Programming vision applications on Zynq using OpenCV and High-Level Synthesis

Kees Vissers
MPSoC, July 2013

Video And Vision processing

Broadcast Reference Monitor

A&D UAV

Driver Assist

HD Surveillance

Video Conferencing

Studio / Cinema Camera

From Pixels

to information

Digital Signage

Consumer Video Displays

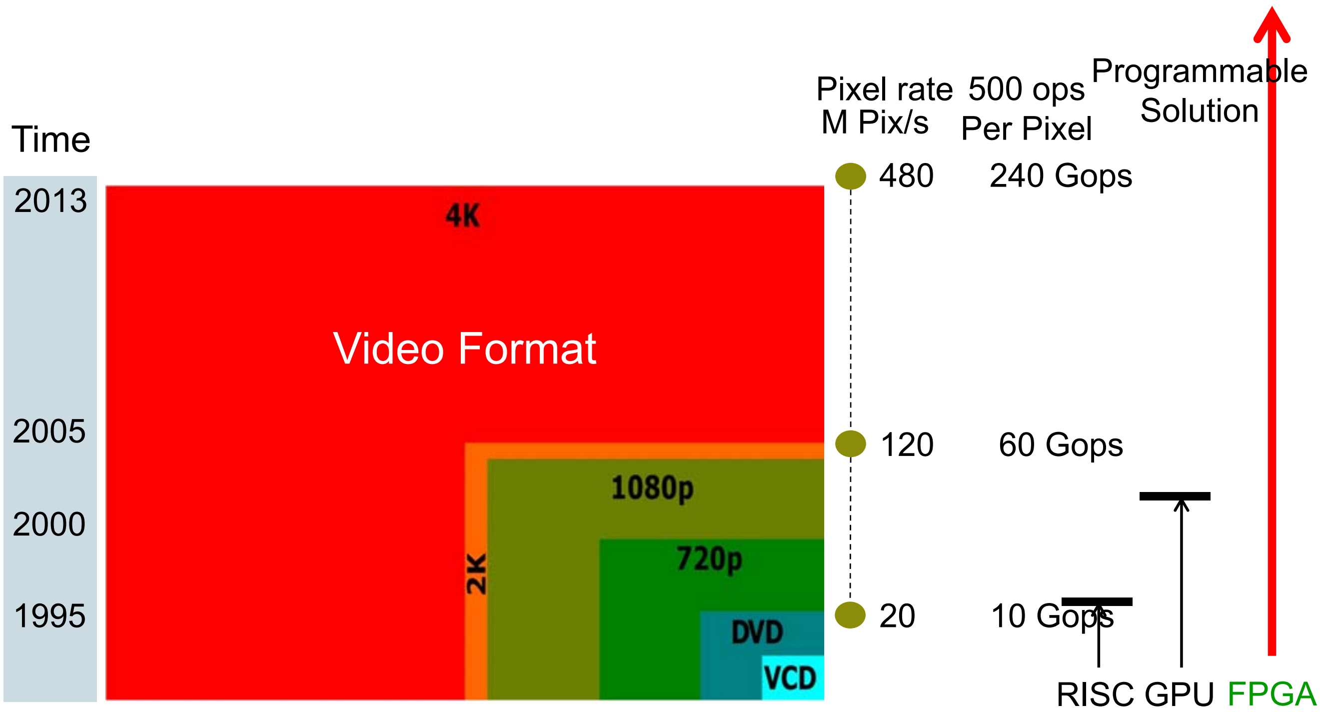
Medical Display

A&D UAV

Office-class MFP

Machine Vision

Required Pixel Rate Processing vs. Capabilities

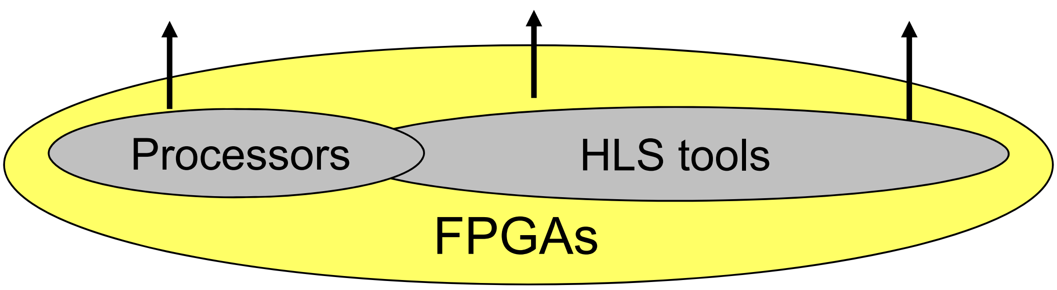


Hi Res. Display Pixel Rate Processing Exceeds RISC Based Capabilities

Processors and Pipelines

Design approach	RISC Proc.	Proc. w/ accels.	Folded datapath	Pipelined datapath	Replicated datapath
clock:sample	1000:1	100:1	10:1	1:1	1:10
Data Rate (200MHz clock)	200Ks/s	2Ms/s	20Ms/s	200Ms/s	2 Gs/s

