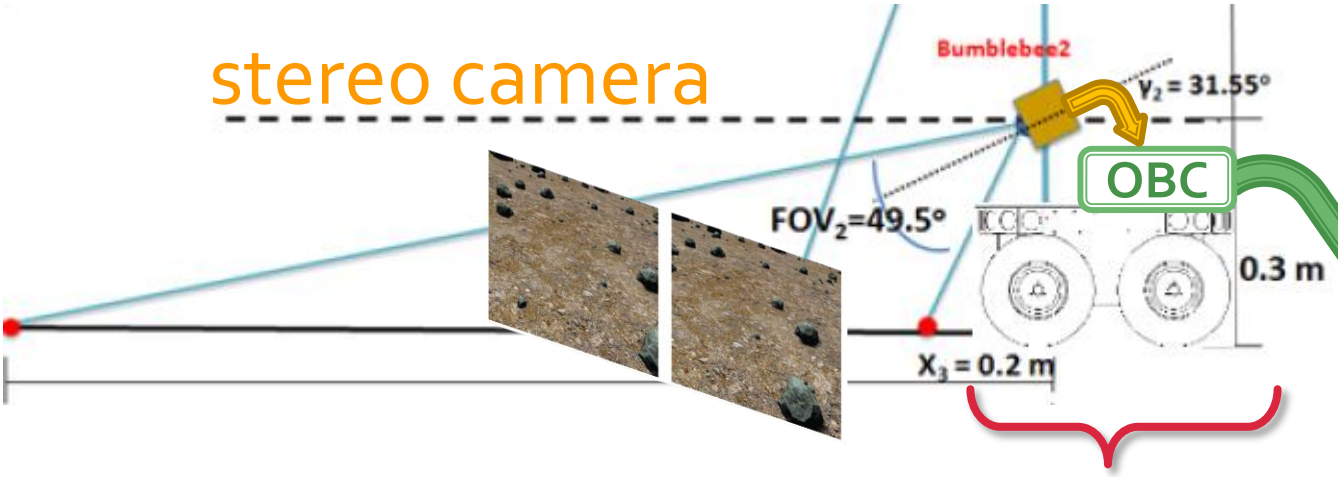
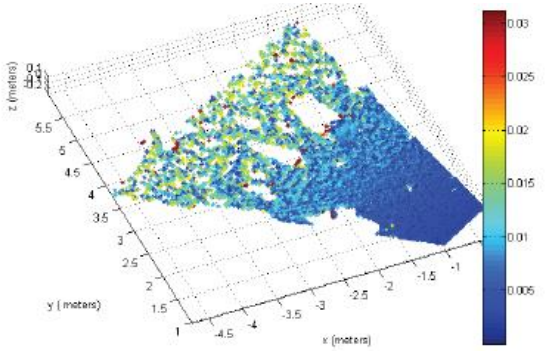
FPGA acceleration of computer vision and optimization for European space applications

G. Lentaris, (glentaris@microlab.ntua.gr)
K. Maragos,
J. Stamoulias,
D. Diamantopoulos,
K. Siozios,
D. Soudris,
M. Lourakis,
X. Zabulis,
M. Aviles Rodrigalvarez

