实验平台： Zedboard(XC7Z020-CLG484-1)
开发工具： Vivado2019.2开发套件
（由于jpg、png等图片格式涉及图像压缩算法，因此在zynq平台读取此类图片较为困难，故目前仅支持bmp图片的读取）

bmp图片格式介绍

BMP(全称Bitmap)是Window操作系统中的标准图像文件格式，可以分成两类：设备相关位图(DDB)和设备无关位图(DIB)，使用非常广。它采用位映射存储格式，除了图像深度可选以外，不采用其他任何压缩，因此，BMP文件所占用的空间很大。BMP文件的图像深度可选1bit、4bit、8bit及24bit。BMP文件存储数据时，图像的扫描方式是按**从左到右、从下到上**的顺序。 由于BMP文件格式是Windows环境中交换与图有关的数据的一种标准，因此在Windows环境中运行的图形图像软件都支持BMP图像格式。
BMP文件格式由文件头和原始位图数据Raw Bitmap Data构成（详见https://en.wikipedia.org/wiki/BMP_file_format）。

文件头

位图格式的文件头长度可变，而且其中参数繁多。但通常情况下bmp格式图片的文件头长度均为**54字节**，其中包括**14字节的Bitmap文件头**以及**40字节的DIB**（**Device Independent Bitmap）数据头，或称位图信息数据头**（**BItmap Information Header**）。

1. 位图文件头 Bitmap File Header (14 bytes) image

2. 位图信息数据头 DIB Header (54 bytes)

Bitmap Information Header

位置（hex/dec）		尺寸（byte）	描述
0E	14	4	DIB header 的大小 通常为 40 bytes 即 0x28
12	18	4	图像宽度（little endian）
16	22	4	图像高度（little endian）
1A	26	2	色彩平面（color plane）的数量 必须为 1
1C	28	2	每像素用多少 bit 来表示
1E	30	4	采用何种压缩方式 通常不压缩，即 BI_RGB，对应值为 0
22	34	4	图片大小（原始位图数据的大小） 对于不压缩的图片，通常表示为 0
26	38	4	横向分辨率（像素/米）
2A	42	4	纵向分辨率（像素/米）
2E	46	4	调色板中颜色数量 通常为 0（不表示没有颜色）
32	50	4	重要颜色的数量（通常被忽略） 通常为 0，表示每种颜色都重要

原始位图数据Raw Bitmap Data

图像数据块从文件头中起始偏移量字段所指出的位置开始，其中存放着位图图像的数据，数据格式由图像参数信息块中的压缩方式选项的取值决定。操作图像数据块时，需要注意： 当压缩方式为RGB时，图像数据块**以“行”为单位双字（4字节）对齐**，如1行像素的字节数不为4的倍数，则进行填充，直至4字节对齐为止（一般填充0）。

HLS Video Library

参考资料：**UG902 (v2018.2) July 2, 2018**
HLS视频库中包含的视频处理函数与现有的OpenCV功能兼容，名称类似。它们不会直接取代现有的OpenCV视频库函数。视频处理函数使用一种数据类型**hls::Mat**。这种数据类型允许将函数作为高性能硬件进行综合和实现。
在HLS视频库中提供了三种类型的函数：

- OpenCV Interface Functions
  实现在AXI4流式数据类型和标准的OpenCV数据类型之间的转换。
- AXI Interface Functions
  实现在视频数据和hls::Mat数据类型之间的转换。
- Video Processing Functions
  兼容标准的OpenCV函数，以操作和处理视频图像。这些函数使用hls::mat数据类型，并可由Vivado HLS