Applications control → audio → mobile video → HDTV → comms → networking



Zynq Products in context for video

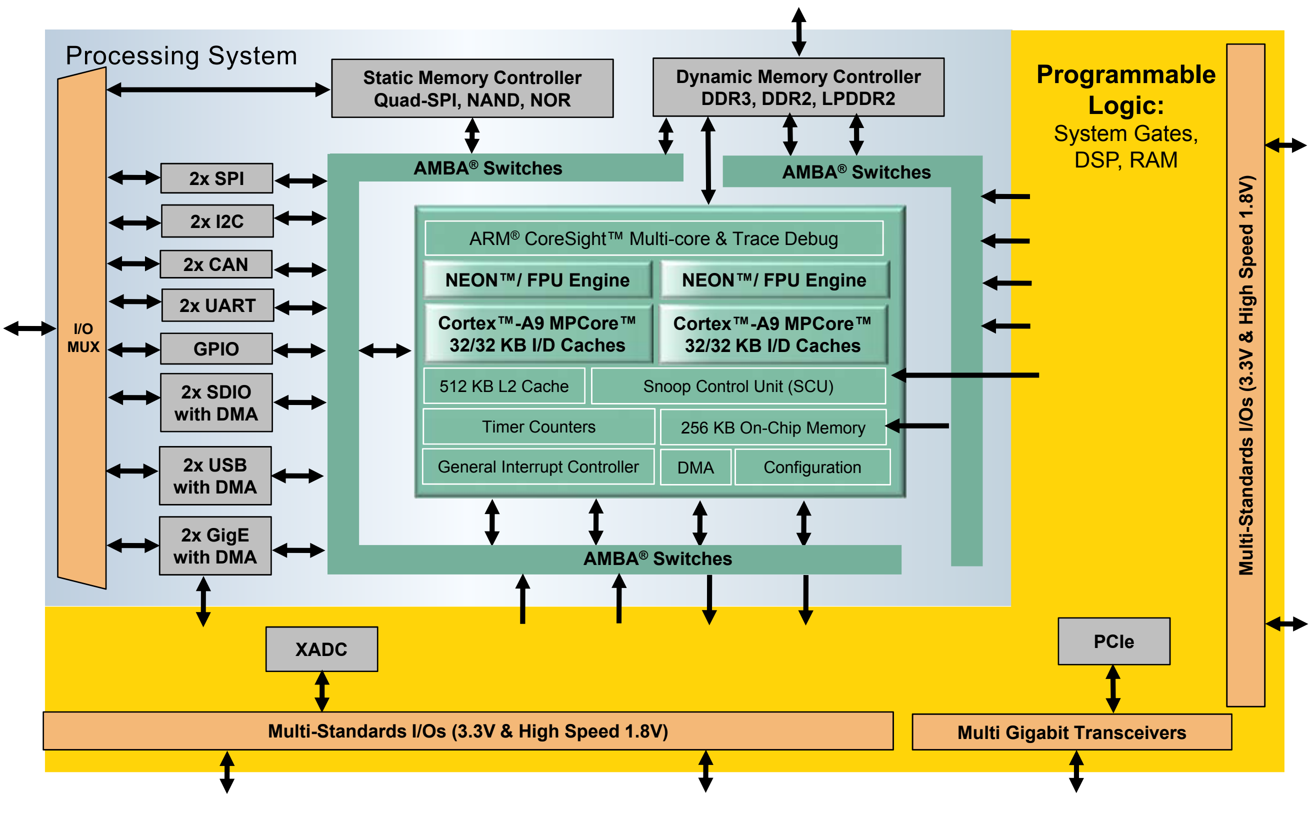
Design approach	RISC Proc.	Proc. w/ accels.	Folded datapath	Pipelined datapath	Replicated datapath
clock:sample	1000:1	100:1	10:1	1:1	1:10
Data Rate (200MHz clock)	200Ks/s	2Ms/s	20Ms/s	200Ms/s	2 Gs/s

Applications frame rate processing → line rate processing → HDTV pixel rate → ...

