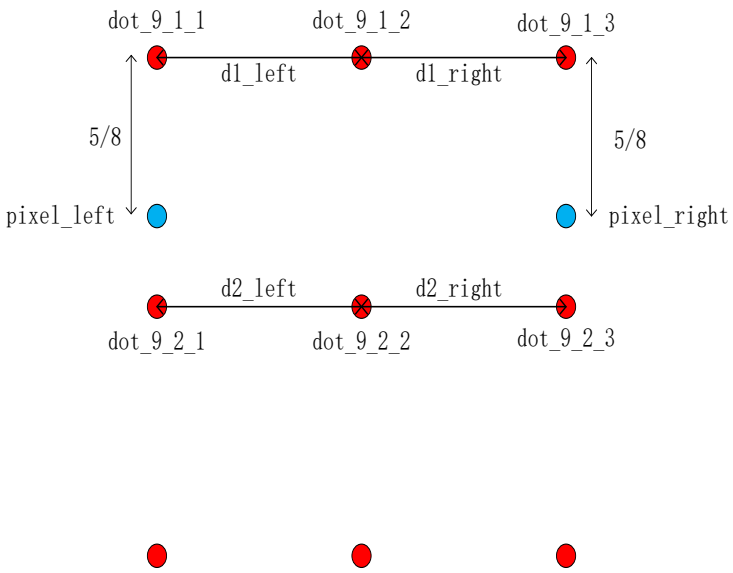
均值下采样与双线性插值均会导致高频分量的丢失，为了弥补这一缺失，我们提出了一种边缘插值的算法，其利用1k图片的9个像素点产生中心点对应的4k图片的16个像素点。



y=Aarctan(kx)+B　　**分段函数**　　**阶跃函数+3x3高斯平滑**

添加两次均值调整得插值算法：边缘插值+均值调整+高斯平滑+均值调整

边缘插值



边缘插值原理图1

以水平方向边缘插值的第一行为例：
先用线性插值求出左右两边的像素值，即图中的蓝色圆点pixel_left和pixel_right

$$pixel\_left=fix(\frac{5}{8}\times dot\_9\_2\_1+\frac{3}{8}\times dot\_9\_1\_1)$$

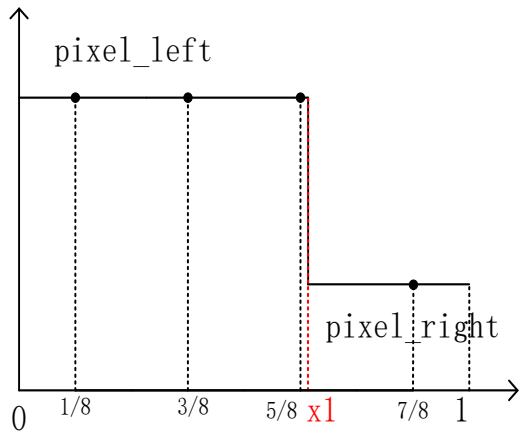
$$pixel\_right=fix(\frac{5}{8}\times dot\_9\_2\_3+\frac{3}{8}\times dot\_9\_1\_3)$$

然后求出第一行左右两点相对于中间点的梯度d_left和d_right：

$$d\_left=fix(\frac{5}{8}\times d2\_left+\frac{3}{8}\times d1\_left)$$

$$d\_right=fix(\frac{5}{8}\times d2\_right+\frac{3}{8}\times d1\_right)$$

边缘插值



边缘插值原理图2

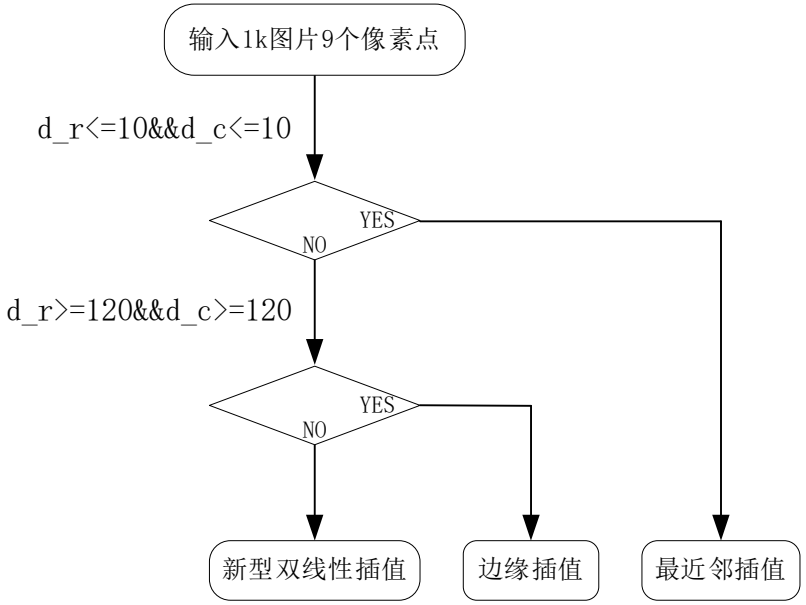
所求4点位于1/8,3/8,5/8,7/8，当所求像素横坐标小于x1，其像素值等于pixel_left，当所求像素横坐标大于x1时，其像素值等于pixel_right，而x1的计算公式如下：

$$x1=\frac{abs(d\_right)}{abs(d\_right)+abs(d\_left)}$$

同理可求出水平方向其它3行的像素值以及竖直方向4列的像素值，加权取平均即得到最终经边缘插值求得的4k图片16点像素值

边缘插值类似于多项式拟合，但不像多项式在无穷处会发散，是一种值得深入研究的新的方法。

分区域插值



分区域插值

分区域插值指利用3x3sobel算子求出水平梯度d_r和竖直梯度d_c，
若 d_r<=10&&d_c<=10，则采用最近邻插值；
若 d_r>=120&&d_c>=120，则采用边缘插值；
其余区域采用新型双线性插值。

