Combining OpenCV and High Level Synthesis to Accelerate your FPGA / SoC EV Application

Adam Taylor – Adiuvo Engineering & Training Ltd
Tends in Embedded Vision

• Ubiquity – Applications are wide spread – ADAS, Medical, AR, Drones, IIOT

• Process at the edge – leverage machine learning techniques

• Use Open High Level libraries like OpenCV and Frame works like OpenVX

• Security & Safety – Especially for autonomous drones and vehicles
Multiple Industries and Applications
Frame Rate, Resolution, BPP

![Graph showing Frame Rate against Data Rate for Different Pixel Sizes. The graph displays lines for 8 Bits Per Pixel, 10 Bits Per Pixel, 12 Bits Per Pixel, and 16 Bits Per Pixel.](image)
OpenCV What is it?

- Open Computer Vision
- Released in 2000
- Library of Image processing algorithms – includes aspects of machine learning
- Predominantly in C++ but bindings in Python, Java, Matlab
- Very popular allows us to accelerate our design
  - Focus upon our value added activity
Simple Open CV Example

Clockwise
Original, LoG, Canny, Sobel

OpenCV is great however it can be computationally intensive which will impact frame rate.

Several additional impacts at system level if addressed with processor /GPU

Cost of solution – more powerful processor
Power dissipation – critical for IIOT & Drones
Larger foot print – scalability
More Advanced – Object tracking non-HoG

- Convert to grey
- Blur
- Difference Against Reference

- Threshold
- Dilate & Identify Contours
- Draw Boxes around differences in unprocessed image
FPGA & SoC

• Programmable logic is ideal for image processing pipeline
  • Inherently parallel
  • Any to Any connectivity with sensors and downstream devices.
  • Can pipeline processing using large on-chip memory, thus avoiding the need to go to DRAM memory

• Traditionally
  • Requires algorithms being written from scratch in Verilog or VHDL
  • Cost and time scale driver

• SoC Architecture like the Zynq or MPSoC
  • Real time analytics and decision making at the edge
Bottlenecks

Low Latency / High Throughput: Pixel Streaming

Typical SoC

DSP/GPU
Gaussian Blur
Gaussian Blur
Subtract

Image Sensor

CPUs
DDR

Most Responsive

Programmable Logic
Gaussian Blur
Gaussian Blur
Subtract
CPUs

ZYNQ
ZYNQ MPSoC

Image Sensor

© Copyright 2017 Xilinx

#ESCconf
Selection of targets
History of PLD Design

Technology
Programming Method
Gate Count
Year

<table>
<thead>
<tr>
<th>Year</th>
<th>PLA</th>
<th>PAL</th>
<th>GAL / CPLD / FPGA</th>
<th>CPLD / FPGA</th>
<th>CPLD / FPGA</th>
<th>FPGA / SoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1969</td>
<td>Mask</td>
<td>Assembler</td>
<td>Schematic / HDL</td>
<td>HDL</td>
<td>HDL / HLS</td>
<td></td>
</tr>
<tr>
<td>&lt;50</td>
<td>&lt;100</td>
<td>600</td>
<td>9000</td>
<td>1 M</td>
<td>50 M</td>
<td></td>
</tr>
</tbody>
</table>

Gate Count

- PLA: <50
- PAL: <100
- GAL / CPLD / FPGA: 600
- CPLD / FPGA: 9000
- FPGA / SoC: 1 M
- FPGA / SoC: 50 M

Year

- 1969
- 1975
- 1985
- 1988
- 2000
- 2010+
It is not just LUT’s and FF’s

- Distributed RAM / Block RAM / Multiply Accumulators
- Ethernet MAC, PCIe / MGT
- Processors – ARM
- Even ADCs – 1 MSPS but see RFSoC
FPGA Trends – Not really Surprising

• Increase in Gate Count
• Increase of FPGA
• Addition of Specific Functions

These trends are responsible for the FPGA becoming the heart of many systems, providing game changing capabilities but they can come at a cost.
Addressing the Trends

• Leverage capabilities provided by Heterogeneous SoC
  • Fuse ARM cores with programmable logic