Arm A9 processors
1-2 Gops

Fabric
10 – 500 Gops

Zynq platform



Video Board: Zync 702 board

Programming

- **Image processing often programmed using streaming and OpenCV libraries**
- **Processor runs Linux, and a window system (Qt)**
- **On chip performance monitors tied to running Display, processor load and FPGA to external memory load**

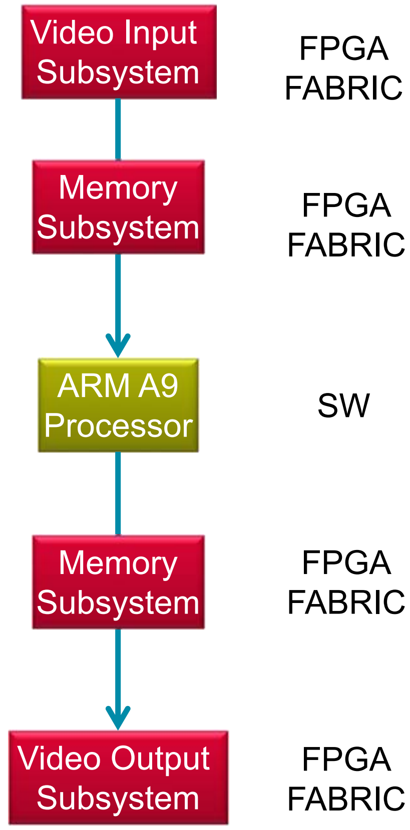
- **Basic idea:**
  - take the edge detect of the current frame and
  - the edge detect of previous frame
  - Subtract: if the same: nothing, if different: show fat pixel

Application Code

Application on Zynq

Source Code Mapping

```
while(1){
//Get the image frame from the video input
frame_current = cvQueryFrame(capture);

//Detect edges in the current frame
cvSobel( gray_current, edge_current2, 1, 0, aperature_size );
cvSobel( gray_current, edge_current1, 0, 1, aperature_size );
cvAdd(edge_current2,edge_current1,edge_current2,NULL);
cvConvertScale(edge_current2,edge_current,scale,0);
cvThreshold(edge_current,edge_current,5,255,CV_THRESH_BINARY);

//Detect edges in the previous frame
cvSobel( gray_prev, edge_prev2,1, 0, aperature_size );
cvSobel( gray_prev, edge_prev1,0, 1, aperature_size );
cvAdd(edge_prev2,edge_prev1,edge_prev2,NULL);
cvConvertScale(edge_prev2,edge_prev,scale,0);
cvThreshold(edge_prev,edge_prev,5,255,CV_THRESH_BINARY);

//Detect edges that are only present in the edge_current image
detect_new_edges(edge_prev,edge_current,new_edge);

//Remove noise from the new edges
cvSmooth(new_edge,filtered_new_edges,CV_MEDIAN,7,7);

//Combine new edges with current frame
//Highlight new edges in red
highlight_blend(frame_current,filtered_new_edges,output_frame);

//Copy current frame into previous frame
cvCopy(frame_current,frame_prev,NULL);

//Display output frame
cvShowImage("Detector Output",output_frame);
}
```