高斯平滑

边缘插值算法过分的拉大了边缘的梯度，中间不存在过渡区，为了缓解这一问题，我们在生成4k图片的基础上对4k图片进行的3x3高斯平滑。

A11	A12	A13
A21	A22	A23
A31	A32	A33

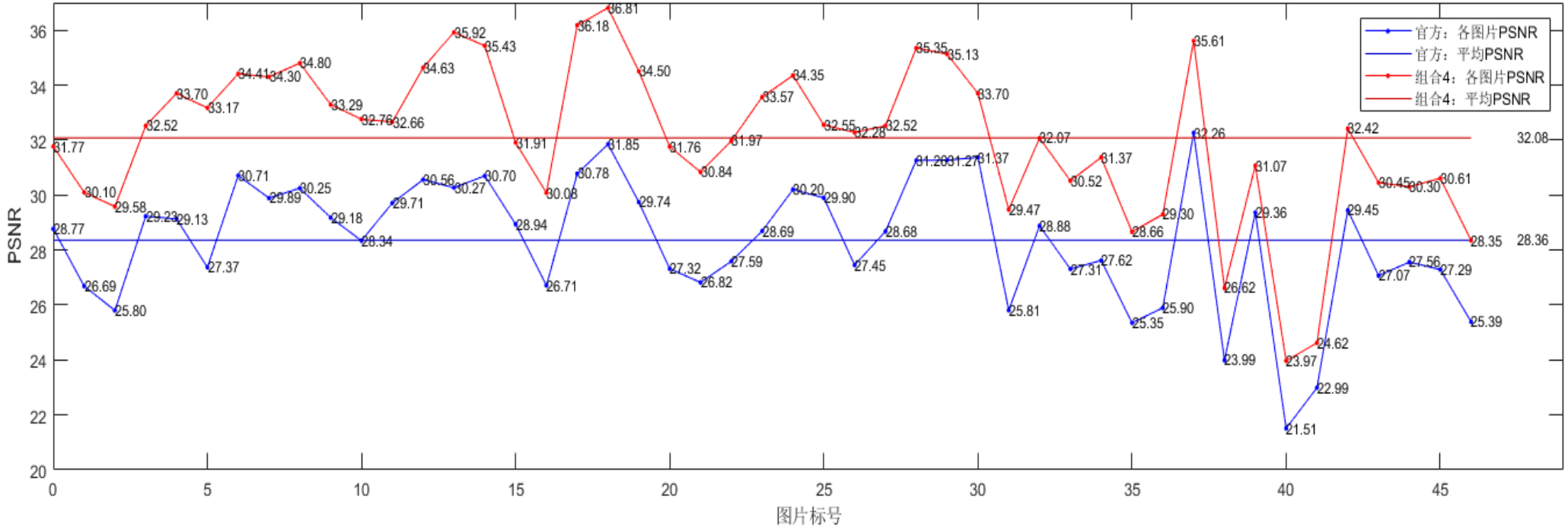
$$\times \frac{1}{16} \times \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \Longrightarrow$$

	A22’	

