FPGA video processing for target imaging systems

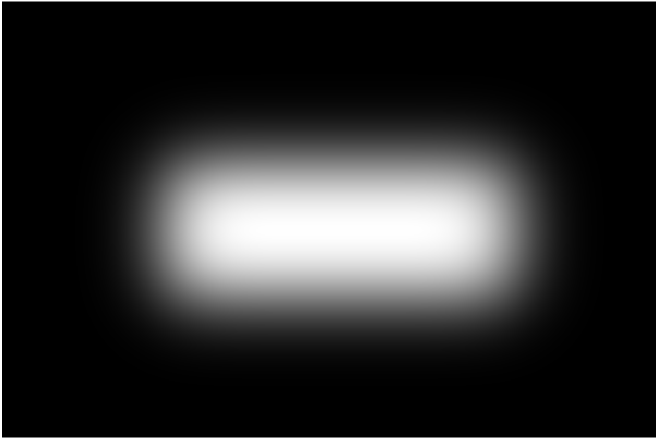
Håvard Gjersdal

University of Oslo

October 22, 2019

Motivation

- Target imaging systems will produce an image for each pulse.
- We must be able to detect errant beam conditions in time for the next pulse.
- FPGAs offer fast, fixed latency image processing.
- Framegrabber, Algorithms, EPICS

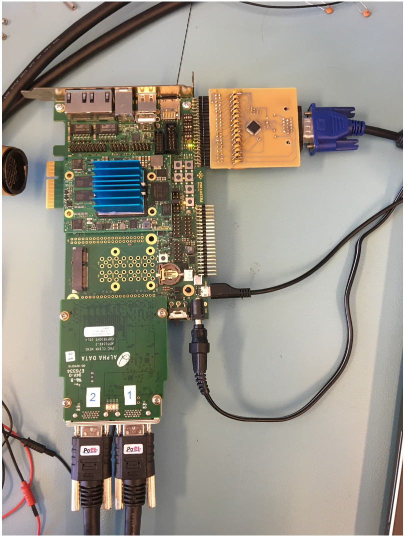


Hamamatsu ORCA-flash4.0 V3 digital CMOS

- 2048 $\times$ 2048 pixels, 16 bits per pixel
- 1.4 e$^-$ rms readout noise, 30k e$^-$ full well capacity
- Camera link Full Configuration Deca Mode
- In theory 100 frames per second, light sheet mode limits to 49 FPS
- 85MhZ, 5 pixels in parallel

Hardware

- Enclustra Mercury XU1+ Xilinx Zynq UltraScale+ MPSoC Module
- Enclustra Mercury PE1-400 base board
- Alpha data FMC for Camera Link
- In-house made VGA connector card



Frame grabber

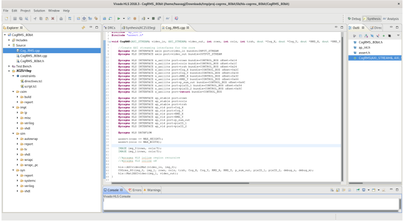
- Interface between camera and FPGA is 21 LVDS lines.
- 15 of these lines are inputted to the frame grabber
- 3 channels, each with 4 data lines, one clock line.
- 6 lines for camera controll and configuration, not part of grabber.

Vivado functional simulation

→ Frame Grabber → AXI4 stream →

Algorithms

- Developed in Vivado-HLS
- 5 pixels in parallel at 85 MhZ
- Before VDMA
- Median, CoG, (beam-in-box, max intensity)