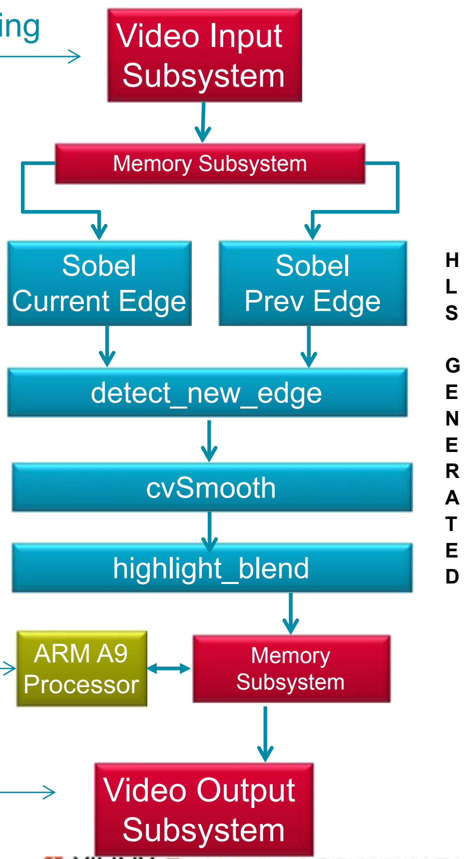


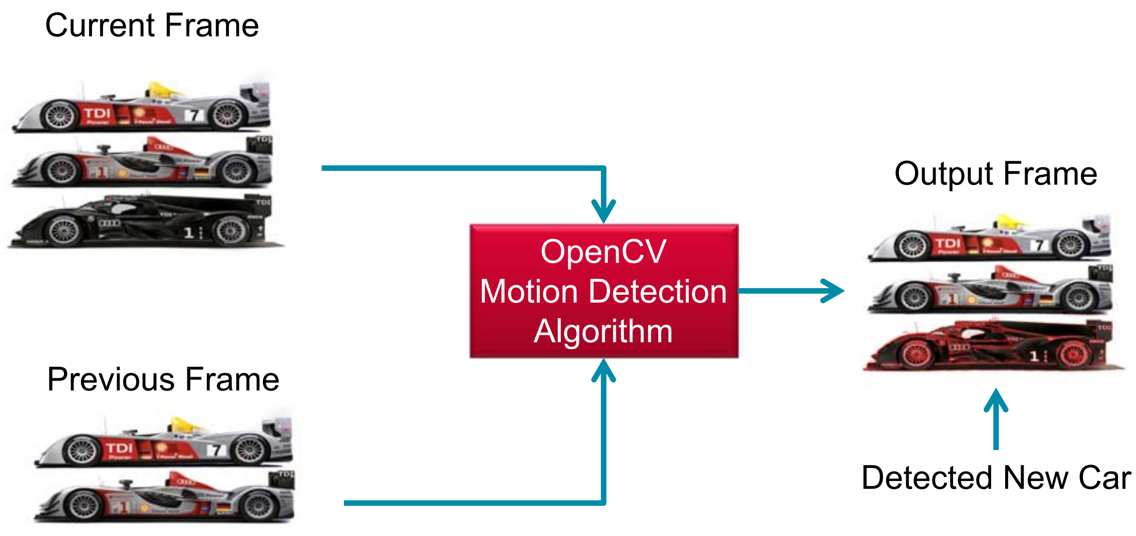
Application Code

Application on Zynq

Source Code Mapping

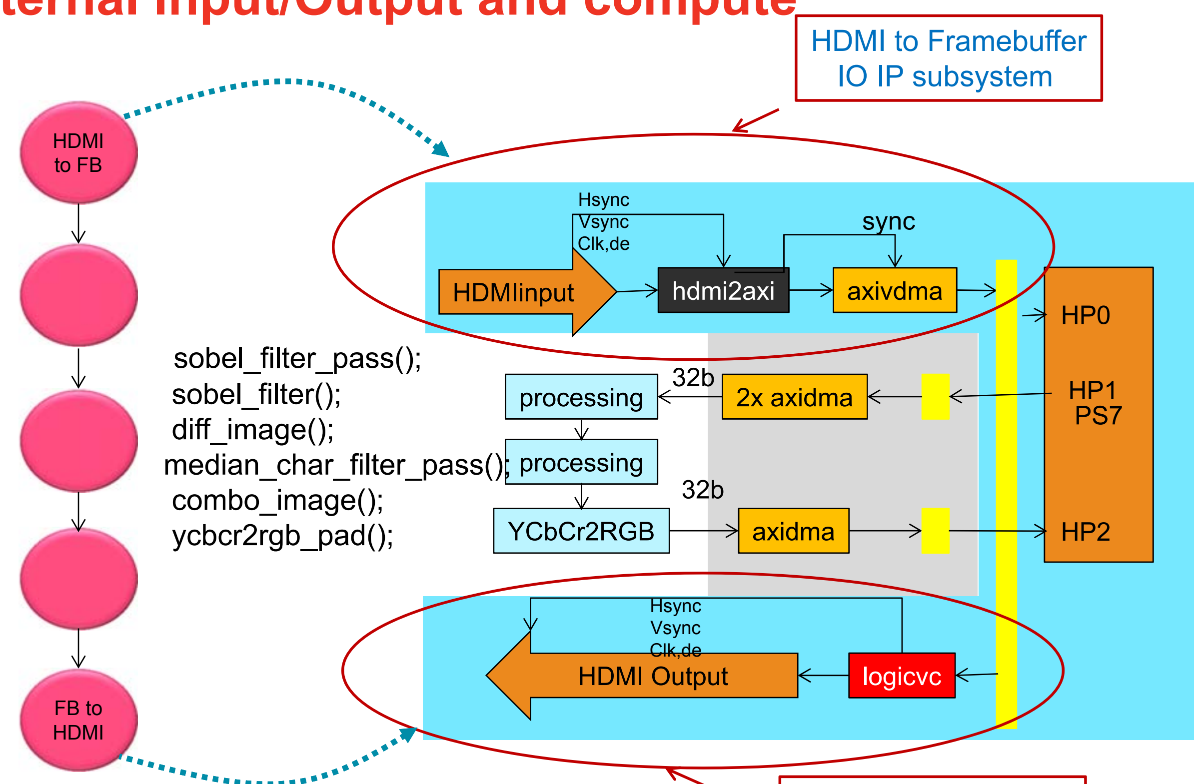
```
while(1){
//Get the image frame from the video input
frame_current = cvQueryFrame(capture);

//Detect edges in the current frame
cvSobel( gray_current, edge_current2, 1, 0, aperature_size );
cvSobel( gray_current, edge_current1, 0, 1, aperature_size );
cvAdd(edge_current2,edge_current1,edge_current2,NULL);
cvConvertScale(edge_current2,edge_current,scale,0);
cvThreshold(edge_current,edge_current,5,255,CV_THRESH_BINARY);

//Detect edges in the previous frame
cvSobel( gray_prev, edge_prev2,1, 0, aperature_size );
cvSobel( gray_prev, edge_prev1,0, 1, aperature_size );
cvAdd(edge_prev2,edge_prev1,edge_prev2,NULL);
cvConvertScale(edge_prev2,edge_prev,scale,0);
cvThreshold(edge_prev,edge_prev,5,255,CV_THRESH_BINARY);

//Detect edges that are only present in the edge_current image
detect_new_edges(edge_prev,edge_current,new_edge);

//Remove noise from the new edges
cvSmooth(new_edge,filtered_new_edges,CV_MEDIAN,7,7);

//Combine new edges with current frame
//Highlight new edges in red
highlight_blend(frame_current,filtered_new_edges,output_frame);

//Copy current frame into previous frame
cvCopy(frame_current,frame_prev,NULL);

//Display output frame
cvShowImage("Detector Output",output_frame);
}
```