ECE, NTUA, GREECE
FORTH, CRETE, GREECE
GMV, MADRID, SPAIN

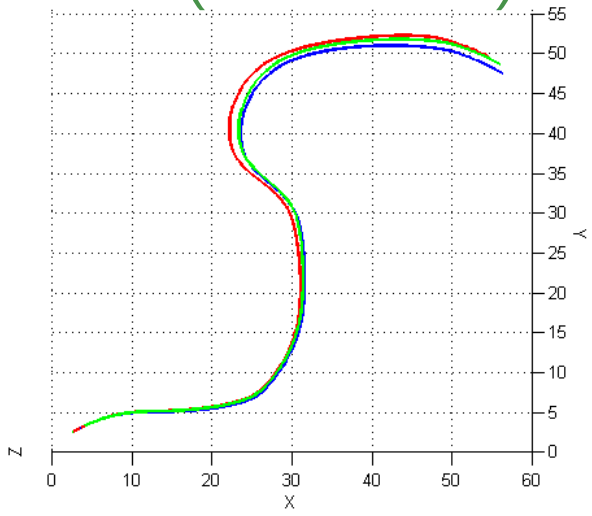
Computer Vision in Space





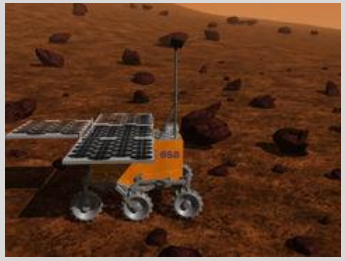
3D mapping

rover position (localization)



MER Rovers (2003)

Curiosity (2012)



ESA ExoMars (2018)

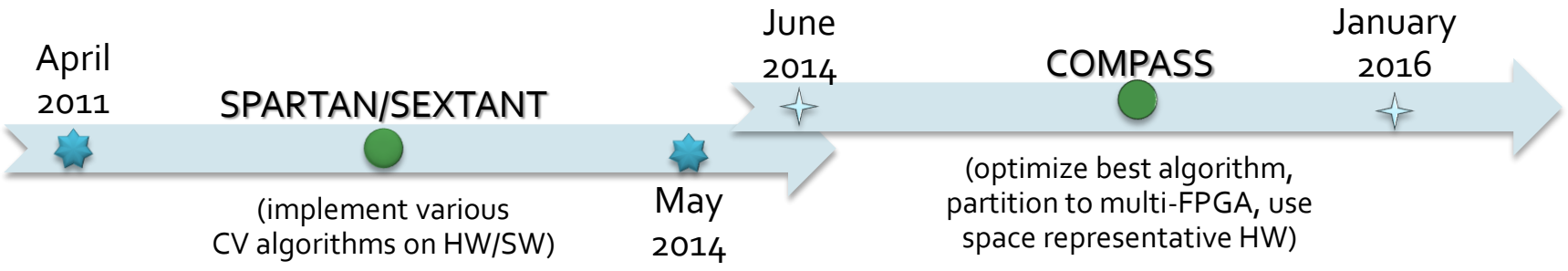
Computer Vision, the Challenge

- high complexity, especially to get high accuracy
- LEON-FT has relatively small processing power
  - slow execution, impractical for advanced algorithms
    - 4 hours for mapping on 150MIPS (4Mpixel, 300 depths)
    - 1 minute for localization on 150MIPS (robust, error < 2%)
  - ESA goal: only 20 sec and 1 sec, respectively
    - looking for speed-up factors 10x to 1000x
- proposed solution (SPARTAN/SEXTANT): **FPGA**
  - develop & accelerate & compare various algorithms
  - very promising results, met ESA specs [SEFUW 2014]

with Space Representative HW

COMPASS project (ESA)

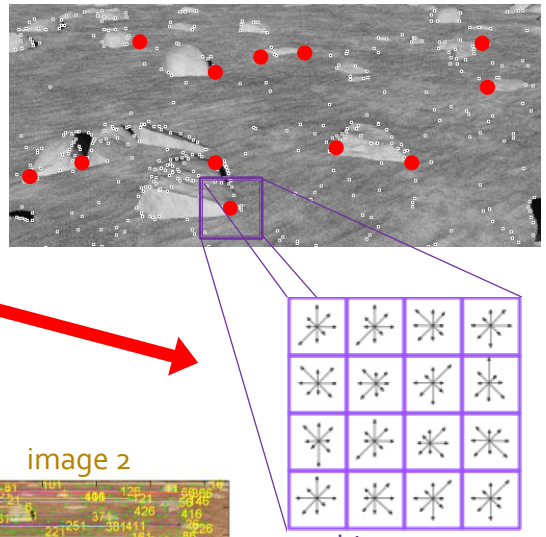
- optimize best HW/SW algorithm of **SPARTAN/SEXTANT**
- target European space HW, consider multi-FPGA
  - CPU: *LEON3 + RTEMS* (scale time to 150 MIPS)
  - FPGA: *BRAVE* (use HAPS-54, limit the FPGA resources)
  - ✓ delivered highly optimized & proof-of-concept designs



