MSOC

Oct. 30, 2020

A HLS Solution for Edge Detection in Driver Assistance Application

Presenter: Ting-Yung Chen, Yu-Cheng Lin, I-Hsuan Liu

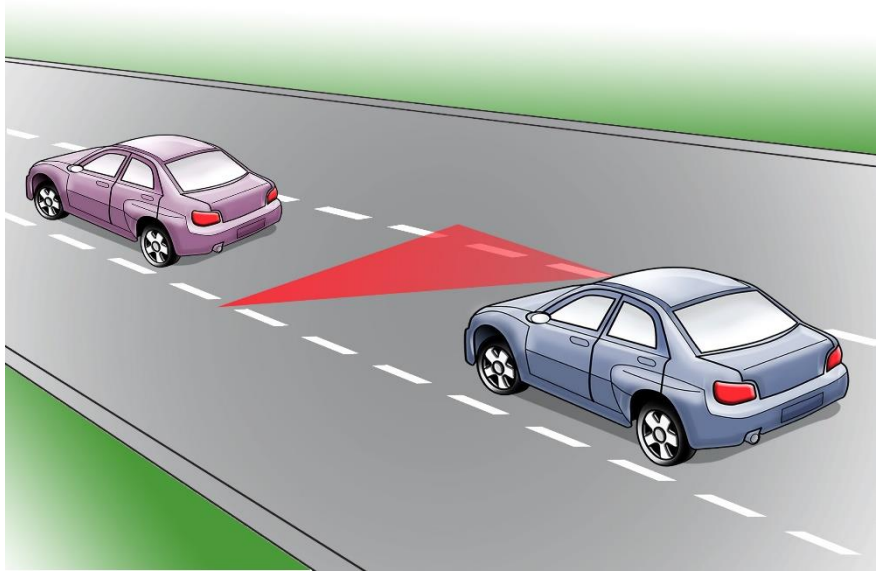
Team#: 1

Outline

- Introduction
- Design Overview
  - Our Solution for Edge Detection
  - Synth. repo
- FPGA implementation
  - Video Demo on FPGA
  - Host Program

Introduction

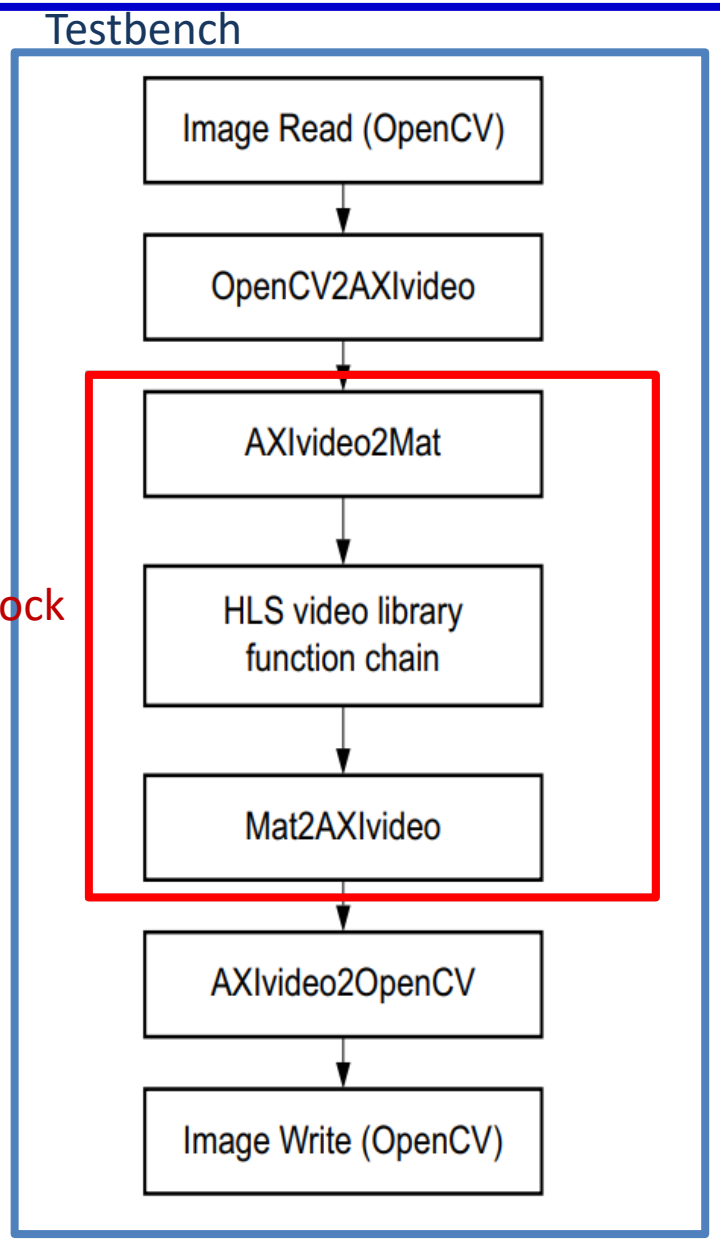
- Driver Assistance Application
  - Edge detection
  - Lane Keeping Aid (LKA)
- Hardware accelerator
  - Real time