各种组合算法及其性能

✦组合4：边缘插值+均值调整+高斯平滑+均值调整

PSNR

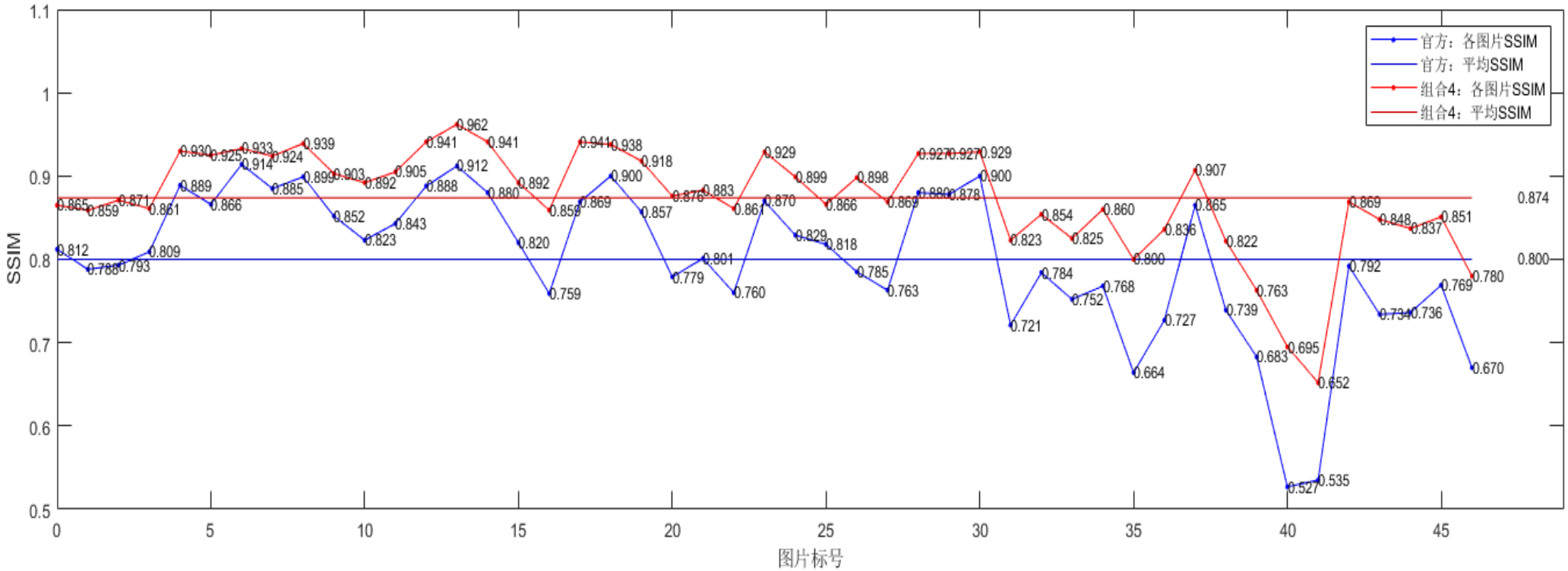


组合4平均：PSNR=32.08　参考平均：PSNR=28.36

各种组合算法及其性能

✦组合4：边缘插值+均值调整+高斯平滑+均值调整

SSIM

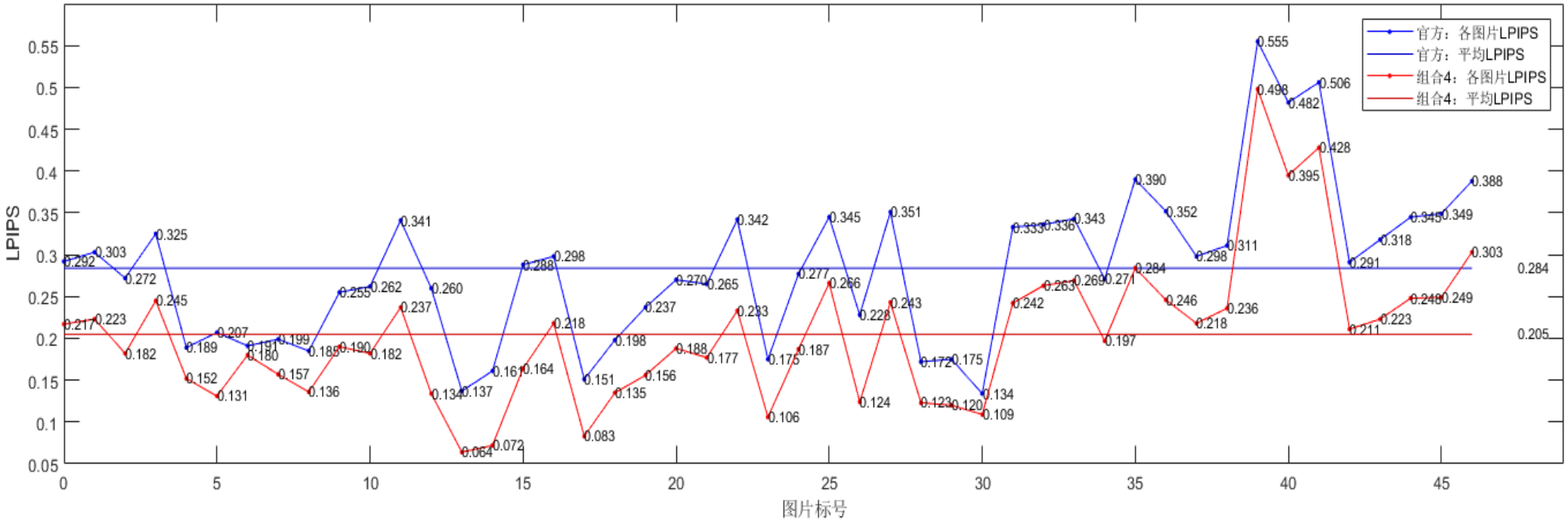


组合4平均：SSIM=0.874　　参考平均：SSIM=0.800