综合。Vivado HLS视频库中包含的视频处理函数是专门用于操作视频图像的。这些函数大多是为了加速相应的OpenCV函数而设计的。

下表总结了HLS视频库中提供的函数：

Function Type	Function	Description
OpenCV Interface	AXIvideo2cvMat	Converts data from AXI4 video stream (hls::stream) format to OpenCV cv::Mat format
OpenCV Interface	AXIvideo2CvMat	Converts data from AXI4 video stream (hls::stream) format to OpenCV CvMat format2
OpenCV Interface	AXIvideo2IplImage	Converts data from AXI4 video stream (hls::stream) format to OpenCV IplImage format
OpenCV Interface	cvMat2AXIvideo	Converts data from OpenCV cv::Mat format to AXI4 video stream (hls::stream) format
OpenCV Interface	CvMat2AXIvideo	Converts data from OpenCV CvMat format to AXI4 video stream (hls::stream) format
OpenCV Interface	cvMat2hlsMat	Converts data from OpenCV cv::Mat format to hls::Mat format
OpenCV Interface	CvMat2hlsMat	Converts data from OpenCV CvMat format to hls::Mat format
OpenCV Interface	CvMat2hlsWindow	Converts data from OpenCV CvMat format to hls::Window format
OpenCV Interface	hlsMat2cvMat	Converts data from hls::Mat format to OpenCV cv::Mat format
OpenCV Interface	hlsMat2CvMat	Converts data from hls::Mat format to OpenCV CvMat format
OpenCV Interface	hlsMat2IplImage	Converts data from hls::Mat format to OpenCV IplImage format
OpenCV Interface	hlsWindow2CvMat	Converts data from hls::Window format to OpenCV CvMat format
OpenCV Interface	IplImage2AXIvideo	Converts data from OpenCV IplImage format to AXI4 video stream (hls::stream) format
OpenCV Interface	IplImage2hlsMat	Converts data from OpenCV IplImage format to hls::Mat format
AXI4-Interface	AXIvideo2Mat	Converts image data stored in hls::Mat format to an AXI4 video stream (hls::stream) format
AXI4-Interface	Mat2AXIvideo	Converts image data stored in AXI4 video stream (hls::stream) format to an image of hls::Mat format

Function Type	Function	Description
AXI4-Interface	Array2Mat	Converts image data stored in an array to an image of hls::Mat format.
AXI4-Interface	Array2Mat	Converts image data stored hls::Mat format to an array.
Video Processing	AbsDiff	Computes the absolute difference between two input images src1 and src2 and saves the result in dst
Video Processing	AddS	Computes the per-element sum of an image src and a scalar scl
Video Processing	AddWeighted	Computes the weighted per-element sum of two image src1 and src2
Video Processing	And	Calculates the per-element bitwise logical conjunction of two images src1 and src2
Video Processing	Avg	Calculates an average of elements in image src
Video Processing	AvgSdv	Calculates an average of elements in image src
Video Processing	Cmp	Performs the per-element comparison of two input images src1 and src2
Video Processing	CmpS	Performs the comparison between the elements of input images src and the input value and saves the result in dst
Video Processing	CornerHarris	This function implements a Harris edge/corner detector
Video Processing	CvtColor	Converts a color image from or to a grayscale image
Video Processing	Dilate	Dilates the image src using the specified structuring element constructed within the kernel
Video Processing	Duplicate	Copies the input image src to two output images dst1 and dst2, for divergent point of two datapaths
Video Processing	EqualizeHist	Computes a histogram of each frame and uses it to normalize the range of the following frame
Video Processing	Erode	Erodes the image src using the specified structuring element constructed within kernel
Video Processing	FASTX	Implements the FAST corner detector, generating either a mask of corners, or an array of coordinates

