Embedded Vision

FPGA image processing and alternatives

Sensor to Image GmbH

Lechtorstrasse 20

D 86956 Schongau

Website: www.sensor-to-image.de

Email: email@sensor-to-image.de

Sensor to Image GmbH

- Company
  - Founded 1989 and privately owned
  - Company goal: development, production and service of OEM image processing hardware tool
  - Production of 1000-2000 boards/year
  - Team of 9 developers, 2 people in production and 3 in administration
  - Full service from training → development → layout → software → sample production → redesign → series production → support → repair is possible
- Development
  - High speed and impedance controlled layout, pitch > 0.8 mm
  - FPGA design tools experience with Altera, Xilinx, Lattice and MicroSemi tools
  - FPGA simulation and verification with ModelSim 10.x
  - Embedded C-Compilers for FPGA based 8051, Coldfire, Microblaze/NIOS, ARM in Zynq/Cyclone V SoC and BLACKFIN running Linux
  - Capability for writing PC drivers VC6 – VC2015
- Production
  - Automatic Pick & Place machine with manual BGA placement and part handling down to size 0402
  - Vapor phase oven for RoHS conform soldering PCB size <= 30x30 cm
  - Thermal box for testing systems up to 50x50x50 cm under operation from -40˚ C up to 90˚ C
  - Mixed signal oscilloscopes with bandwidth of 5 GHz

Certificated & Applications

- EN ISO9001:2000 and EN ISO 14001 certified procedures since 1996
- CE Production to EN 61000-6-3:2007 and EN 61000-6-4:2007 standard
- Production to MIL-STD-810E 501.3, ...502.3, … 514.4-All, … 516.4 and VG95328, VG95373, VG96903 and IPC610A/B/C
- Production to Marine standard EN 60945:2002
- Asia OEM application: inspection of copper in PCB production, >= 700 PCIe line scanning systems
- German OEM application: A0 printer & scanner electronics, >= 2000 CIS & USB3 IF boards
- American OEM application: 3D OEAM camera, standard CMOS Sensor FPGA pre-processing and embedded Linux, >= 1000 system

Agenda

- What is "embedded vision“
  - Marketing phrase
  - Samples of "embedded vision“ over the years
- CPU, FPGA , SoC und "embedded vision“
  - (Dis)advantage pure CPU solution
  - (Dis)advantage pure FPGA solution
  - (Dis)advantage SoC solution
  - What else is around
- How to plan and design an „embedded vision“ system
  - Sensor and data rates around the application
  - Develop and test the algorithm for the application
  - Implement and test the algorithm for the application
  - Possible tool chains

1. What is "embedded vision“ - Marketing

Embedded Vision Alliance: https://www.embedded-vision.com/what-is-embedded-vision

BASLER, Camera supplier: https://www.baslerweb.com/de/vision-campus/kamera-technologie/was-ist-embedded-vision/

Inspect Online, paper: http://www.inspect-online.com/topstories/topics/embedded-vision-die-zukunft-der-bildverarbeitung

Sensor to Image: https://www.s2i.org/index.php/products-mainmenu-33/cameras-mainmenu-39/intelligent-cameras-mainmenu-61

Maybe a short version is:

- a system, which can solve a compete application
- More then a simple, traditional camera
- Could be an FPGA with image sensor, a Raspberry PI with a MIPI camera module or THE cloud
- Trying to get attention for mass market applications: UPS tracking box, self driving car, intelligent rubbish bins, …
- Trying to phrase something for new technologies, which might need to new product approaches - SoC
- At least a real system with light-lense-sensor-electronics-actor

Samples of "embedded vision“ over the years

To be discussed, as yours will be unique

2. **CPU**, FPGA , SoC and "embedded vision“ things to look for in a CPU based system, part1

- Programming mainly done in C
- Data rate into the CPU at 640*480@60fps 18MByte/sec, 1080p60 120MByte/sec, …
- Easy to program a solution in a single-thread application, Multi-threading would be nice
- Which real performance is needed – depending on:
  1. Image size
  2. Frames/sec
  3. Algorithms

What can you expect in the embedded world → FPGA Tage 2016, Anton Zöchbauer, Sensor to Image

- Dhrystone 2.1, Synthetic performance benchmark
- Easy-to-use
- General-purpose (“integer”)
- 12 different subroutines:
- Pointer and array access operations
- Simple arithmetic and logic operations
- If statements and string operations
- Result: DMIPS(/MHz), Dhrystone Million Instructions Per Second, 1 DMIPS = 1757 Dhrystones / s