XILINX ALL PROGRAMMABLE.

External Input/Output and compute



- Frame Buffer is the application level abstraction for HDMI input/output

XILINX ALL PROGRAMMABLE.

Laptop demo with webcam

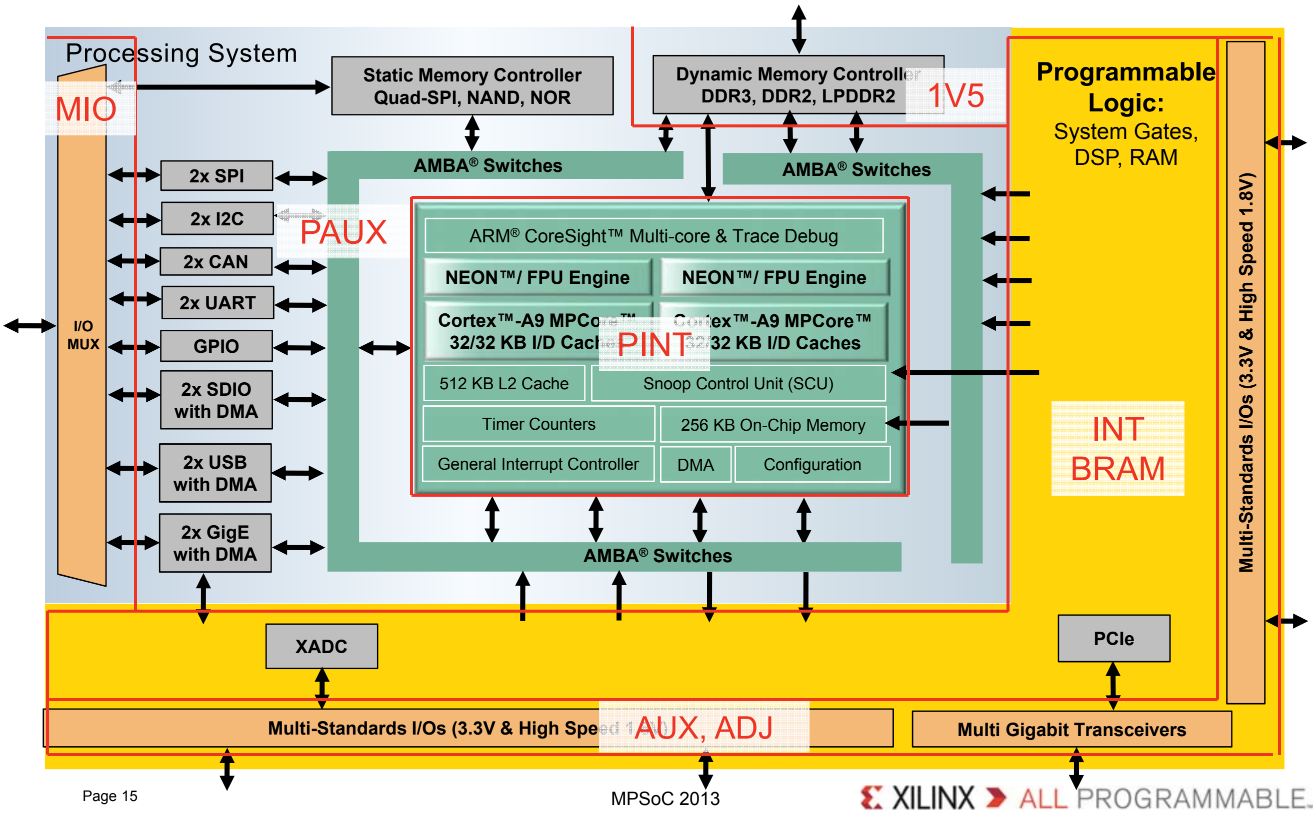
- Exactly the same OpenCV image processing pipe
- Using OpenCV libraries optimized for Intel SSE vector processing
- Webcam is 1280 x 720 (720p)
- Runs on Intel i7 with ~2.7GHZ processor and 8Gbyte DRAM
- Net result in the range of one frame every 1-2 seconds