Function Type	Function	Description
Video Processing	Filter2D	Applies an arbitrary linear filter to the image src using the specified kernel
Video Processing	GaussianBlur	Applies a normalized 2D Gaussian Blur filter to the input
Video Processing	Harris	This function implements a Harris edge or corner detector
Video Processing	HoughLines2	Implements the Hough line transform
Video Processing	Integral	Implements the computation of an integral image
Video Processing	InitUndistortRectifyMap	Generates map1 and map2, based on a set of parameters, where map1 and map2 are suitable inputs for hls::Remap()
Video Processing	Max	Calculates per-element maximum of two input images src1 and src2 and saves the result in dst
Video Processing	MaxS	Calculates the maximum between the elements of input images src and the input value and saves the result in dst
Video Processing	Mean	Calculates an average of elements in image src, and return the value of first channel of result scalar
Video Processing	Merge	Composes a multichannel image dst from several single-channel images
Video Processing	Min	Calculates per-element minimum of two input images src1 and src2 and saves the result in dst
Video Processing	MinMaxLoc	Finds the global minimum and maximum and their locations in input image src
Video Processing	MinS	Calculates the minimum between the elements of input images src and the input value and saves the result in dst
Video Processing	Mul	Calculates the per-element product of two input images src1 and src2
Video Processing	Not	Performs per-element bitwise inversion of image src
Video Processing	PaintMask	Each pixel of the destination image is either set to color (if mask is not zero) or the corresponding pixel from the input image
Video Processing	PyrDown	Blurs the image and then reduces the size by a factor of 2.

Function Type	Function	Description
Video Processing	PyrUp	Upsamples the image by a factor of 2 and then blurs the image.
Video Processing	Range	Sets all value in image src by the following rule and return the result as image dst:dst(I) = (end-start)x(ixdst.cols+j)/(dst.rows*dst.cols)
Video Processing	Remap	Remaps the source image src to the destination image dst according to the given remapping
Video Processing	Reduce	Reduces 2D image src along dimension dim to a vector dst
Video Processing	Resize	Resizes the input image to the size of the output image using bilinear interpolation
Video Processing	Set	Sets elements in image src to a given scalar value scl
Video Processing	Scale	Converts an input image src with optional linear transformation
Video Processing	Sobel	Computes a horizontal or vertical Sobel filter, returning an estimate of the horizontal or vertical derivative.
Video Processing	Split	Divides a multichannel image src to several single-channel images
Video Processing	SubRS	Computes the differences between scalar value scl and elements of image src
Video Processing	SubS	Computes the differences between elements of image src and scalar value scl
Video Processing	Sum	Sums the elements of an image
Video Processing	Threshold	Performs a fixed-level threshold to each element in a single-channel image
Video Processing	Zero	Sets elements in image src to 0

Demo流程

HLS开发

1. 创建HLS工程，选择板子型号为XC7Z020-CLG484-1，添加hls文件夹内的源文件及测试平台文件，然后**依次点击Project、Project Settings...、Synthesis、Browse...，选择sobel作为顶层函数。**