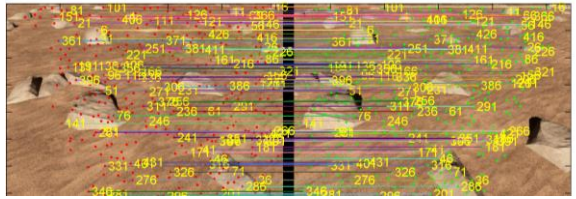
COMPASS: Selected Algorithm

LOCALIZATION:

1. Feature Detection: Harris (x2)
2. Feature Description: SIFT (x2)
3. Matching: x^2-distance (x2)
4. Filter outlier matches
5. Motion Estimation



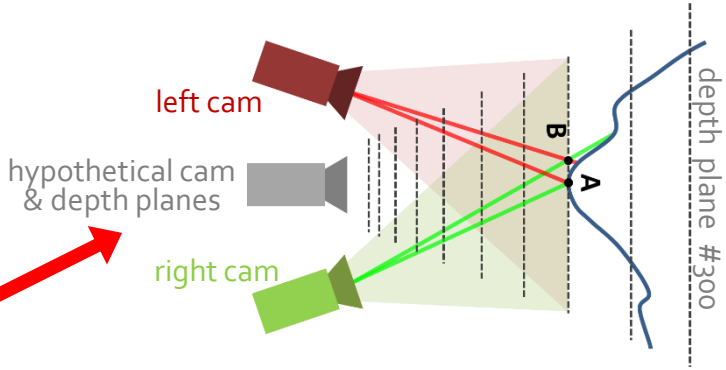
histogram of gradients



matching of histograms

MAPPING:

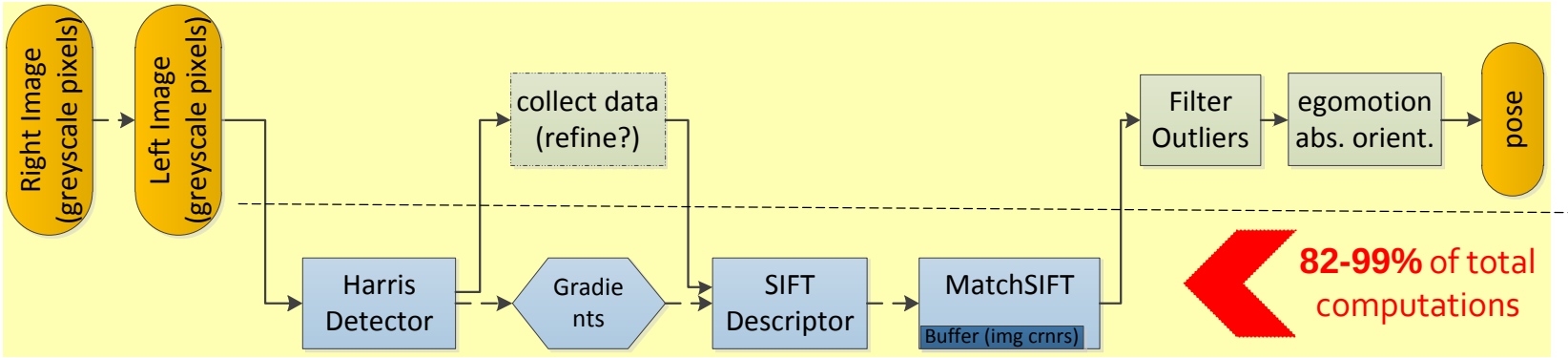
brute-force examine 300 depth planes: SpaceSweep



COMPASS: HW/SW Partitioning

- partitioning & co-design based on methodology
  - profile on LEON3 (time, memory, comm., arithmetic)
  - select kernels, develop VHDL by hand, optimize

LOCALIZATION



MAPPING

- 99.9% FPGA (except a final floating-point transform)