- Demo

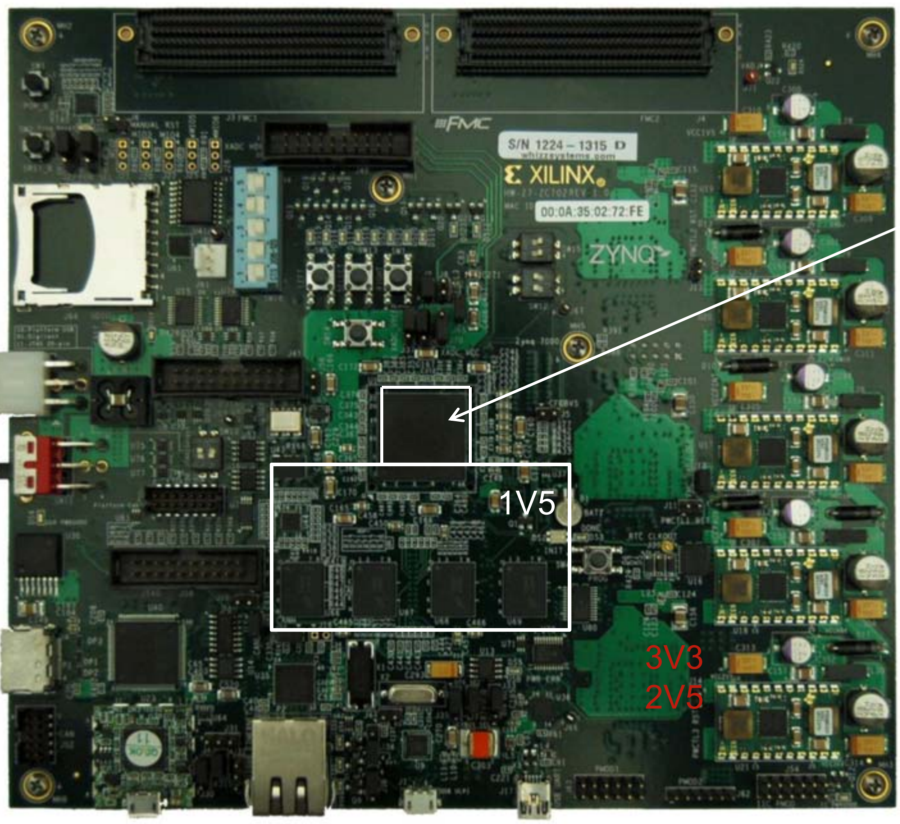
Complete setup:

- Model train
- HDTV camera 1080p 60Frames per second, HDMI
- Board, with application running, linux, Qt, processor + Bus -load
- Mouse tied to a register to set threshold value dynamically
- HDTV, 1080p , HDMI