各种组合算法及其性能

✦组合4：边缘插值+均值调整+高斯平滑+均值调整

LPIPS



组合4平均：LPIPS=0.205　参考平均：LPIPS=0.284

各种组合算法及其性能

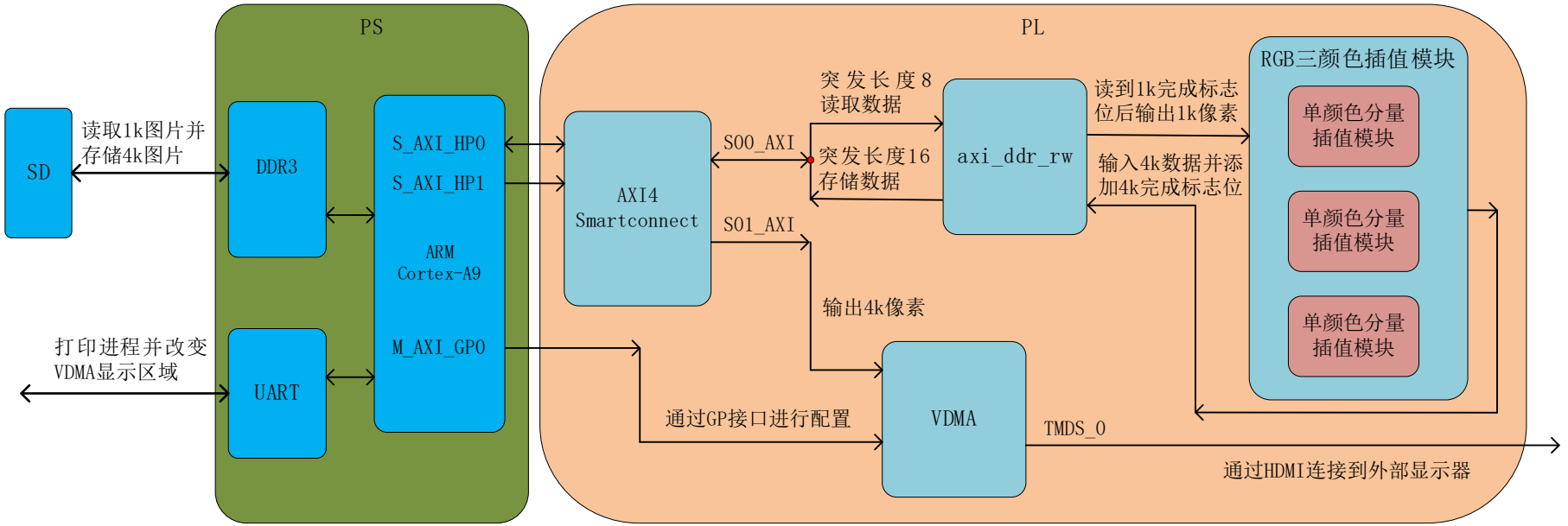
组合1：新型双线性插值+均值调整
组合2：新型双线性插值+均值调整+高斯平滑+均值调整
组合3：分区域插值+均值调整+高斯平滑+均值调整
组合4：边缘插值+均值调整+高斯平滑+均值调整
组合5：边缘插值+均值调整+高斯平滑