COMPASS: Architecture Overview

- **Target low-cost implementations** (on FPGA)
  - especially w.r.t. memory: bottleneck for CV on FPGA
    - resource reuse: decompose input data, process successively
- **Target sufficient speed-up** (for ESA specs)
  - pipelining on pixel-basis & systolic architectures
    - burst read of image, transform on-the-fly (1 datum/cycle)
  - parallel memories & parallel processing elements
    - support parallel calculation of arithmetic formulas
- **Target configurability** (tuning, adaptation)
  - parametric VHDL: data size, accuracy, parallelization,

COMPASS: Extra Optimizations

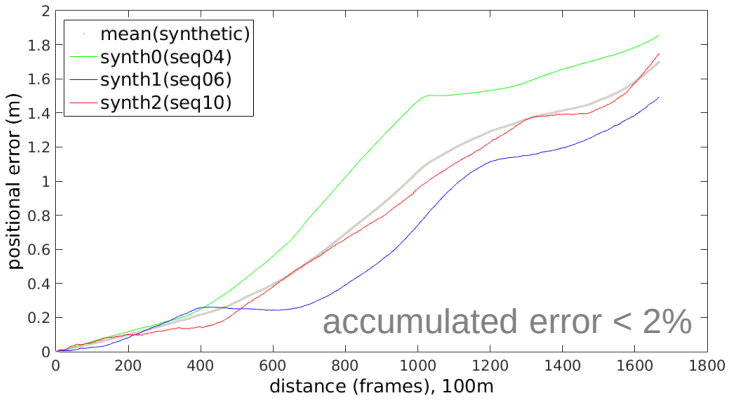
- **Target <u>even lower</u> cost implementations**
  1) customize algorithm exactly for the ESA scenario
     - discard functionality that is rarely/never used (cost on HW!)
     - word-length optimization, adapt to datasets & rover setup
  2) share resources among algorithms (merge in 1 unit)
     - more tight collaboration among HW modules/developers
     - sacrifice generality and ease of development/debugging
  3) design space exploration & platform customization
     - exploit parametric VHDL, compromise cost/accuracy
     - adapt to underlying FPGA platform (e.g., use SLICEM)
- 12 optimizations on FPGA, 2 major on CPU
  - 5 more implemented, tested, rejected due to low accuracy

COMPASS: Optimization Results

- **Initial FPGA Cost:** almost two *Virtex6 VLX240t*
  - mapping (14% slices, 57% mem) + localiz. (57% slices, 78% mem)
- **Total Optimization:**
  - *LUTs*: **-25%** ← low priority (low cost, increases error)
  - *DFFs*: **-35%** ← main reason: optimized word-lengths
  - *DSPs*: **-41%** ← : fixed-point & adaptation to scenario
  - *RAM*: **-51%** ← : RAM sharing & word-lengths & adapt
  - *Time*: **-41%** (HW), **-90%** (SW) ← : reschedule & adapt
  - at component level: up to **-57%** logic and **-79%** RAM

Fit in a Single Space-Grade FPGA

- consequently, entire design fitted in *Xilinx-5QV* !
  - Mapping & Localization & Ethernet (or Spacewire)
  - utilization: 63% LUTs, 51% DFF, 42% DSP, 98% RAMB
- design meets initial ESA specs, no compromise
  - localization: 1-2 fps, 1.7% error (512x384 images, 100m paths)
  - mapping: 17.4 sec, 2cm error (1120x1120x3 images, 4m depth)



97% coverage, 7mm RMS error,
3.6% mistakes (>2cm error)

Feasible with European HW

- BRAVE (vs. Virtex-5QV): similar logic, half memory
  - cannot fit entire COMPASS VHDL
- solution1: dynamic reconfiguration
  - e.g., like (or during) memory scrubbing, interchange between mapping & localization (every 1-2 minutes)
  - use 1 big-BRAVE, need further study on actual device
- solution2: multi-FPGA
  - assume 2 or 3, medium- or hypothetical-big-, BRAVEs
  - perform HW/HW partitioning and sync optimization