Power zones

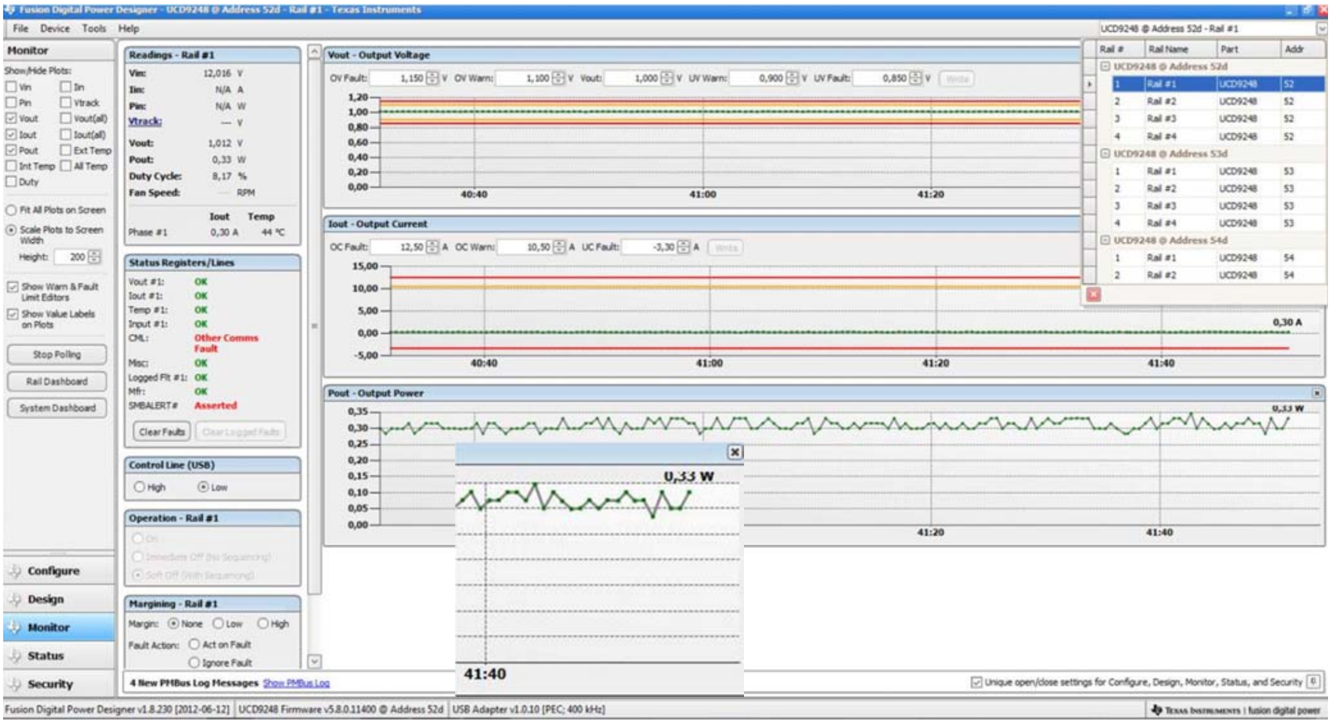


Power zones



- INT
- PINT
- AUX
- PAUX
- ADJ
- BRAM
- MIO
- 1V5 = DDR
- 3V3
- 2V5

Power Measurement output on running board



- Power in Watts
- Only few measurements/sec
- 10% noise
- 10% difference between bit streams

XILINX ALL PROGRAMMABLE.

