JÜLICH
FORSCHUNGSZENTRUM

Heterogeneous Systems in Computing Intensive Applications: FPGA – GPU complexes

Central Institute for Engineering, Electronics and Analytics

Dr. Sergey Suslov

Outline

Talk covers:

- Premise for Heterogeneous System Employment
- Peculiarity of Heterogeneous System Design
- Compact High Performance Computing with FPGAs and GPUs (Project Examples)
- FPGA vs GPU: comparing technologies

Premise

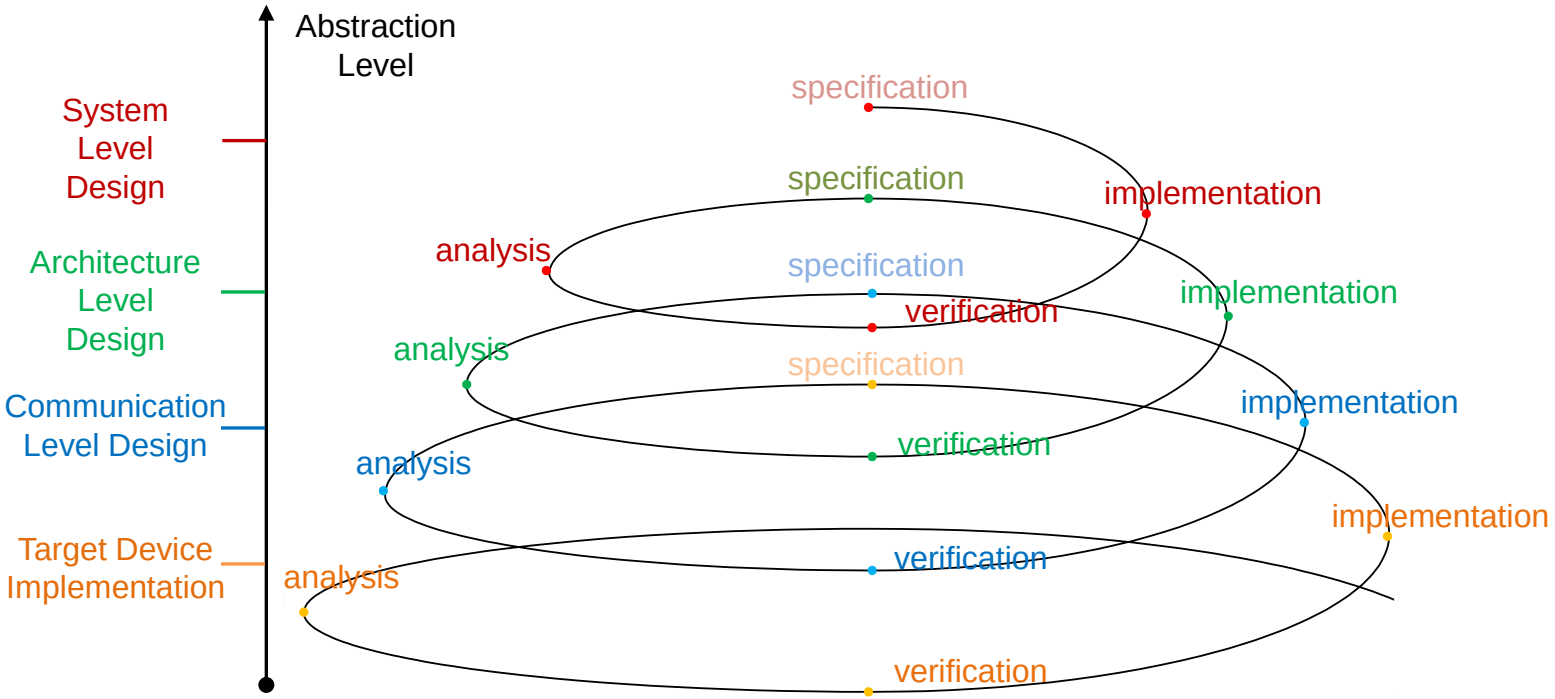
Status of Technology

- improvements in computer performance:
  - less frequency growth but more computing parallelism
- many calculation intensive applications have intrinsic parallelism
  - benefit from parallel platforms
- increase in IC integration level enables High Performance Computing in workstations (Compact HPC)
- three main approaches:
  - symmetric multiprocessor (SMP)
  - configurable logic – Field-Programmable Gate Array (FPGA)
  - stream processing architecture – Graphics Processing Unit (GPU)
- trends:
  - Accelerated Processing Units (APU): multi-core CPU + GPU + shared address space
  - System on a Programmable Chip (SoPC): multi-core CPU + configurable logic FPGA