COMPASS: multi-FPGA Design

- platform: HAPS-54 (4 xc5vlx330)
  - 65nm (as BRAVE), bigger size
  - limit utilization, emulate BRAVE

- refined methodology for partitioning & sync
  - best partitioning approach: semi-automatic
    - *manually*: select partitions, map FPGAs, refine connections
    - *auto*: evaluate cost, VHDL wrappers, pin assignment (LOCs)
  - high-speed sync: improve results in progressive steps
    - regarding clock skew, I/O registers, platform characteristics

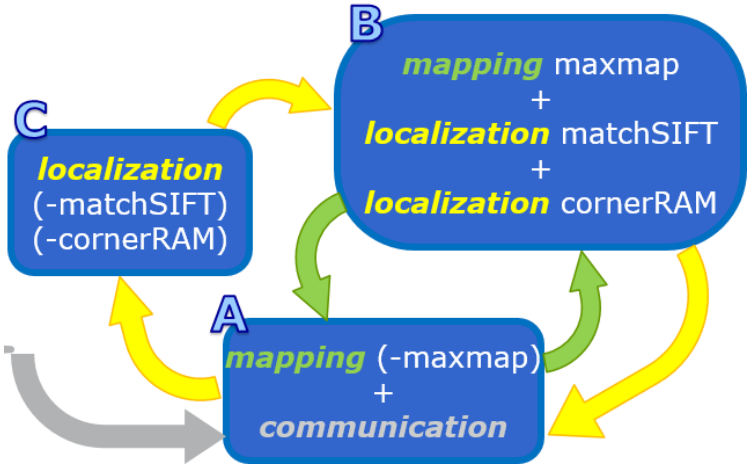
COMPASS: multi-FPGA Results

- 2 proof-of-concept demos (more studied/tested)
- equivalent to single-FPGA (accuracy + speed)
  - triple-FPGA: 1 big-BRAVE + 2 medium-BRAVE
  - double-FPGA: 2 big-BRAVE

BRAVE	LUT	DSP	RAMB18
A (med)	50%	54%	~99%
B (big)	17%	5%	~99%
C (med)	75%	~99%	92%

BRAVE	LUT	DSP	RAMB18
A (big)	9%	13%	97%
B (big)	32%	32%	98%

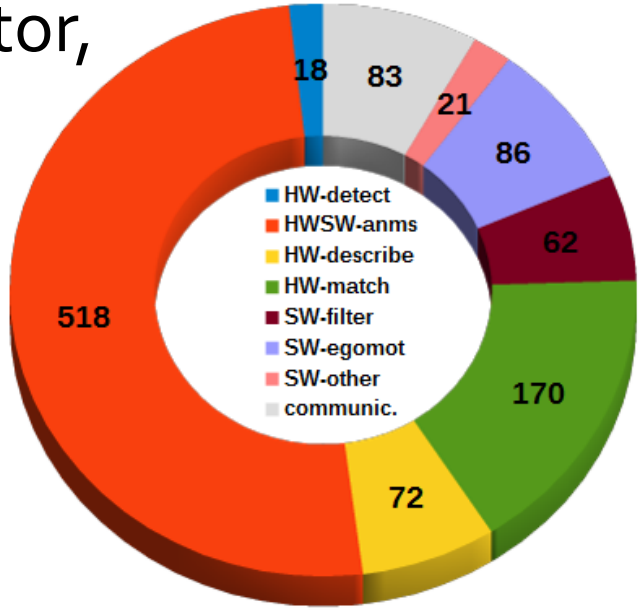
on board: 90 – 400 traces, 20Mbps – 6Gbps



System Timing and Acceleration

- LEON(150MIPS): 4 hours mapping, 35 sec localiz.
- LEON+BRAVE: 17 sec mapping, ½-1 sec localiz.
- system speedup: **796x** mapping, **34-56x** localiz.
  - *kernels*: 1257x sweep, 158x detector, 257x descriptor, 17-70x matching