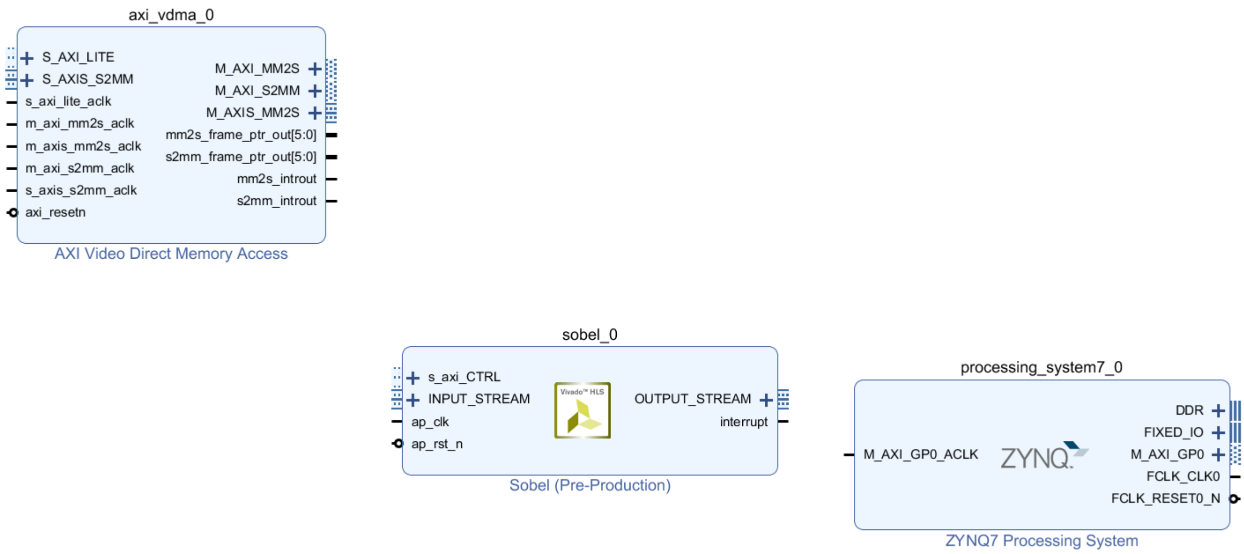
2. 修改宏INPUT_IMAGE和宏OUTPUT_IMAGE，分别表示待处理图片的文件路径以及图片输出路径。
3. 点击Run C Simulation，进行C仿真。
4. 点击Run C Synthesis，进行C综合。
5. 点击Run C/RTL Cosimulation，进行C/RTL协同仿真。
6. 点击Export RTL as IP，将综合后的代码导出为IP。

Block Design

1. 创建Vivado工程，选择板子型号为XC7Z020-CLG484-1。

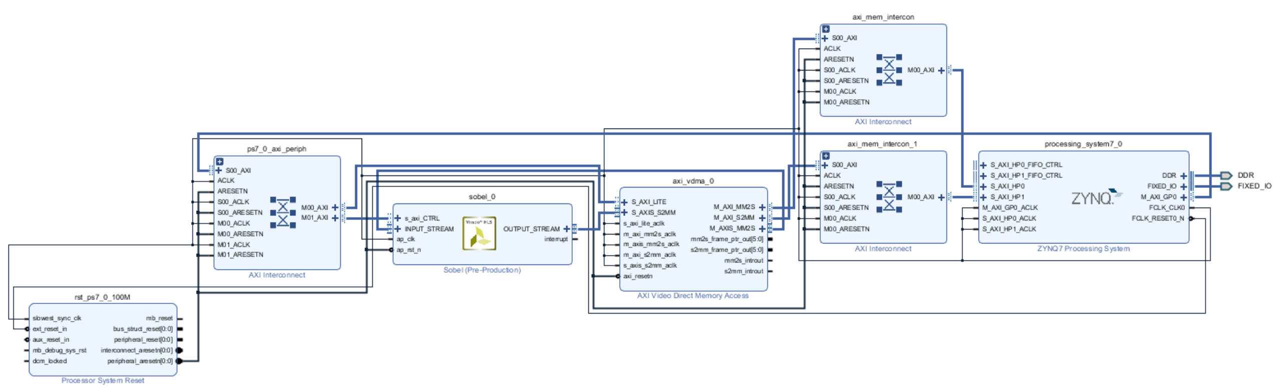
2. 点击IP Catalog，添加HLS导出IP的路径。

3. 点击Create Block Design，分别添加HLS导出的IP（名为sobel）、AXI VDMA IP核以及ZYNQ IP核。如下图所示：



4. 配置IP核
   sobel ip：无需配置。
   axi vdma ip：将Frame Buffers设置为1，其余保持默认即可。
   zynq ip：包括勾选1个AXI GP接口，2个AXI HP接口，设置PL端时钟频率，勾选UART和SD，配置DDR型号（因开发板型号不同而不同）。