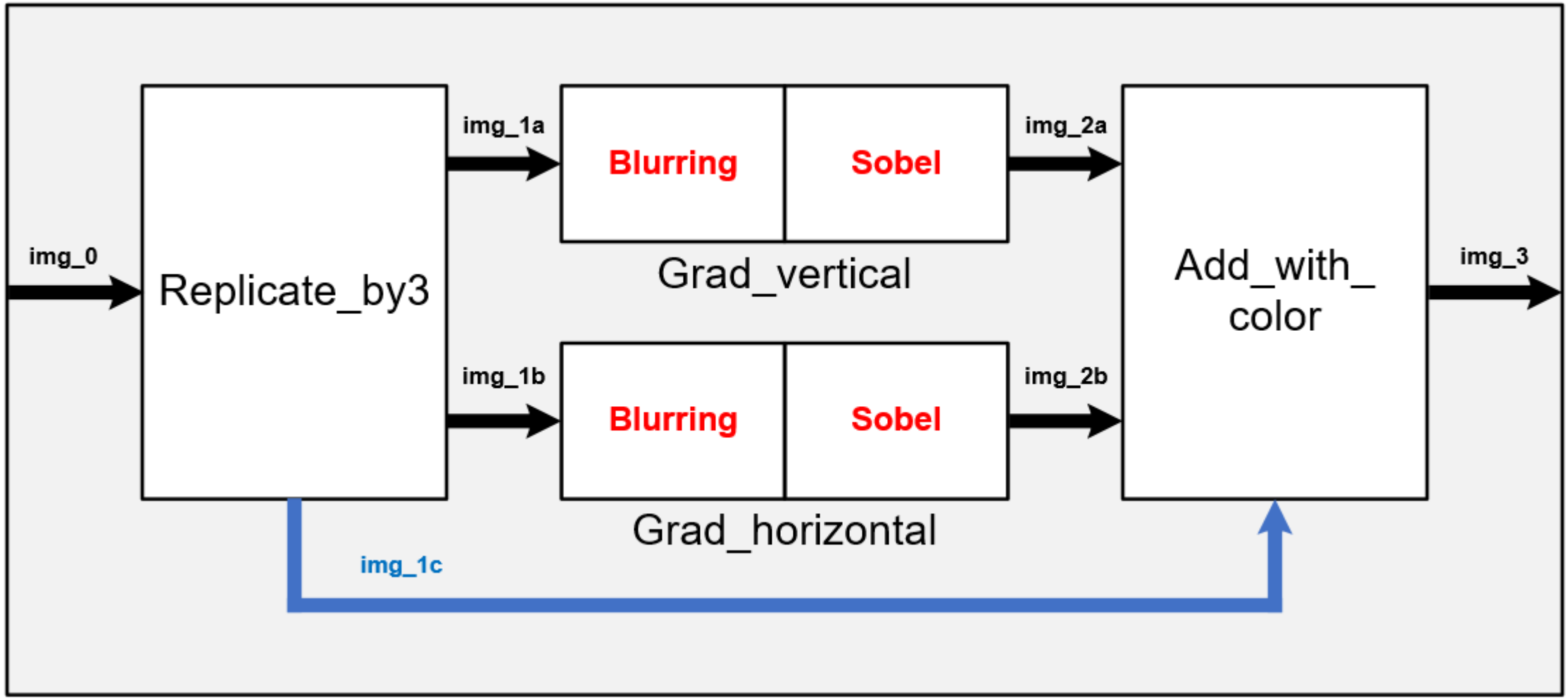
Design Overview

- OpenCV-based testbenches (non-synthesizable)
  - Read/Write input frames with OpenCV application
  - OpenCV2AXIvideo, AXIvideo2OpenCV
  - hls::IplImage2AXIvideo, hls::AXIvideo2IplImage

- Synthesizable block
  - AXIvideo2Mat, Mat2AXIvideo
  - hls::AXIvideo2Mat, hls::Mat2AXIvideo



Our Solution for Edge Detection



Blur and Filter Function Design

- Blur function

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

- Sobel filter

$$\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix} \qquad \begin{bmatrix}+1 & +2 & +1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$

Decision Boundary

```
if ((row <= rows-10 && col <= cols-10)&&(row >= 120) && ( (pin0.val[0]>80 )  && (pin0.val[1]>80 )
&& (pin0.val[2]>80 ))&& ( (pin0.val[0]<240 )   && (pin0.val[1]<240 )  && (pin0.val[2]<240 ))) { // vertical edges
  pout.val[0] = 0;    // B
  pout.val[1] = 0;  // G
  pout.val[2] = 255;    // R
} else if ((row <= rows-10 && col <= cols-10)&&(row >= 120) && ((pin1.val[0]>255 )  && (pin1.val[1]>255 )
&& (pin1.val[2]>255) )) { // horizontal edges
  pout.val[0] = 0;
  pout.val[1] = 0;
  pout.val[2] = 255;
} else {
  //pout.val[0] = 0; //for showing edge dection result only
  //pout.val[1] = 0;
  //pout.val[2] = 0;
  pout.val[0] = pin2.val[0];
  pout.val[1] = pin2.val[1];
  pout.val[2] = pin2.val[2];
}
```

Filters