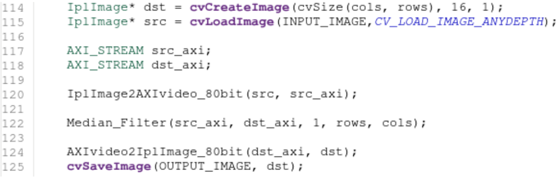


```
#include "ap_int.h"
#include "assert.h"

void CogRMS(AXI_STREAM& video_in, AXI_STREAM& video_out, int rows, int cols, int trsh, dout *Cog_X, dout *Cog_Y, dout *RMS_X, dout *RMS_Y

    //Create AXI streaming interfaces for the core
    #pragma HLS INTERFACE axis port=video_in bundle=INPUT_STREAM
    #pragma HLS INTERFACE axis port=video_out bundle=OUTPUT_STREAM

    #pragma HLS INTERFACE s_axilite port=rows bundle=CONTROL_BUS offset=0x14
    #pragma HLS INTERFACE s_axilite port=cols bundle=CONTROL_BUS offset=0x1C
    #pragma HLS INTERFACE s_axilite port=trsh bundle=CONTROL_BUS offset=0x24
    #pragma HLS INTERFACE s_axilite port=Cog_X bundle=CONTROL_BUS offset=0x2C
    #pragma HLS INTERFACE s_axilite port=Cog_Y bundle=CONTROL_BUS offset=0x34
    #pragma HLS INTERFACE s_axilite port=RMS_X bundle=CONTROL_BUS offset=0x3C
    #pragma HLS INTERFACE s_axilite port=RMS_Y bundle=CONTROL_BUS offset=0x44
    #pragma HLS INTERFACE s_axilite port=p_sum_out bundle=CONTROL_BUS offset=0x4C
    #pragma HLS INTERFACE s_axilite port=pixID_1 bundle=CONTROL_BUS offset=0x54
    #pragma HLS INTERFACE s_axilite port=pixID_2 bundle=CONTROL_BUS offset=0x5C
    #pragma HLS INTERFACE s_axilite port=return bundle=CONTROL_BUS

    #pragma HLS INTERFACE ap_stable port=rows
    #pragma HLS INTERFACE ap_stable port=cols
    #pragma HLS INTERFACE ap_stable port=trsh
    #pragma HLS INTERFACE ap_vld port=Cog_X
    #pragma HLS INTERFACE ap_vld port=Cog_Y
    #pragma HLS INTERFACE ap_vld port=RMS_X
    #pragma HLS INTERFACE ap_vld port=RMS_Y
    #pragma HLS INTERFACE ap_vld port=p_sum_out
    #pragma HLS INTERFACE ap_vld port=pixID_1
    #pragma HLS INTERFACE ap_vld port=pixID_2

    #pragma HLS DATAFLOW

    assert(rows <= MAX_HEIGHT);
    assert(cols <= MAX_WIDTH);

    IMAGE img_0(rows, cols/5);
    IMAGE img_1(rows, cols/5);

    //#pragma HLS inline region recursive
    //#pragma HLS inline off

    hls::AXIvideo2Mat(video_in, img_0);
    COGrms_80(img_0, img_1, rows, cols, trsh, Cog_X, Cog_Y, RMS_X, RMS_Y, p_sum_out, pixID_1, pixID_2, debug_n, debug_m);
    hls::Mat2AXIvideo(img_1, video_out);
}
```

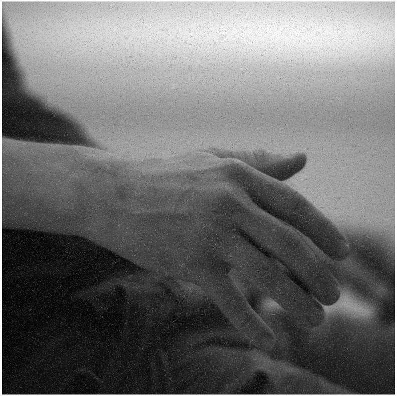
Demo of Median Filter and Center of Gravity

Frame Grabber → Median filter → CoG → VDMA → CrossGen → VGA

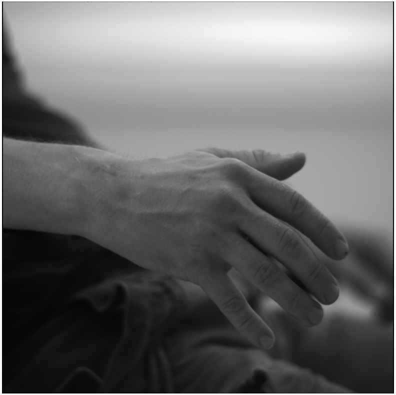
HLS test bench, C-simulation/RTL-simulation