Peculiarities of Design

Methodology:

- top-down design approach
- early partitioning
- gradual refinement
- cyclic: specification → implementation → verification → analysis



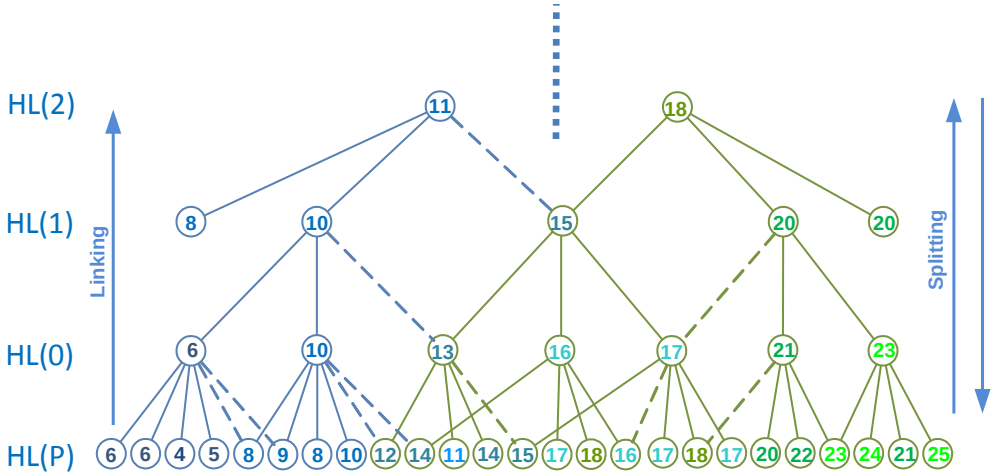
Compact High Performance Computing with FPGAs and GPUs

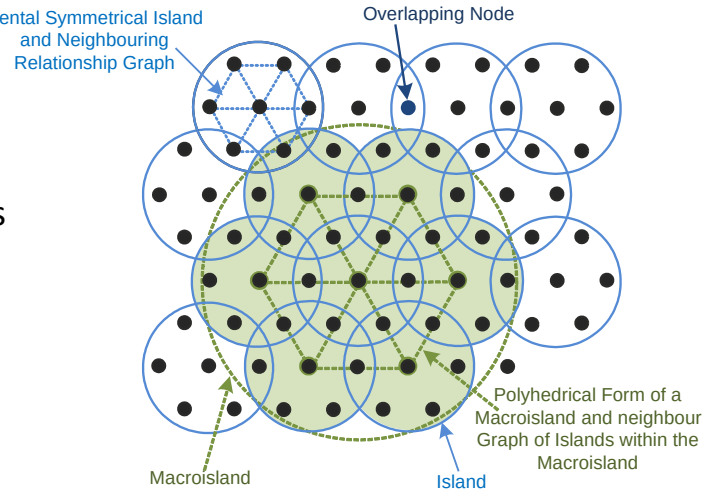
Example Projects:

- Gray Scale Code (GSC) segmentation (FPGA vs GPU)
- Cone-Beam Computer Tomography reconstruction (GPU only)
- Magnetoencephalography (FPGA & GPU)

Fundamentals of the GSC method

- hierarchical (multi-resolution) region-growing approach
- topology: hexagonal lattice
- region (R) forming criteria:
  - area
  - homogeneity of a feature (F)
  - spatial neighbourhood
- phases:
  - coding: (island$_{HL(P)}$; F(P); Neighbouring Graph) → $R_{HL(0)}$
  - linking: (island$_{HL(n)}$; F($R_{HL(n)}$); Overlapping) → $R_{HL(n+1)}$, branches
  - splitting: generate disjoint segments (S) in region forest
  - result generation: F(S) → F(P)