  - FPGA:CPU computation ≈ 99:1
  - FPGA:CPU time ≈ 2:1
  - comput.:comm. time ≈ 5:1
  - device utilization ≈ 20% – 70%



time analysis (msec), total=1030msec
150MIPS LEON + HAPS@136MHz

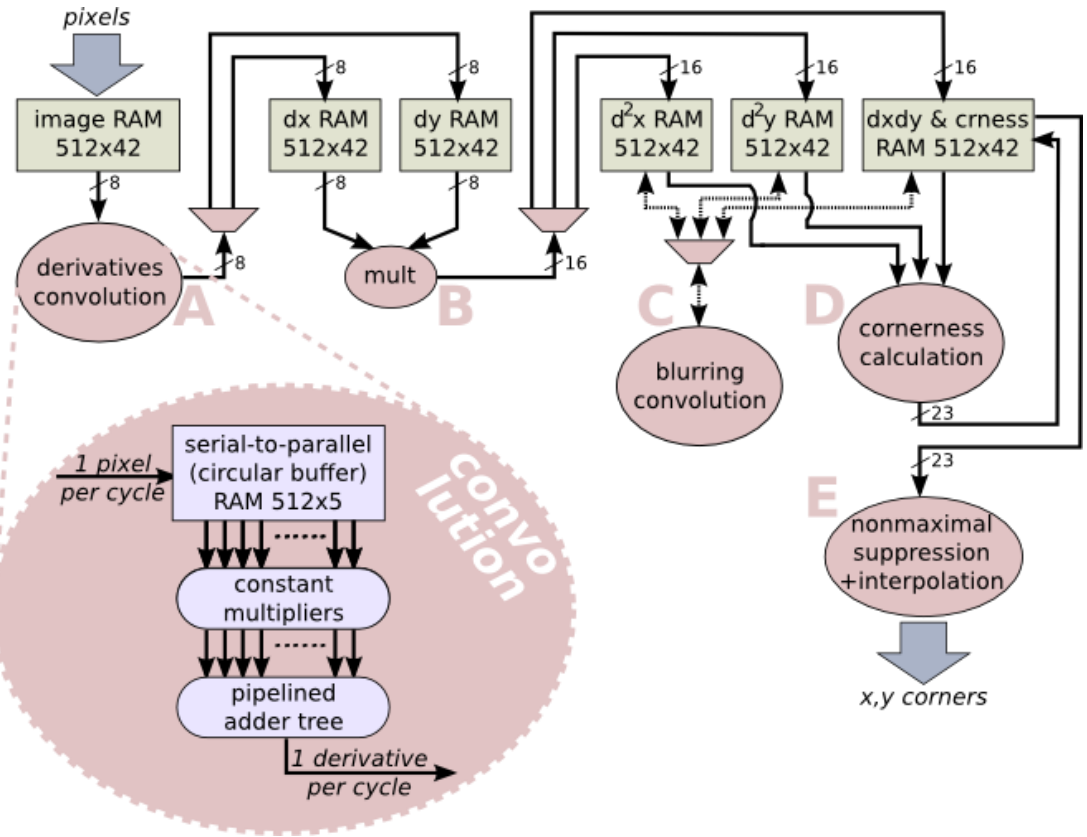
Conclusions

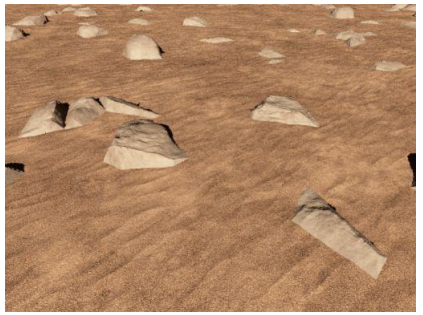
- modern/advanced vision algorithms too slow on space-grade CPUs (not only LEON @ 150 MIPS)
  - solution: space-grade FPGA (10x-1000x acceleration)
- VHDL by hand & optimization/customization to application had big cost decease (roughly, half)
  - rover's CV in 1 Xilinx-5QV, very high accuracy/speed
- feasible with the new European FPGA: BRAVE
  - multi-FPGA, or 1-FPGA (dynamic reconfiguration)

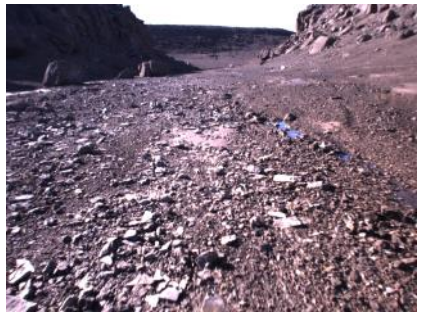
Thank You!
Questions?