5. 手动连接AXI STREAM接口，其余自动连线即可，连线完成后，进行Validate Design，最终的硬件系统如下所示



6. 右键block design文件，点击Create HDL Wrapper。

7. 点击Generate Bitstream，进行综合、布局布线、生成比特流。

8. File-->Export-->Export Hardware，勾选Include bitstream，然后点击OK。

9. Tool-->Launch Vitis。

SDK开发

1. Launch Vitis后，选择Workspace路径，点击Launch。
2. **创建平台**
   Create Platform Project，输入Project Name， next， next，点击Browse选择vivado导出的.xsa硬件描述文件，最后点击Finish。
3. 选择zynq_fsbl下的Board Support Package，点击Modify BSP Settings，勾选xilffs，并将xilffs设置中的use_lfn的修改为1，以支持长文件名的读取。对standalone on ps7_cortexa9_0下的Board Support Package进行相同的操作。
4. **创建SDK工程**
   点击File-->New-->Application Project，填写Project name， next， next， next，选择Empty Application，点击Finish。
5. 将sdk文件夹下的所有文件拷贝至创建工程的src文件夹下。
6. 双击lscript.ld，修改堆栈大小。如将Stack Size和Heap Size均调节为0x2000000。
7. 右键工程，点击Build Project。连接开发板，打开串口调试助手（波特率选择115200）。将bmp图片存入SD卡，插到FPGA开发板上。
8. **烧写程序**。右键工程，点击Run As-->Run Configuartions，双击Single Application Debug，点击Run。
9. 可通过修改const char* input_image和const char* output_image来改变待测试图片和输出图片的文件名。
10. 运行完成后，拔出SD卡，在PC上通过读卡器查看结果。

更换为其它HLS Open-CV处理函数

在hls源文件imgProc.cpp中，共包含6个图像处理的示例函数，分别为

```
void sobel(AXI_STREAM& INPUT_STREAM,AXI_STREAM& OUTPUT_STREAM,int rows,int cols);

void rgb2gray(AXI_STREAM& INPUT_STREAM,AXI_STREAM& OUTPUT_STREAM,int rows,int
cols);

void gaussian(AXI_STREAM& INPUT_STREAM,AXI_STREAM& OUTPUT_STREAM,int rows,int
cols);

void binary(AXI_STREAM& INPUT_STREAM,AXI_STREAM& OUTPUT_STREAM,int rows,int cols);

void dilate_erode(AXI_STREAM& INPUT_STREAM,AXI_STREAM& OUTPUT_STREAM,int rows,int
cols);

void fast_corner(AXI_STREAM & INPUT_STREAM,AXI_STREAM & OUTPUT_STREAM,int rows,int
cols);
```

可以看到，这6个函数的参数是一致的：其中，INPUT_STREAM表示通过AXI STREAM接口输入的待处理图像，OUTPUT_STREAM表示通过AXI STREAM接口输出的已处理图像，rows、cols分别表示待处理图像的高、宽。为了测试其它HLS Open-CV处理函数，仅需修改Demo中的：

- HLS开发中的第1步：选择顶层函数，例如fast_corner。
- SDK开发中的第5步：将main.c中的第11、80、110-113以及122行中的IP核名称修改为当前IP核的名字。

```
//line11
#include "xsobel.h"--->#include "fast_corner.h"
//line80
XSobel hls_inst;--->XFast_corner hls_inst;
//line 110-113
XSobel_Initialize(&hls_inst,0)--->XFast_corner_Initialize(&hls_inst,0);
XSobel_Set_rows(&hls_inst, (u32)h);--->XFast_corner_Set_rows(&hls_inst,(u32)h);
XSobel_Set_cols(&hls_inst, (u32)w);--->XFast_corner_Set_cols(&hls_inst, (u32)w);
XSobel_Start(&hls_inst);--->XFast_corner_Start(&hls_inst);
//line122
while(XSobel_IsDone(&hls_inst));--->while(XFast_corner_IsDone(&hls_inst));
```

由于在绝大多数情况下，HLS编写的图像处理IP核均具有上述形式，因此，仅需修改HLS代码，以及HLS开发中第1步、SDK开发中第5步，即可实现功能的快速切换。