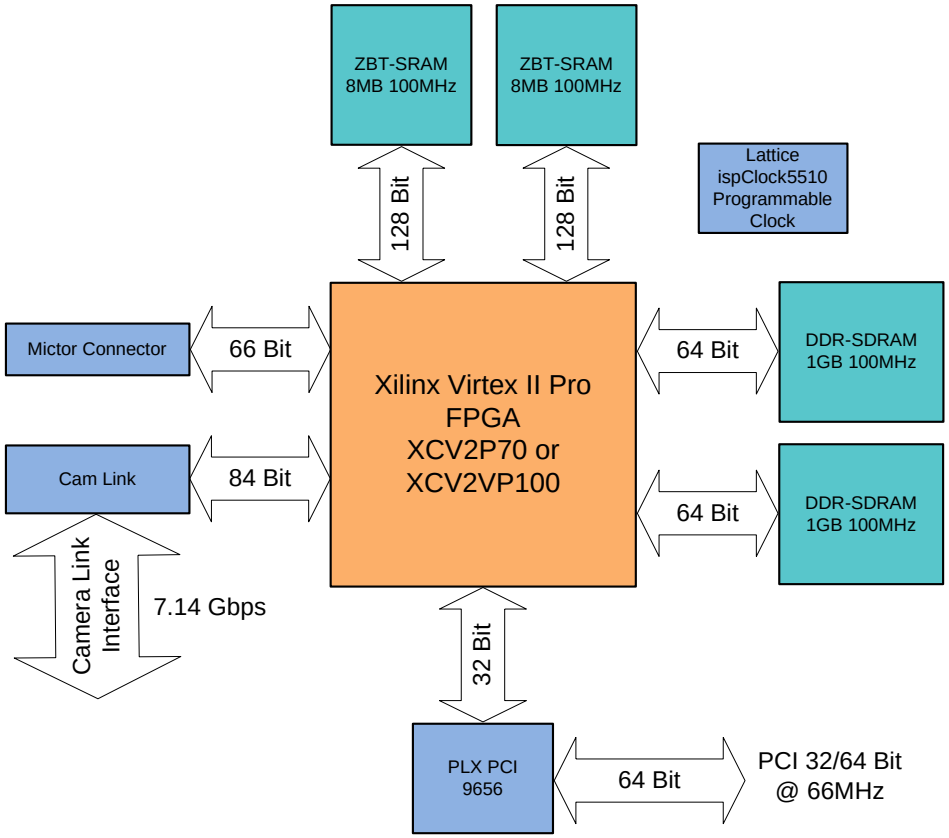




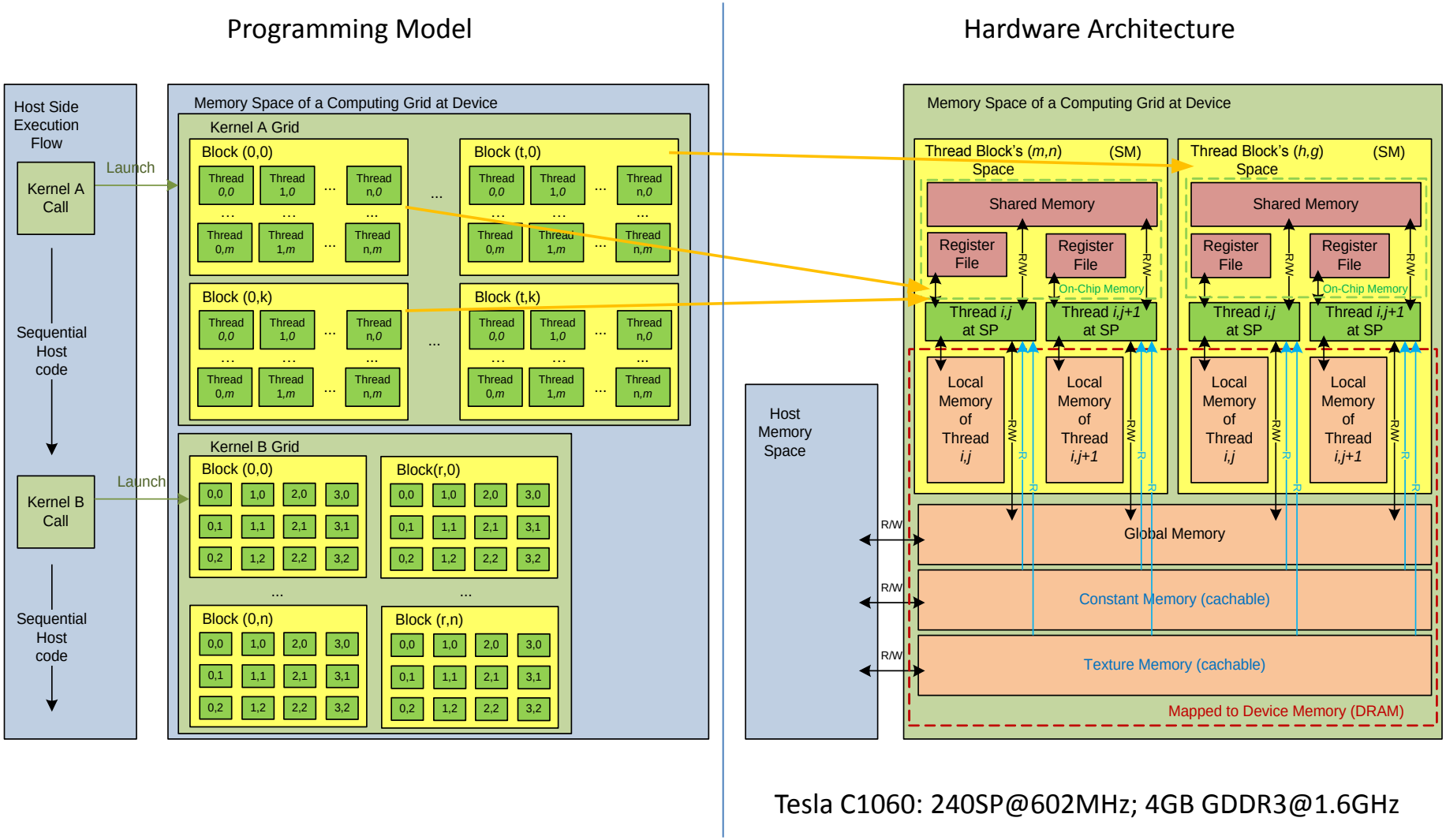
Implementation Platform FPGA

FPGA Expansion Board (Virtex II Pro)

- configurable logic (99,216 LCs @100Mhz)
- two DDR SDRAM banks (1GB@100MHz each)
- two ZBT SRAM banks (8MB@100MHz each)



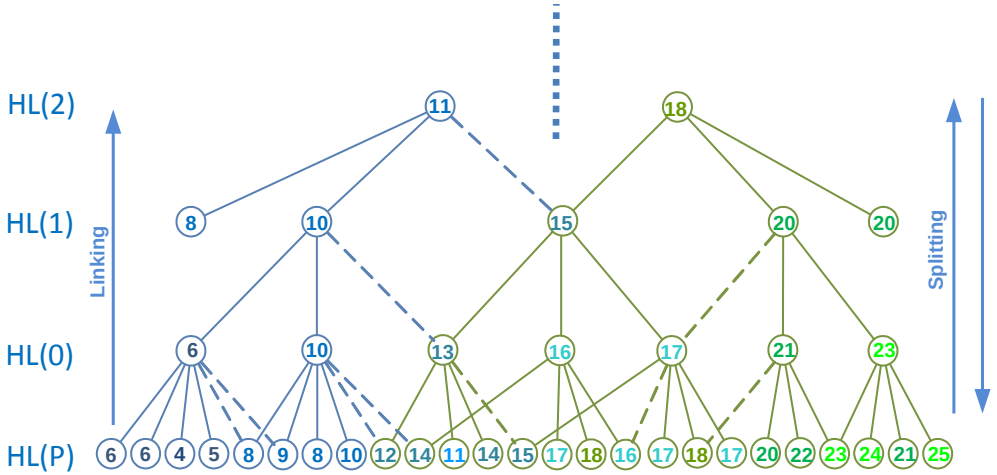
Implementation Platform GPU

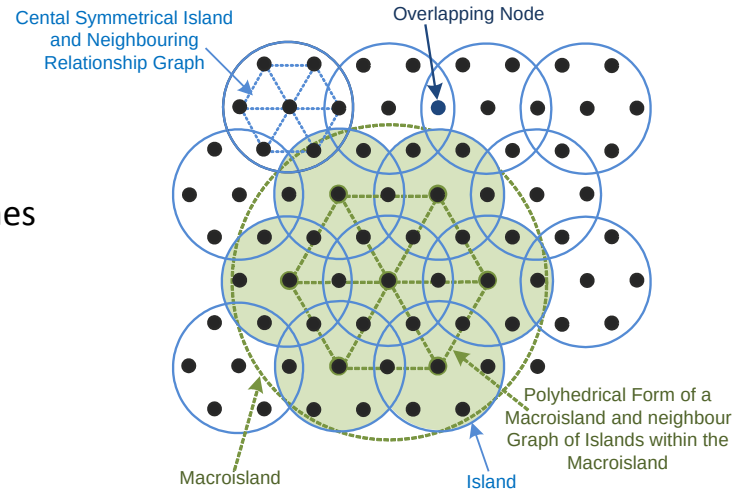


Tesla C1060: 240SP@602MHz; 4GB GDDR3@1.6GHz

Fundamentals of the GSC method

- hierarchical (multi-resolution) region-growing approach
- topology: hexagonal lattice
- region (R) forming criteria:
  - area
  - homogeneity of a feature (F)
  - spatial neighbourhood
- phases:
  - coding: (island$_{HL(P)}$; F(P); Neighbouring Graph) → R$_{HL(0)}$
  - linking: (island$_{HL(n)}$; F(R$_{HL(n)}$); Overlapping) → R$_{HL(n+1)}$, branches
  - splitting: generate disjoint segments (S) in region forest
  - result generation: F(S) → F(P)