```
const COEF_T coef_v1[KS][KS]= { //p
    {1.0/9, 1.0/9, 1.0/9},
    {1.0/9, 1.0/9, 1.0/9},
    {1.0/9, 1.0/9, 1.0/9}
};
const COEF_T coef_v2[KS][KS] = { //
    {1,0,-1},
        {2,0,-2},
            {1,0,-1}
};
```

Synth. Report

FPS:1666 (220*220)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	6.00 ns	5.250 ns	0.75 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
100573	100573	0.603 ms	0.603 ms	100561	100561	dataflow

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	
grad_vertical_prepro_U0	grad_vertical_prepro	100560	100560	0.603 ms	0.603 ms	100560	100560	
grad_horizontal_prep_U0	grad_horizontal_prep	100560	100560	0.603 ms	0.603 ms	100560	100560	
grad_vertical_edge_d_U0	grad_vertical_edge_d	100560	100560	0.603 ms	0.603 ms	100560	100560	
grad_horizontal_edge_U0	grad_horizontal_edge	100560	100560	0.603 ms	0.603 ms	100560	100560	
add_with_color_U0	add_with_color	48400	48401	0.290 ms	0.290 ms	48400	48400	loop
AXIvideo2Mat_U0	AXIvideo2Mat	49503	49503	0.297 ms	0.297 ms	49503	49503	
Mat2AXIvideo_U0	Mat2AXIvideo	49281	49281	0.296 ms	0.296 ms	49281	49281	
replicate_by2_U0	replicate_by2	48399	48400	0.290 ms	0.290 ms	48399	48400	

Loop

N/A

FPS:40 (1920*1080)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	6.00 ns	5.250 ns	0.75 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
4166191	4166191	24.997 ms	24.997 ms	4166180	4166180	dataflow