• Use Programmable Logic to implement the image processing pipelines

• Use Security features such as AES, Trust Zone, In built ADC to protect system

• Leverage Open Source Frame works? – Enter HLS
Focus on three aspects

• PYNQ – Python and Zynq – Examples above use a PYNQ

• Zynq / Zynq MPSoC / FPGA – High Level Synthesis IP Module Creation

• Zynq / Zynq MPSoC – System level design – SDSoC / reVISION
PYNQ
while(True):
    ret, frame_vga = videoIn.read()

    if (ret):
        grey = cv2.cvtColor(frame_vga, cv2.COLOR_BGR2GRAY)
        # calculate optical flow
        p1, st, err = cv2.calcOpticalFlowPyrLK(reference_grey, grey, p0, None, **lk_params)
        # Select good points
        good_new = p1[st==1]
        good_old = p0[st==1]
        # draw the tracks
        for i,(new,old) in enumerate(zip(good_new,good_old)):
            a,b = new.ravel()
            c,d = old.ravel()
            mask = cv2.line(mask,(a,b),(c,d), color[i].tolist(), 2)
            frame_vga = cv2.circle(frame_vga,(a,b),5,color[i].tolist(),-1)
        img = cv2.add(frame_vga,mask)
        #cv2.imshow('frame_vga',img)
        #cv2.imwrite('/home/xilinx/jupyter_notebooks/examples/data/opt_flow.jpg',img)
        frame_1080p[0:480,0:640,:] = img[0:480,0:640,:]
        hddl_out.frame_raw(bytarray(frame_1080p.astype(np.int8)))
        # Now update the previous frame and previous points
        reference_grey = grey.copy()
        p0 = good_new.reshape(-1,1,2)
        #p0 - cv2.goodFeaturesToTrack(reference_grey, mask = None, **feature_params)
What is HLS?

• High Level Synthesis enables us to define the algorithm using a language like C, C++ or System C

• Output is generated a VHDL or Verilog

• There are some constraints
  • Like HDL’s some synthesisable constructs, some non synthesisable constructs
HLS Overview

- **Scheduling** – Determines the order of operations
- **Binding** – Operations determined are mapped to available resources
- **Control Extraction** – Control logic and state machine generation for the flow.

```c
int foo(char x, char a, char b, char c) {
    char y;
    y = x*a+b+c;
    return y
}
```
Interfaces

• Interfaces are automatically generated (especially in SDSoC)

• Block-Level Interface – This is the default for a generic interface that provides clock and reset inputs along with start, ready, idle, and done handshaking signals.

• Port-Level Interface – This interface implements a more complete I/O protocol on the data ports such as AXI, AXI streaming, or AXI Lite.

```vhdl
entity auMedianBlurTop is
  port (  
    ap_clk : IN STD_LOGIC;
    ap_rst : IN STD_LOGIC;
    ap_start : IN STD_LOGIC;
    ap_done : OUT STD_LOGIC;
    ap_idle : OUT STD_LOGIC;
    ap_ready : OUT STD_LOGIC;
    in strm V V dout : IN STD_LOGIC_VECTOR (7 downto 0);
    in strm V V empty n : IN STD_LOGIC;
    in strm V V read : OUT STD_LOGIC;
    out strm V V din : OUT STD_LOGIC_VECTOR (7 downto 0);
    out strm V V full n : IN STD_LOGIC;
    out strm V V write : OUT STD_LOGIC;
    height : IN STD_LOGIC_VECTOR (15 downto 0);
    width : IN STD_LOGIC_VECTOR (15 downto 0));
  end;
```
HLS Support Image Processing

• reVision
  • SDSoC – System Optimising Compiler – Enable System development in C/C++
  • Acceleration Ready Open CV Libraries
  • Machine Learning Flow

• Vivado HLS
  • Support for OpenCV for test benching
  • Acceleration ready libraries used to create the IP Core
<table>
<thead>
<tr>
<th>OpenVX</th>
<th>Caffe</th>
<th>Application Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCV</td>
<td>DNN</td>
<td>Algorithm Development</td>
</tr>
<tr>
<td></td>
<td>CNN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSD</td>
<td>Platform Development</td>
</tr>
</tbody>
</table>
reVISION Machine Learning
SDSoC How does it work

- **Platform**
  - Predefined or Custom

- **Develop SW Application**
  - Develop in C or C++

- **Profile Application**
  - Determine performance bottle necks

- **Accelerate**
  - Use profile information to accelerate functions / Optimize
What makes a Platform?