Dhrystone benchmark results

CPU	frequency [MHz]	Dhrystone run [us]	Dhrystones / second	DMIPS / second	DMIPS / MHz
Nios II gen2	100	10.660	93,808	53.4	0.533
MicroBlaze	100	9.430	106,044	60.4	0.603
Zynq Bare Metal	666	0.324	3,086,419	1,756.6	2.637
Cyclone V SoC Bare Metal	925	8.320	120,180	68.4	0.073
Zynq Linux	666	0,482	2,075,980	1181,5	1,774
Cyclone V SoC Linux	925	0.347	2,884,616	1,641.8	1.775
Intel Core i3-4130	3,400	0.035	28,193,634	16,046.5	4.585

2. **CPU,** FPGA , SoC and "embedded vision“ things to look for in a CPU based system, part2

Pro:

- Single thread C programming is a good solution for complex image analysis as you can very easy follow complex image structures with simple post/pre- increment of image columns and lines
- Many algorithms are available as free or paid versions. E.G. HALCON http://www.mvtec.com/products/halcon/product-information/technical-data/ has a library with about 2000 operators (where les then 100 are available for OpenCL optimization)
- Modern desktop CPU cache technology enable fast execution of quite chaotic code execution with quite perfect branch and jump prediction

Con:

- When multi threading needed the right partitioning is needed, which is almost the same problem as running standard FPGA processes
- Power and size

Think about:

- Single/Multi-Tread Application "automatic operator parallelization that actively supports this speed enhancement. Of course, not all vision operations profit in the same way from parallelization“ → http://www.mvtec.com/products/halcon/product-information/multi-core-performance/
- Choice of OS is important to keep up with the constant incoming data rate → standard Windows/Linux

2. CPU, **FPGA** , SoC and "embedded vision“ things to look for in a FPGA based system, part1

- Programming mainly done in **VHDL/Verilog** – C – SDSoC - …
- Data rate into the CPU at 640*480@60fps 18MByte/sec, 1080p60 120MByte/sec, …
- No easy to program any solution
- Which real performance is needed – depending on:
  1. Image size
  2. Frames/sec
  3. Algorithms

So FPGA is a special solution for high bandwidth - low latency application, where one of the key factor is the bandwidth of the FPGA memory controller as the external DDRx → FPGA Tage 2016, Werner Feith, Sensor to Image

- AXI3 max. burst length=16, AXI4 max. burst length=DDRx row size
- Low/mid class FPGA running AXI at 150MHz, 64bit → 1.2 GByte/sec → <10 Gbps
- FPGA internal bus bandwidth 10 Gbps → 1.2 GByte/sec → 10 1080p60 video stream → less then 5 internal operation in real-time on full 1080p60 images
- External/internal memory caching
- out-of-order DDRx access performance

Which mechanisms ensure high bandwidth?

- ✓ Read and understand DDRx data sheet
- ✓ Long burst on DDRx
- ✓ Right scheduler on DDRx
- ✓ Right memory organization on UMA architecture
- ✓ Match DDRx and FPGA memory controller settings
- × UMA, unified memory architecture = CPU + processing on 1 memory bank
- × Short burst
- × DDRx

FPGA bandwidth values

FPGA type	memory clock [MHz]	memory width	practical FIFO bandwidth [GBit]	internal bus multiply 2/4/8	internal AXI bus [bit]	internal F DIV 1/2/4	internal AXI speed [MHz]	FPGA AXI max [bit]	FPGA AXI max [MHz]
Artix7-2	400	16	4,2624	2	32	1	400	128	150
Artix7-2	400	16	4,2624	8	128	4	100	128	150
Artix7-2	333	16	3,548448	4	64	2	166,5	128	150
Artix7-2	333	16	3,548448	8	128	4	83,25	128	150
Artix7-2	200	16	2,1312	4	64	2	100	128	150
Artix7-3	533	16	5,679648	8	128	4	133,25	128	150
Kintex7-2 HP	666	32	14,193792	8	256	4	166,5	256	250
Kintex7-2 HR	533	32	11,359296	8	256	4	133,25	256	250
Kintex7-3 HP	800	32	17,0496	8	256	4	200	256	250
Kintex7-3 HR	533	32	11,359296	8	256	4	133,25	256	250

2. CPU, **FPGA** , SoC and "embedded vision“ things to look for in a FPGA based system, part2

Pro:

- (very) fast (for the right application)
- Power and size

Con:

- (maybe) difficult to partition development, test and maintain over the years

Think about:

- How to develop an image processing algorithm in VHDL with the problems of
  1. Out of order access to the next pixel
  2. No standard pixel cache around
  3. Algorithm typical require FLOAT operation → FPU/slow → fixed point float format
  4. How to manage and test changes in VHDL code of fixed point float format
- Internal/external IP maintenance
- Overall development effort