Detail

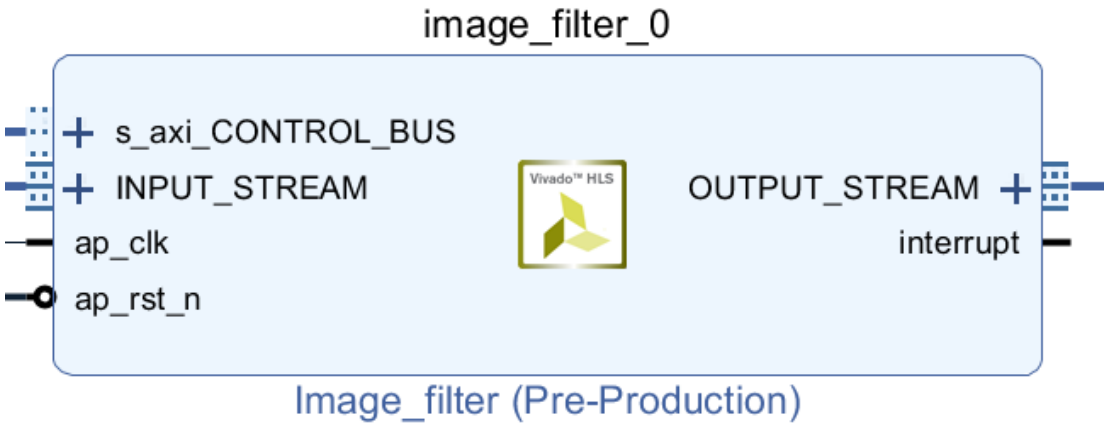
Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)	
Instance	Module	min	max	min	max	min	max
grad_vertical_prepro_U0	grad_vertical_prepro	4166179	4166179	24.997 ms	24.997 ms	4166179	4166179
grad_horizontal_prep_U0	grad_horizontal_prep	4166179	4166179	24.997 ms	24.997 ms	4166179	4166179
grad_vertical_Mat_U0	grad_vertical_Mat_s	4166179	4166179	24.997 ms	24.997 ms	4166179	4166179
grad_horizontal_U0	grad_horizontal	4166179	4166179	24.997 ms	24.997 ms	4166179	4166179
add_with_color96_U0	add_with_color96	2073600	2073601	12.442 ms	12.442 ms	2073600	2073600
AXIvideo2Mat_U0	AXIvideo2Mat	2079003	2079003	12.474 ms	12.474 ms	2079003	2079003
Mat2AXIvideo_U0	Mat2AXIvideo	2077921	2077921	12.468 ms	12.468 ms	2077921	2077921
replicate_by293_U0	replicate_by293	2073599	2073600	12.442 ms	12.442 ms	2073599	2073600
passthru_color94_U0	passthru_color94	2073599	2073600	12.442 ms	12.442 ms	2073599	2073600
passthru_color95_U0	passthru_color95	2073599	2073600	12.442 ms	12.442 ms	2073599	2073600

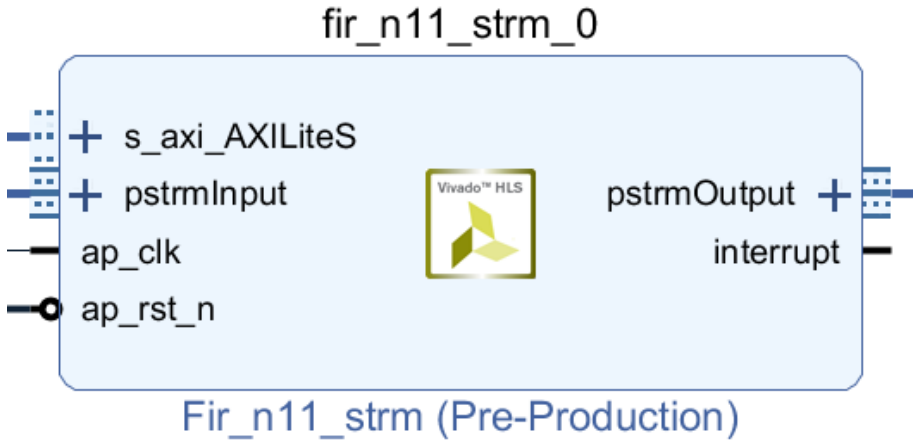
Loop

Block Diagram

- The IO of this project is similar to Lab2-2 (except the AXILiteS control signal)
- The PYNQ program is similar to Lab2-2 (except mmio)
- But, …