	平均PSNR	平均SSIM	平均LPIPS
官方参考	28.36	0.800	0.284
组合1	31.73	0.871	0.212
组合2	31.88	0.873	0.223
组合3	32.11	0.875	0.211
✦组合4	32.08	0.874	0.205
组合5	31.97	0.870	0.224

硬件架构

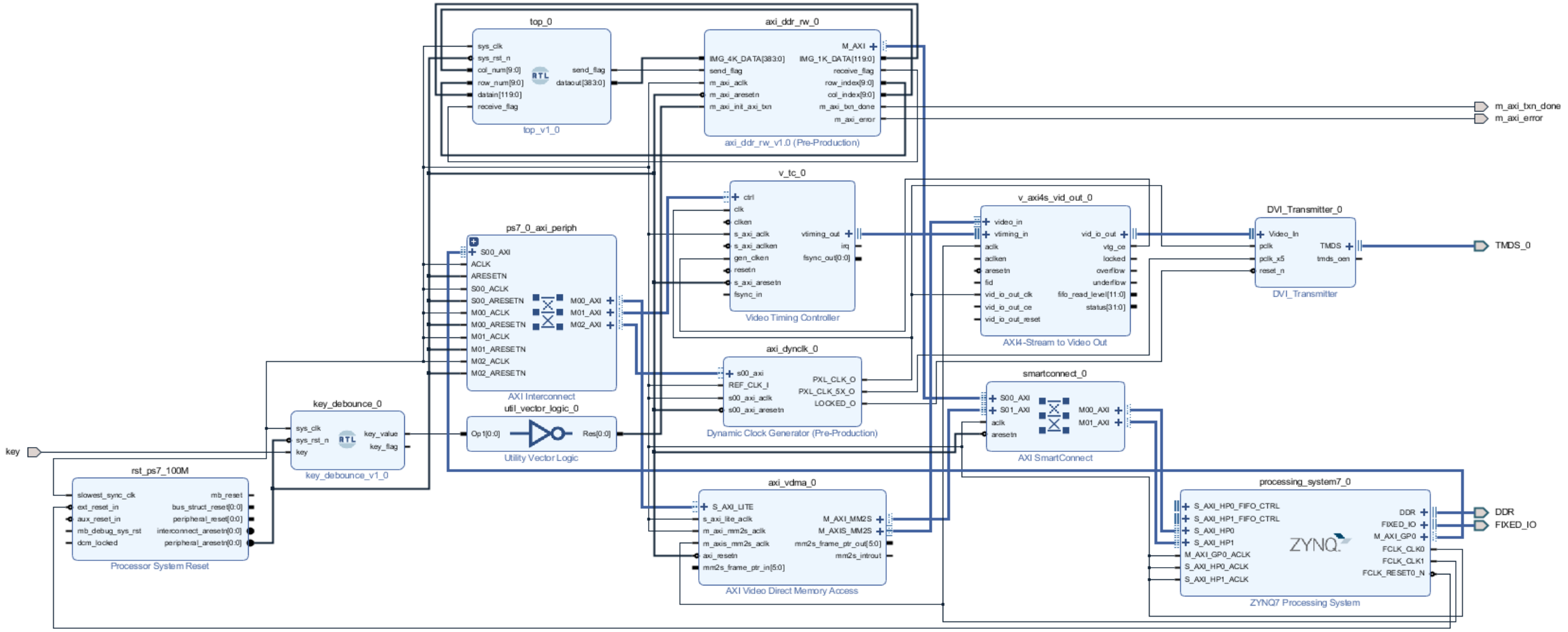
系统硬件架构

我们采用组合4作为最终硬件实现的算法，即边缘插值+均值调整+高斯平滑+均值调整。



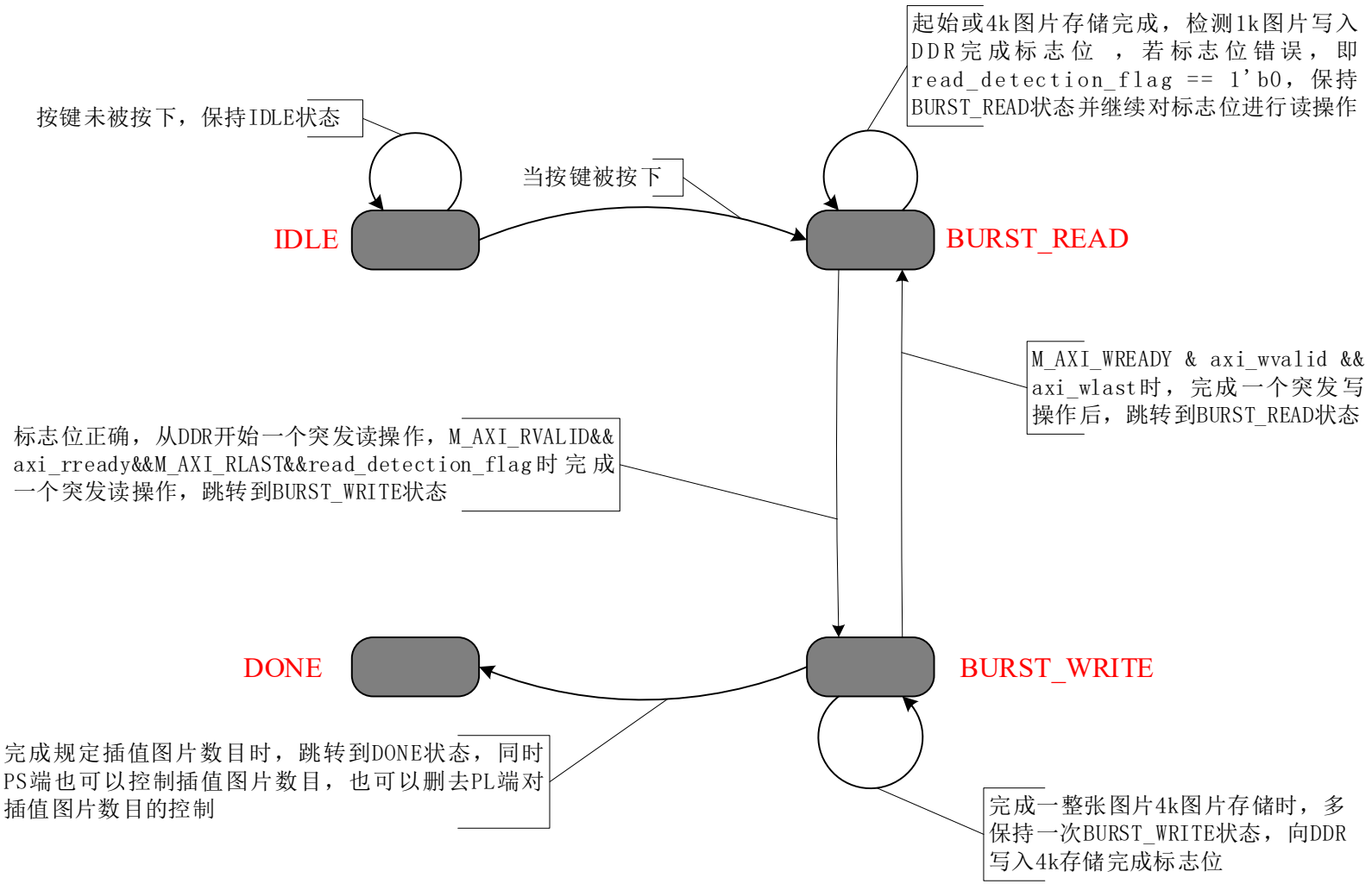
算法硬件框架图

PL端硬件架构



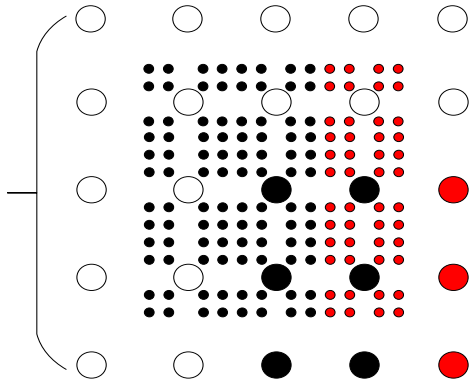
系统Block Design框图

PL端硬件架构



axi_ddr_rw 状态机

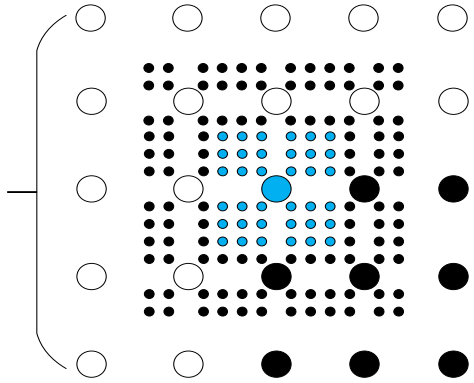
单颜色分量插值模块



插值流程示意图1

大圆点表示1k像素点，小圆点表示4k像素点，空心圆点代表扩展像素点

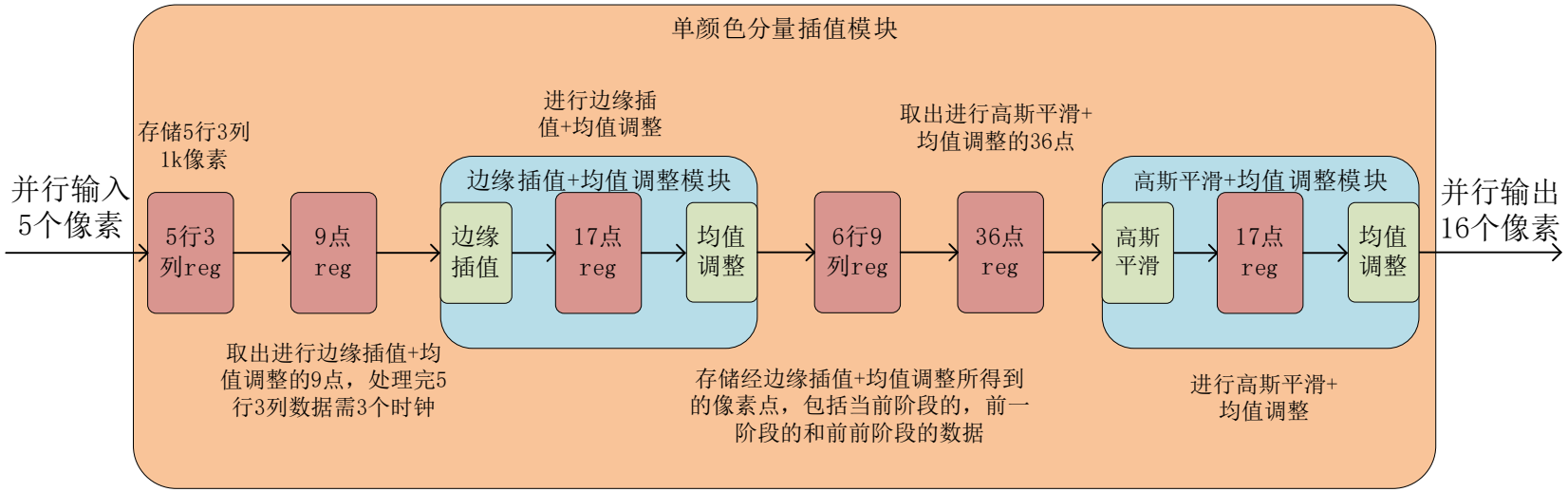
当接收到1k图片一列的5个像素点时，即图中的红色大圆点以及扩展的大圆点，利用原先存储的1k像素点，可对前一列的中间3个1k像素点进行边缘插值+均值调整，得到图中的红色小圆点。



插值流程示意图2

对于图中蓝色大圆点，我们已经求得其周边36个4k像素点，即图中的蓝色小圆点，此时可以对这部分像素点进行高斯平滑+均值调整，得到该1k像素点最终对应的4k图片的16个像素点。

单颜色分量插值模块



单颜色分量插值模块框架

3个时钟处理一个1k像素点，100M的时钟下，理想状态每秒能处理约64张图片，但由于只能对DDR进行单个像素存取，且AXI传输并非一直有效，实际仅能处理约3张图片。

资源消耗

Resource	Utilization	Available	Utilization %
LUT	11220	53200	21.09
FF	3586	106400	3.37

顶层模块资源消耗

	LUT资源消耗	使用率
顶层模块	11220	21.09%
新型双线性插值模块	837	1.57%
边缘插值模块	1815	3.41%
均值调整模块	605	1.14%
高斯平滑模块	40	0.08%

各模块LUT资源消耗

Name	Slice LUTs (53200)	Slice Registers (106400)	Bonded IOB (125)	BUFGCTRL (32)
top	11220	3586	516	1
single_color_top_b (single_color_top)	3048	1199	0	0
gauss_filter_dot_36_4k_inst (gauss_filt...	848	136	0	0
interpolate_16point_inst (interpolate_16...	1663	136	0	0
single_color_top_g (single_color_top_0)	2592	1192	0	0
gauss_filter_dot_36_4k_inst (gauss_filt...	848	136	0	0
interpolate_16point_inst (interpolate_16...	1663	136	0	0
single_color_top_r (single_color_top_1)	2592	1195	0	0
gauss_filter_dot_36_4k_inst (gauss_filt...	848	136	0	0
interpolate_16point_inst (interpolate_16...	1663	136	0	0