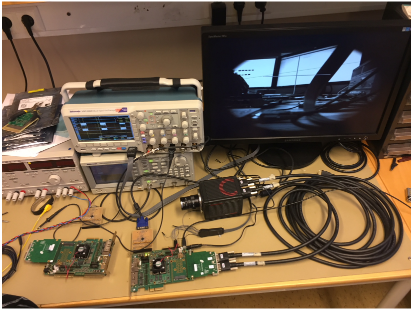
```
    IplImage* dst = cvCreateImage(cvSize(cols, rows), 16, 1);
    IplImage* src = cvLoadImage(INPUT_IMAGE,CV_LOAD_IMAGE_ANYDEPTH);

    AXI_STREAM src_axi;
    AXI_STREAM dst_axi;

    IplImage2AXIvideo_80bit(src, src_axi);

    Median_Filter(src_axi, dst_axi, 1, rows, cols);

    AXIvideo2IplImage_80bit(dst_axi, dst);
    cvSaveImage(OUTPUT_IMAGE, dst);
```

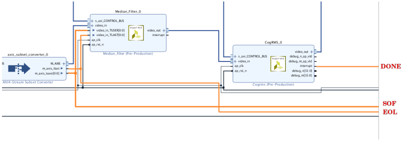


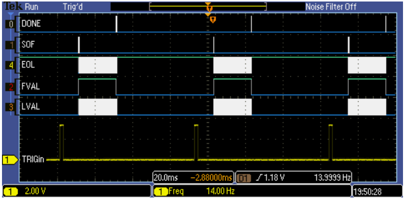
⇒



Timing

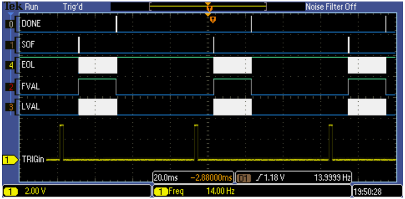


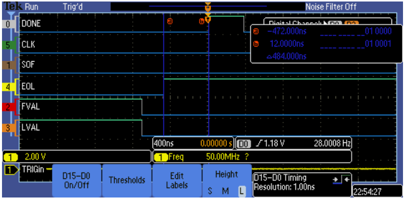

Median_Filter_0
s_axi_CONTROL_BUS
video_in
video_in_TUSER[0:0]
video_in_TLAST[0:0]
ap_clk
ap_rst_n
video_out
interrupt
Median_filter (Pre-Production)
axis_subset_converter_0
M_AXIS
m_axis_tlast
m_axis_tuser[0:0]
AXI4-Stream Subset Converter
CogRMS_0
s_axi_CONTROL_BUS
video_in
ap_clk
ap_rst_n
video_out
debug_n_ap_vld
debug_m_ap_vld
interrupt
debug_n[31:0]
debug_m[31:0]
Cogrms (Pre-Production)
DONE
SOF
EOL


Tek Run Trig'd
Noise Filter Off
0 DONE
1 SOF
4 EOL
2 FVAL
3 LVAL
1 TRIGin
20.0ms
−2.88000ms
D1
1.18 V
13.9999 Hz
1 2.00 V
1 Freq 14.00 Hz
19:50:28

Timing

- Trigger rate: 28 hZ
- Interrupt arrives 470ns after final EOL





Software

- Currently running bare-metal implementation for initialization and ISR, developed in XSDK
- Goal is to run EPICS on linux
- AreaDetector for camera read-out and configuration
- Configure IPs (thresholds, windows ...)
- Have been able to get framed from VDMA to V4L (Zybo-Z7 Zynq-7010)
- Have working Yocto layers for EPICS and AreaDetector
- A lot of work remaining on SW/EPICS

Summary

- Thanks to David Michael Bang-Hauge!
- Thanks to Ole Røhne!
- More info on the wiki
- We should be able to deliver a test platform with camera by christmas.

Thanks for listening!