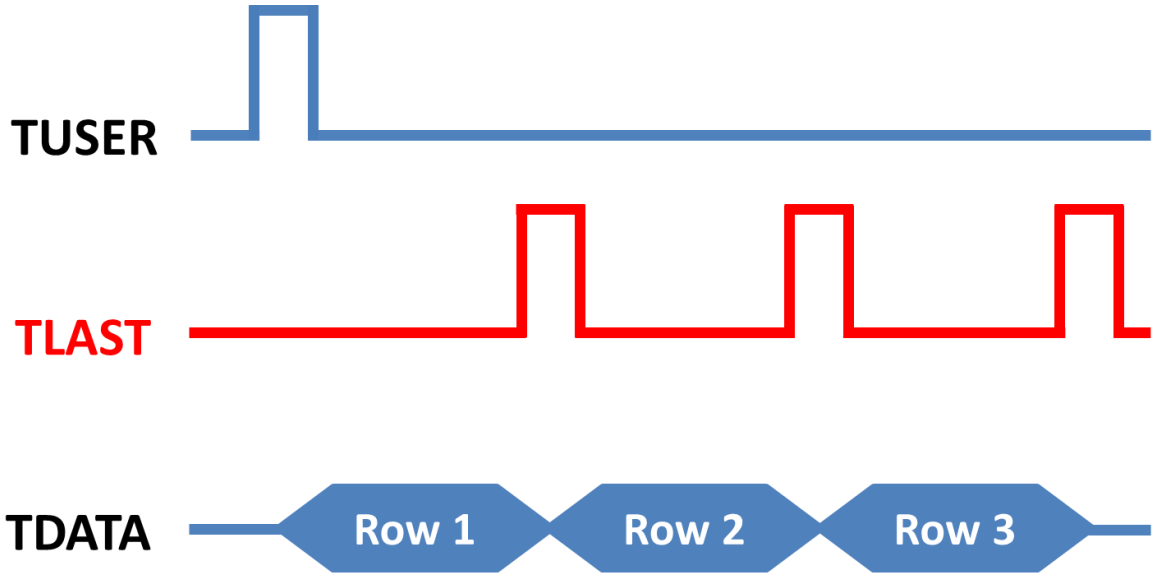
Block diagram of this project



Block diagram of lab 2-2

Solutions

- The hls::AXIvideo2Mat function handles the data transfer in AXI interface, and transform into hls::Mat datatype
- However, the data transfer format cannot be support by PYNQ host program



IO format in this project

IO format in lab 2-2

Insights

Interface:
Stream
TLAST
TUSER

Datatype:
ap_axiu<32,1,1,1>
hls::Mat<T>
hls::AXIvideo2Mat()

Wire connection

MMIO:
Write(0x14,rows)
Write(0x1C,cols)

XInk:
allocate

#pragma
offset
bundle
ap_stable

Cosim waveform

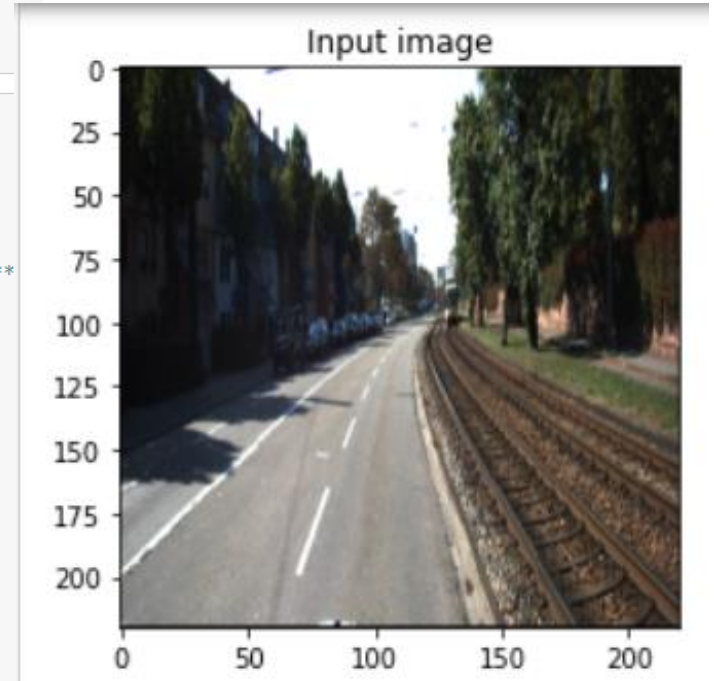
ILA

Overlay:
recvchannel()