Number of predefined platforms however you can define your own

Ability to define your own provides
1) Ability to design for a custom board just using SDSoC
2) Ability to define a base hardware platform in Vivado and accelerate SW performance further using remaining unused resources

Platform definition requires two things

Hardware Definition – AXI ports available, clocks, Interrupts & Resets
Software Definition – Library Files, Boot Files, Device Trees, linker Scripts
Simple Harris Example

```c
void auCornerHarrisTop(hls::stream< IN_TYPE > &in_img, hls::stream< ap_uint<32> > &keypoints,
                       uint16_t thresh, uint32_t *n_corners, uint16_t k)
{
```

![Simple Harris Example Diagram](image-url)
Simple Example

```cpp
void image_filter(AXI_STREAM& video_in, AXI_STREAM& video_out, int rows, int cols) {
    YUV_IMAGE img_0(rows, cols);
    YUV_IMAGE img_1(rows, cols);

    hls::AXIvideo2Mat(video_in, img_0);
    hls::GaussianBlur<3,3>(img_0, img_1, 0, 0);
    hls::Mat2AXIvideo(img_1, video_out);
}
```
Verification and Simulation
Results of Simple Blur

Input Image

HLS Result

Open CV Ref

HLS, OpenCV Difference
Performance Estimation

### Utilization Estimates

<table>
<thead>
<tr>
<th>Name</th>
<th>BRAM18K</th>
<th>DSP48E</th>
<th>FF</th>
<th>LUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Expression</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FIFO</td>
<td>0</td>
<td>-</td>
<td>20</td>
<td>80</td>
</tr>
<tr>
<td>Instance</td>
<td>6</td>
<td>8</td>
<td>925</td>
<td>1463</td>
</tr>
<tr>
<td>Memory</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Multiplexer</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Register</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>6</td>
<td>8</td>
<td>945</td>
<td>1544</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Latency (min)</th>
<th>Latency (max)</th>
<th>Interval (min)</th>
<th>Interval (max)</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear_Row_Loop</td>
<td>1920</td>
<td>1920</td>
<td>1938</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Row_Loop</td>
<td>2093040</td>
<td>2093040</td>
<td>1920</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Col_Loop</td>
<td>1926</td>
<td>1926</td>
<td>1920</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>auApplyMaskLoop</td>
<td>9</td>
<td>9</td>
<td>1920</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

### Latency (clock cycles)

#### Summary

- **Latency (min)**: 2094963
- **Max Latency (max)**: 2094963
- **Max Interval (max)**: 2094963
- **Type**: none

#### Detail

- **Instance**
  - N/A

- **Loop**

<table>
<thead>
<tr>
<th>Loop Name</th>
<th>Latency (min)</th>
<th>Latency (max)</th>
<th>Iteration Latency</th>
<th>Initiation Interval (achieved)</th>
<th>Target</th>
<th>Trip Count</th>
<th>Pipelined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear_Row_Loop</td>
<td>1920</td>
<td>1920</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1920</td>
<td>yes</td>
</tr>
<tr>
<td>Row_Loop</td>
<td>2093040</td>
<td>2093040</td>
<td>1938</td>
<td>-</td>
<td>-</td>
<td>1080</td>
<td>no</td>
</tr>
<tr>
<td>Col_Loop</td>
<td>1926</td>
<td>1926</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1920</td>
<td>yes</td>
</tr>
<tr>
<td>auApplyMaskLoop</td>
<td>9</td>
<td>9</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>9</td>
<td>no</td>
</tr>
</tbody>
</table>
Package and Use In Vivado

- Control bus – Enable / Disable
- Connect to Upstream components e.g. CFA, Chroma Conversion

Connect to downstream components e.g. VDMA
Wider Picture
Beyond HLS

Supported by System Level Synthesis Tools

Sensor Interface
- LVDS
- MIPI

Pre-processing
- ISP / Debayer
- Color Space Conversion
- Scaling

Computer Vision
- OpenCV / OpenVX
- Deep Learning (xNN)
- Optical Flow
- SLAM
- Stereo Vision

Output Interface
- HDMI
- GigE Vision
- CoaXPress
- Camera Link
- USB3
- SDI

Supported by High Level Synthesis Tools
Cloud and Edge Processing

• Number of Frameworks which have recently been announced to support processing at the edge and cloud
  • Including Reconfigurable Acceleration Stack (RAS) designed to support cloud based applications

• ReVision – Edge based Support for OpenVx and Caffe Frameworks
  • OpenCV – Acceleration Capable libraries
  • Machine Learning Inference – pre optimised for PL implementation
Wrapping it up

• EV is ubiquitous – Brings with it challenges

• Heterogeneous SoC provide ability to process at the edge

• HLS provides the ability to leverage OpenCV – Acceleration ready libraries

• What are you going to do with it?
Speaker/Author Details

- [www.adiuvoengineering.com](http://www.adiuvoengineering.com)
- adam@adiuvoengineering.com
- LinkedIn Adam Taylor CEng FIET
- Twitter @ATaylorCEngFIET
Thank You!

Questions?