2. CPU, FPGA , **SoC** and "embedded vision“ things to look for in a SOC based system, part1

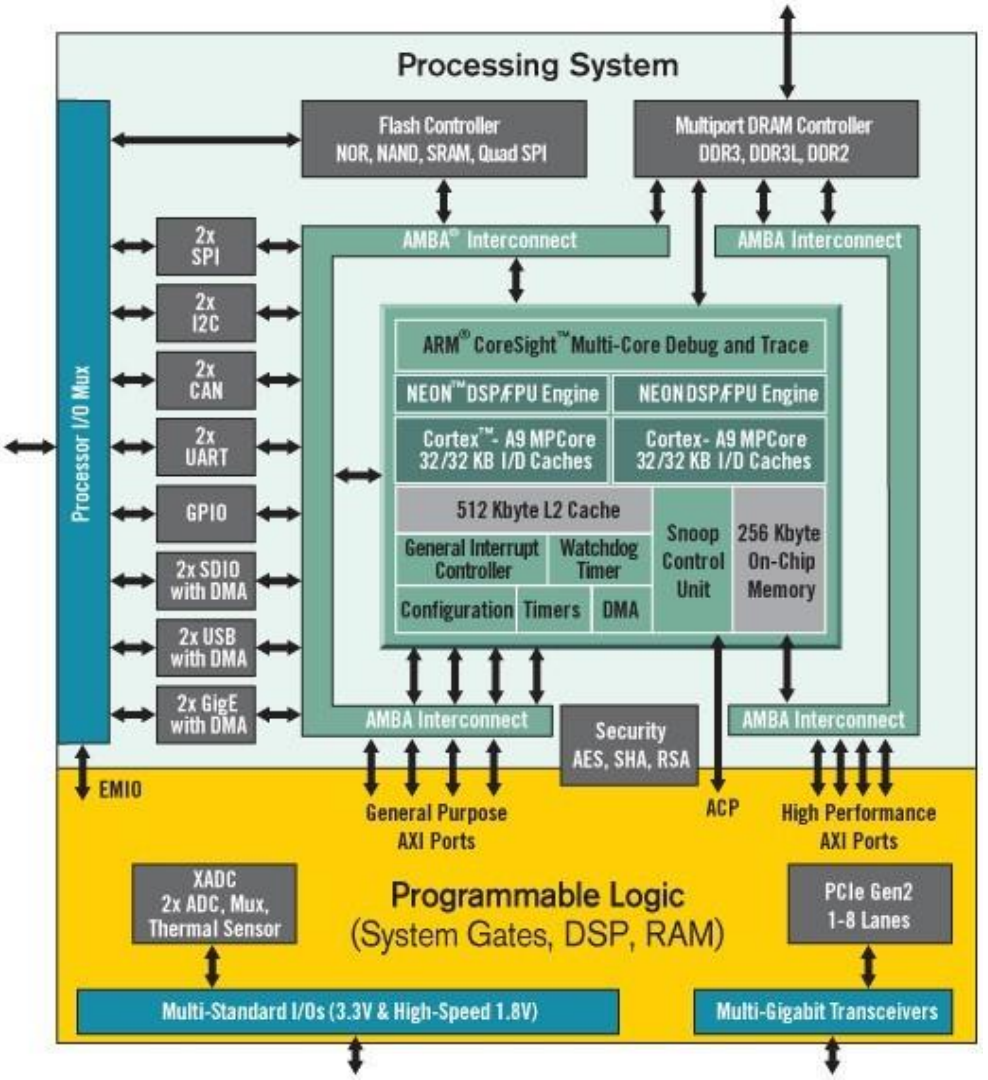
- Programming mainly done in a combination of CPU and FPGA
- Data rate into the CPU at 640*480@60fps 18MByte/sec, 1080p60 120MByte/sec, …
- easy to program a low performance solution in C, faster then Raspberry PI with better IO interfaces, but at higher cost
- Which real performance is needed – depending on:
    1. Image size
    2. Frames/sec
    3. Algorithms

What is an SoC → FPGA Tage 2016, Anton Zöchbauer, Sensor to Image

- See remarks from page 7 on CPU performance
- See remarks from Page 10 on FPGA performance

Xilinx Zynq

- 28 nm techology
- Processor System (PS)
  - 32 bit dual-core ARM® Cortex™-A9 MPCore
  - ARM AMBA AXI3 Interconnect
  - I/O peripherals
  - Memory interfaces
- Programmable Logic (PL)
  - Dynamical & partial reconfiguration
  - Complex logic block
  - 36 Kb Block RAM
  - 48 bit DSP



2. CPU, FPGA , **SoC** and "embedded vision“ things to look for in a SOC based system, part2

Pro:

- You can solve many-all problem from isolated CPU and FPGA systems with this architecture. But that you can does not automatically you easily succeed in solving the problems
- Power and size

Con:

- (very) difficult to find the optimal partition layout of hard- and software, development, test and maintain over the years

Think about:

- The typical FPGA=PL memory performance are not solved with a SoC, maybe getting even lower bandwidth, as the PS and PL typically work from a common memory adding more out-of-order accesses
- SoC CPU can not compete in pure C performance numbers with x86-64 desktop CPU

2. CPU, FPGA , SoC and "embedded vision“ which other possibilities are around

GPU

Pro:

- Many CPU to be programmed in OpenGL/CUDA/C/…
- Robust and available in small quantities
- Can be very fast (when you find the)

Con:

- Find the right fit in algorithm and partitioning

Cloud

Pro:

- Endless CPU power
- Widely used in similar application → ALEXA/SIRI/…

Con:

- Initial cost of hard- and software development
- No high speed interface into the cloud

???, what did I forget

3. How to plan and design an „embedded vision“ system - Sensor and data rates around the application

Sensor

- Size, speed, pixel type, sensor quality at given light and desired speed → **Ohne Licht sieht man mich nicht**
- Sensor interface with/without glue logic to application circuit. X86-64 are possible to directly interface to GEV and U3V cameras, which takes us bad to “embedded vision” as a standard PC application with an embedded PC
- Is the sensor single/multi-tap? Can the XYZ toolchain cope with multiple pixels coming in in on clock cycle? HLS/SDSoC/OpenGL → default NO (→ workaround from S2I ☺)

Data rates around the application, which are

- sensor → application
- is the application
  1. pure streaming → broadcast, no added bandwidth
  2. heaving image (pre)processing → IR camera, add 2N, N stages of preprocessing, times the incoming stream bandwidth to your local AXI memory bandwidth
  3. complex image analysis → add another 2x of incoming stream bandwidth to your AXI memory bandwidth and never start without SoC
  4. This estimated bandwidth should be ≤66% of the theoretical available AXI AND external DDRx bandwidth

3. How to plan and design an „embedded vision“ system - Develop, implement and test the algorithm for the application

Develop and test the algorithm for the application

- Never start developing a new algorithm inside an FPGA or SoC, the debug- and change management will be just too hard
- Think about how to connect your algorithms, so that they become testable and reusable mid/long term
- Think about how to configure your algorithms to fixed-point-float, so that they become adjustable and usable
- Maybe forget about FPGA and SoC for the first days and weeks and play with light, lens, PC, existing camera and a standard library to (pre)develop an algorithm. This will give you a better feeling which architecture to get prepared for

Implement and test the algorithm for the application

- Start as late as possible with an FPGA and SoC implementation, as any structural change on VHDL code is risky, expensive and painful
- Try to establish and keep a good verification setup to constantly compare the developed algorithm and the implemented algorithm and to keep them in sync
- This verification/simulation is not only used for testing but for profiling the algorithm code. Based on the profiling you can make the final decision if a part of the algorithm will be implemented in VHDL, HLS or C
- Keep the FPGA code open to register balancing technics, as this will enable easy timing closure

3. How to plan and design an „embedded vision“ system - Possible tool chains, tool vendors

MathLab

Pro:

- Is really very close to the requirement and technics of page 18/19

Con:

- Expensive MathLab/SimuLink/VHDL-coder 30kUS$
- Needs constant work with it, as the tool is not straight forward use

Vivado SDSoC Edition

Pro:

- Reasonable cost
- Reasonable results with some → lot thinking compared to MathLab

Con:

- SDSoC not ready for the public use. Expect extra time and cost for e.g. PLC2 training
- Complex SoC system simulation with C- and VHDL-code not really supported in ISim → Expect extra time and cost for ≥ModelSim PE (3kUS$) for test benches in SystemC/SystemVerilog

3. How to plan and design an „embedded vision“ system - Possible tool chains, planning/development

Developer

Pro:

- Very interesting work

Con:

- Be prepared to either work with many tools – or work in a bigger team with one colleague per tool communication skill needed
- Be prepared to some frustration, as setting a an “embedded vision” system is a complex and high potential failure rate task

Management

Pro:

- “embedded vision” is easy to sell, as it is delivering only a dedicated function
- typical higher sales profit compared to standard image processing system due to less support cost

Con:

- added time/cost is needed due to higher integration and risk involved with “embedded vision” products
- added patience is need in the team, as they have to learn skills of light/lens/…/HLS/embedded programming, which might have been never used before (, but e.g. Sensor to Image can help ☺)

Thanks for your interest and time!

Sensor to Image GmbH
Lechtorstrasse 20
D 86956 Schongau

Website: www.sensor-to-image.de
Email: email@sensor-to-image.de

Tel.: +49 8861 2369 0
Fax : +49 8861 2369 69
EU-VAT-ID : DE-812693714
Register Court ID Munich: HRB125437