Host Program

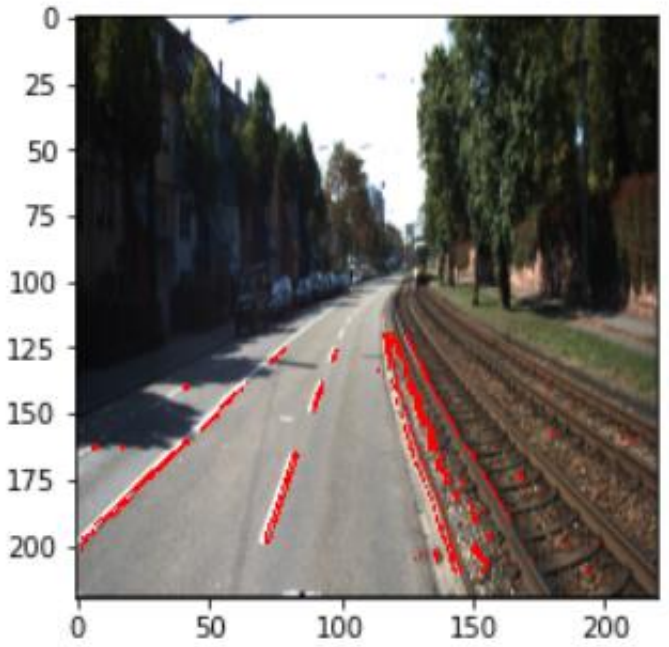
```
# ol = Overlay("/home/xilinx/IPBitFile/yclin/FIRN11Stream.bit")
# ipFIRN11 = ol.fir_n11_strm_0
ol = Overlay("/home/xilinx/IPBitFile/yclin/design_1.bit")
ipvideo_edge = ol.image_filter_0
ipDMAIn = ol.axi_dma_0
ipDMAOut = ol.axi_dma_1
# ipDMAIn = ol.axi_vdma_0
# ipDMAOut = ol.axi_vdma_1
```

```
n_row = 220
n_col = 220
# n_channel = 3
# numSamples = n_row*n_col
image = io.imread('car_view.png')[:n_row,:n_col,:]
# image = np.stack([np.eye(220), np.eye(220), np.eye(220)], axis=2)
# image_cat = image[:,:,0] + image[:,:,1]*256 + image[:,:,2]*(256**2) + 255*(256**
image_cat = np.concatenate([image, np.ones((n_row,n_col,1))*255], axis=2)
print("Shape of image_cat", image_cat.shape)
plt.imshow(image)
plt.title('Input image')
plt.show()

xlnk = Xlnk()
inBuffer0 = xlnk.cma_array(shape=(n_row, n_col, 4), dtype=np.uint8)
outBuffer0 = xlnk.cma_array(shape=(n_row, n_col, 4), dtype=np.uint8)
#for i in range(n_row):
    #inBuffer0[i] = image_reshape[i]
    #print(inBuffer0[i])
inBuffer0[:,:,:] = image[:,:,:]

# ipFIRN11.write(0x80, numSamples*4)
# ipFIRN11.write(0x00, 0x01)

# ipvideo_edge.write(0x14, n_row)
# ipvideo_edge.write(0x1C, n_col)
start = time()
ipvideo_edge.write(0x00, 0x01)
ipDMAIn.sendchannel.transfer(inBuffer0)
ipDMAOut.recvchannel.transfer(outBuffer0)
ipDMAIn.sendchannel.wait()
ipDMAOut.recvchannel.wait()
end = time()
```




```
plt.imshow(outBuffer0)
plt.plot()
```

```
[]
```



Video Demo on FPGA

Original

Lane detection