George Lentaris, ECE, NTUA, Greece (glentaris@microlab.ntua.gr)

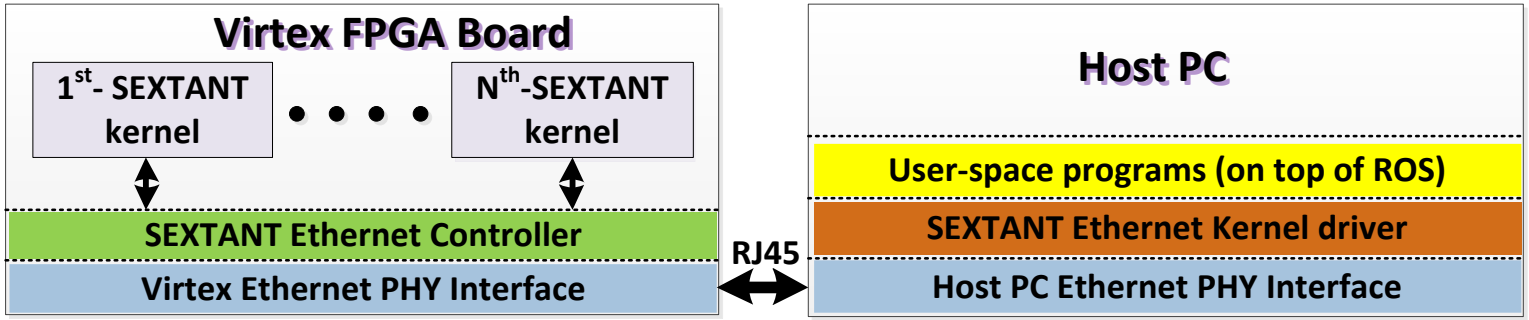
BACKUP1

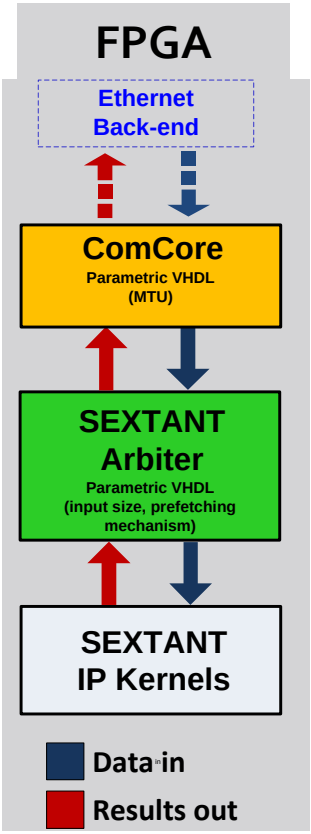

pixels
image RAM 512x42
dx RAM 512x42
dy RAM 512x42
d²x RAM 512x42
d²y RAM 512x42
dxdy & crness RAM 512x42
derivatives convolution
A
mult
B
C
blurring convolution
D
cornerness calculation
E
nonmaximal suppression +interpolation
x,y corners
1 pixel per cycle
serial-to-parallel (circular buffer) RAM 512x5
constant multipliers
pipelined adder tree
1 derivative per cycle
convolution





BACKUP2





- custom scheme with **raw Ethernet**
  - on CPU: developed kernel driver
    - LKM, Rx-Tx SysCalls at Network layer, C++ API
  - on FPGA: developed data-flow controller
    - low-level functions of Link-layer by "Eth. MAC IP" from OpenCores (CSMA/CD LAN IEEE 802.3)
    - custom: packets, handshake, backoff, arbitration