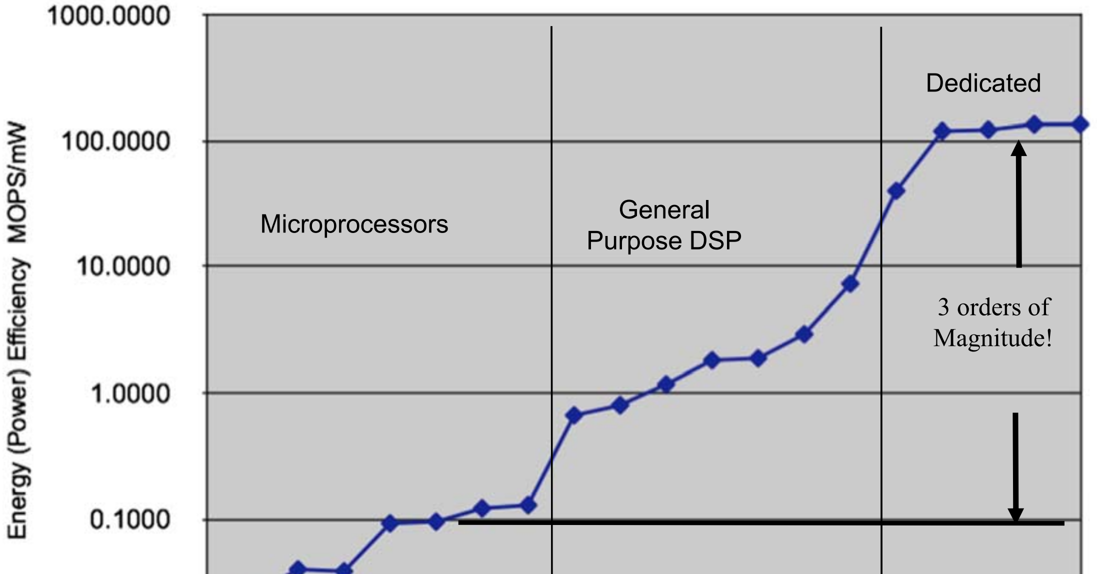
Performance and Power Measurements

- **All Pixel processing with OpenCV libraries running on ARM A9: one frame per 13 secs**
- **All Pixel processing with C++ running on A9: 1 frame per 1-2 seconds**
- **All Pixel processing with C++ libraries implemented via HLS in FPGA: 60 frames per second, FPGA runs 130MHz**
- **A9 processors : 500mW – 800mW**
- **FPGA fabric fully running: 500mW – 1W**
- **On Chip I/O few hundred mW, on board DRAM 800mW**
- **Result: ~ 100x speedup at SAME power consumption, ~100GOps**
- **Energy efficiency is in the 100 – 200 Gops/W range for the FPGA in the complete system!**
- **You can put your finger on the running chip in the system, warm but not HOT : less than ~2 W!**

XILINX ALL PROGRAMMABLE.

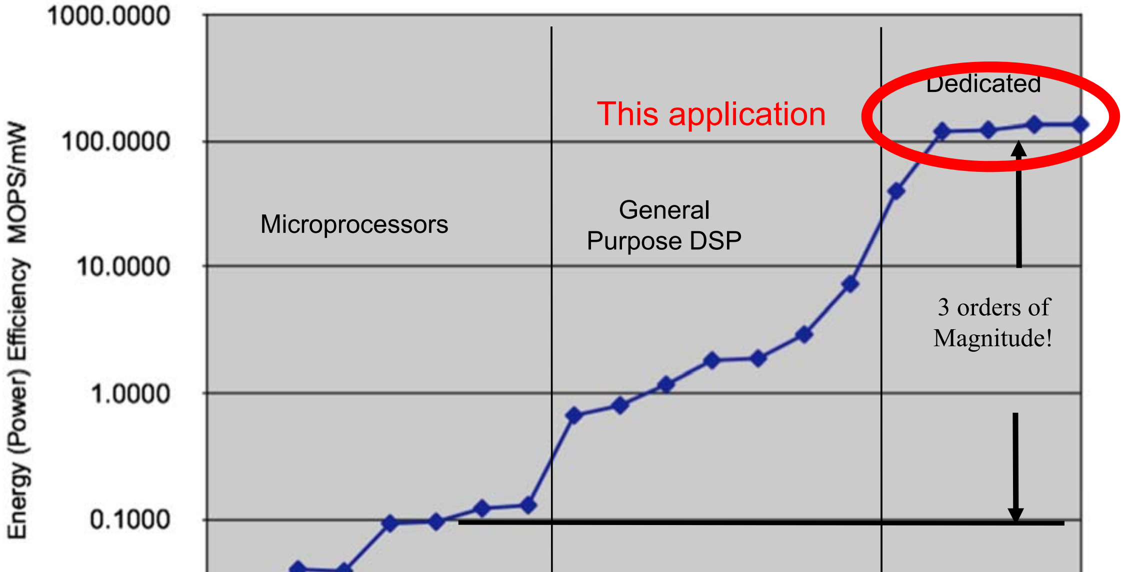
Energy Efficiency (MOPS/mW or OP/nJ)

- Courtesy Bob Brodersen, based on published results at ISSCC conferences.



Energy Efficiency (MOPS/mW or GOPS/W)

- Courtesy Bob Brodersen, based on published results at ISSCC conferences.



Conclusion

- Every processor benefits from a combination with FPGA: Zynq device
- Every FPGA benefits from a combination with a processor: Zynq device
- We have shown an application programmed in OpenCV, leveraging Vivado HLS running on a 1-2W Zynq device at 1080p 60fps real-time.

Special thanks to the team that worked on the demo:

Jack Lo, Fernando Martinez Vallina, S. Mohan, Vinod Kathail,

and to many colleagues in the Vivado High Level Synthesis team and the Video Platform teams.

You can do this too:

- OpenCV and HLS video: http://www.xilinx.com/csi/training/vivado/leveraging-opencv-and-high-level-synthesis-with-vivado.htm
- OpenCV and HLS application note: http://www.xilinx.com/support/documentation/application_notes/xapp1167.pdf
- Xilinx Zynq 702 board: http://www.xilinx.com/products/boards-and-kits/EK-Z7-ZC702-G.htm
- http://www.zedboard.org/