顶层模块资源消耗分布

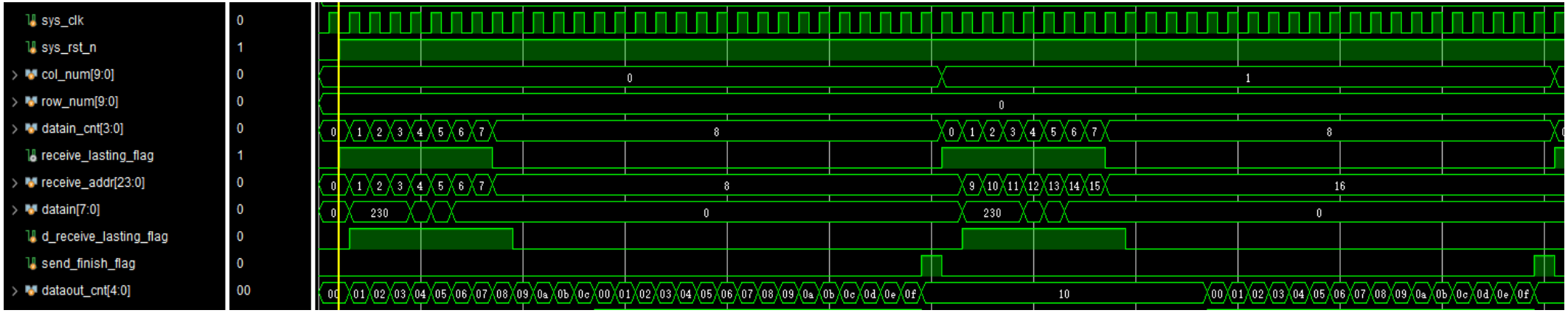
仿真测试

多线程

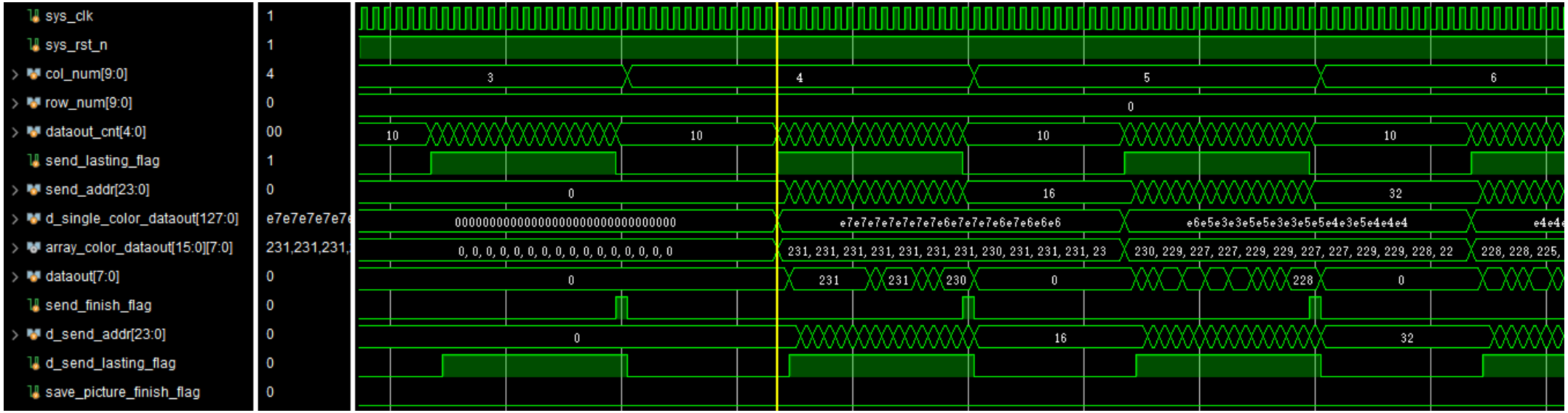
我们在VMware Workstation 16Pro的ubuntu 20.04中编写c代码，完成单线程软件仿真，然后我们分离RGB三个颜色分量，进行三线程仿真运行。我们将生成的单线程与多线程4k图片导入MATLAB，然后进行一一比对，结果表明，两者数据一致，以下为23张图片插值单线程与三线程总的运行时间：

23张1k图片单线程运行总时间	23张1k图片三线程运行总时间
2m53.696s	1m6.773s

硬件仿真波形图

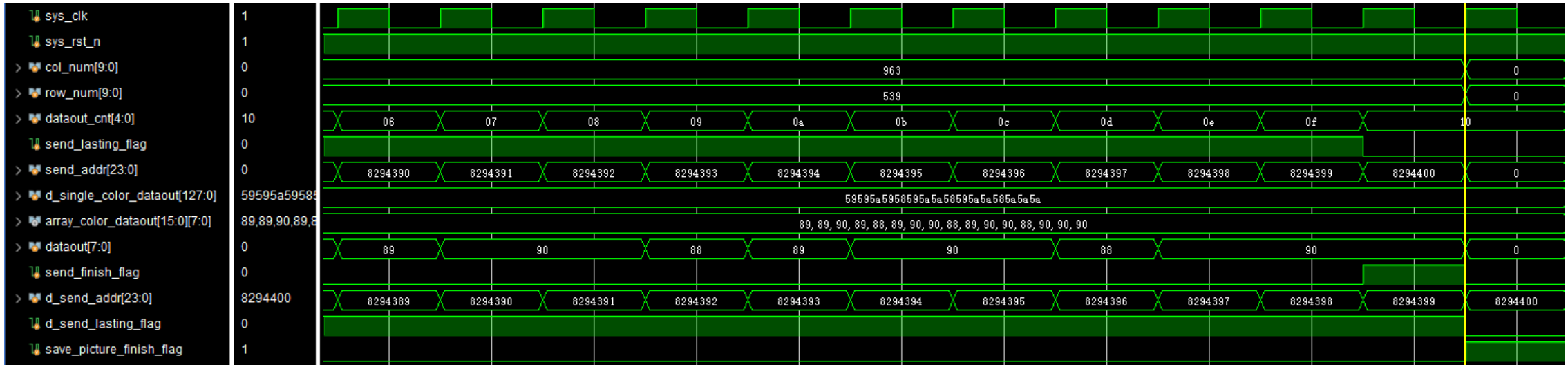


开始读取1k数据波形图



开始存储4k数据波形图

硬件仿真波形图



1k图片插值完成波形图

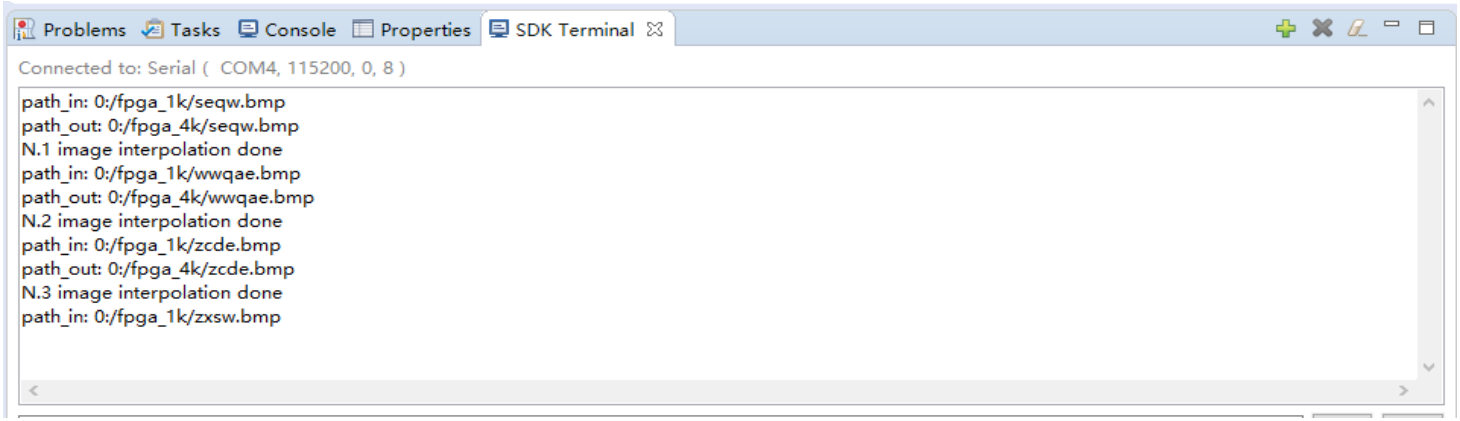
当行数和列数到达最大值，且send_finish_flag拉高，此时save_picture_flag拉高一个时钟周期，4k图片存储完成，发送地址d_send_addr最终的值为8294399，即共存储了8294400个像素数据，与4k图片行数与列数之积3840x2160=8294400相匹配。

由于我们仿真生成了一整张4k图片，而且仿真结果与软件一致，这表明我们的整个插值模块是正确的，且仿真的代码覆盖率为100%。

FPGA验证

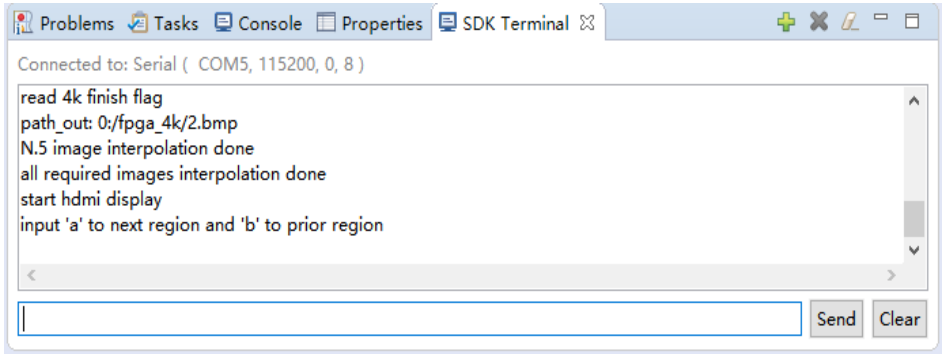
FPGA验证截图

我们采用组合4：边缘插值+均值调整+高斯平滑+均值调整的方案作为最终硬件实现，基于此，我们在Xilinx Zynq-7020 FPGA开发板上搭建整个系统。
针对1k图像集的图片，我们首先通过软件仿真得到对应的上采样4k图片，再将1k图像集存入SD卡中，通过FPGA对其进行上采样，同样相应的上采样4k图片，最后将软件与硬件得到的对应4k图片导入MATLAB进行数据对比，结果一致，表明软硬件对比一致，FPGA验证通过。



串口打印进程信息

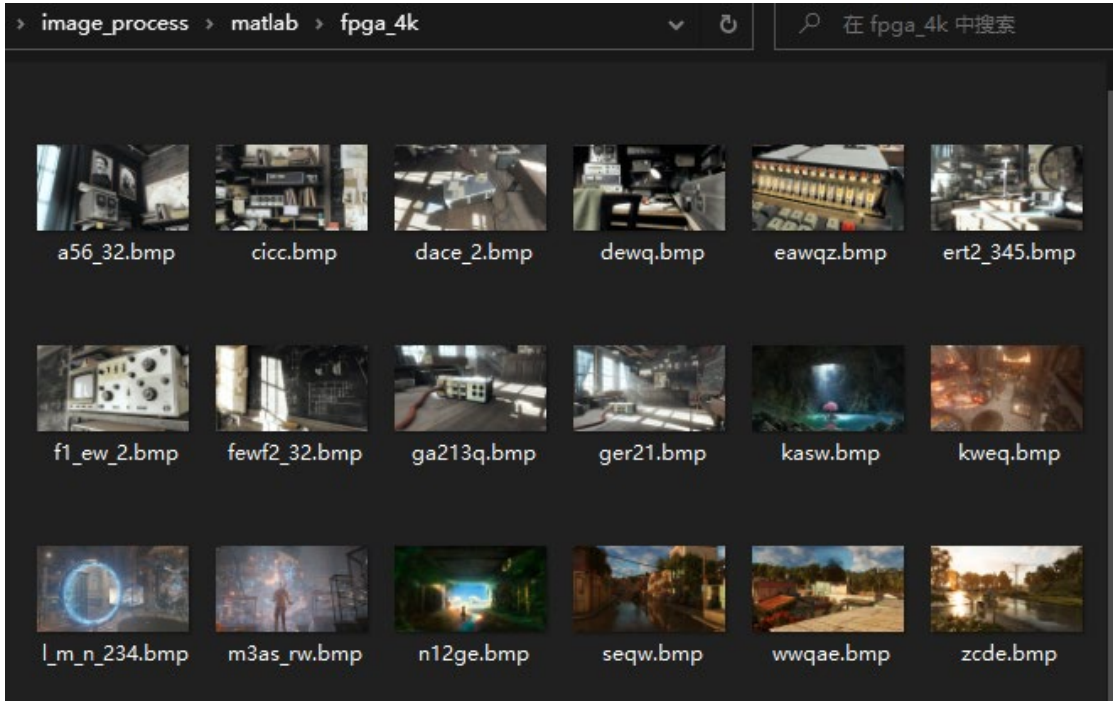
FPGA验证截图



插值完成开始hdmi显示

分时显示4k图片

FPGA验证截图



FPGA生成4k图片截图

景嘉微杯

恳请批评与指正！

队伍编号：CICC1837

团队名称：啊对对对对对对

团队成员：朱涛、徐振华、李子谅