GSC Implemented Models

Methodology:

- top-down design approach
- gradual refinement
- cyclic: specification → implementation → verification → analysis

Main GSC models:

Common

- functional executable model
- static data-flow model

FPGA specific

- static data-traffic model
- architectural executable model
- pre-implementation resource model
- interface and communicational model
- RTL synthesizable model

GPU specific

- SIMD functional model
- CUDA C program

GSC GPU implementation

GPU Optimisation strategy

- Global memory optimisation
  - addressing scheme
  - data structure layout
- Memory architecture peculiarity exploitation
  - texture memory
  - constant memory
- Shared memory layout
  - addressing scheme
  - data volume optimization
  - memory reuse
- Kernel control-flow optimisation
  - execution path optimisation
  - branching minimisation
- Operation level optimisation
  - intrinsic GPU instructions
  - specialised math
- Block size parameters selection

GSC Results

Performance

- 2048^2: 85 ms (FPGA), 99 ms (GPU) vs. 1985 ms (CPU)
- achieved acceleration endorses the use of the parallel GSC in real-time and interactive applications
- even older generation FPGA beat newer GPU processors

Efforts

- analysis of the method
  - extreme important
  - discover the parallelization potential of the algorithm
- development cycle is significantly longer for FPGA

CT Reconstruction

in collaboration with Central Institute for Technology (ZAT, F. Pauly)

The Micro-Focus CT Equipment

The micro-focus CT equipment (Figure 1) in the Zentral Institute for Technology (ZAT) of the Research Center Jülich consist of the following components:

Source:	VISCOM XT 9225, DED, 225 kV / 3 mA / 320 W, focal spot: 3-7 µm
Detektor A:	PerkinElmer XRD 512-400 AI1 LFS, 205x205 mm, 512x512 pixel, 400 µm
Detektor B:	PerkinElmer XRD 1621 AN4, 410x410 mm, 2048x2048 pixel, 200 µm
Gantry:	8 axis manipulator, Sauerwein MPS, Stapf Electronic PR 1-6, 1 µm precision

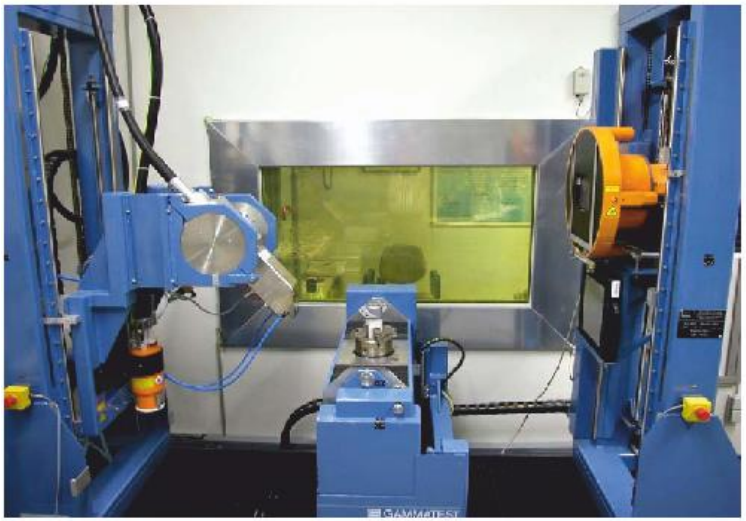


Figure 1: The Micro-Focus CT Setup

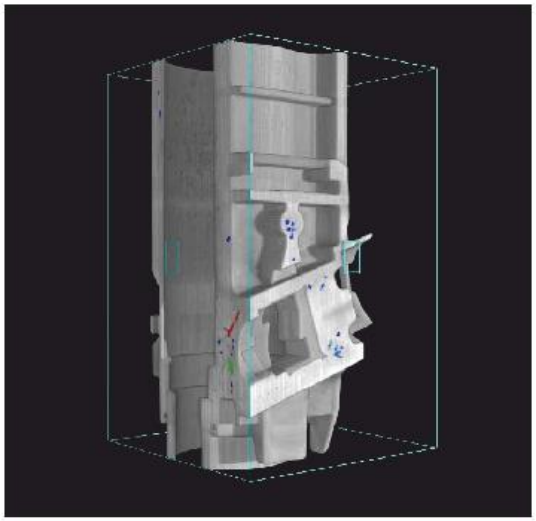


Figure 2: ADR of an Mg Casting

CT Reconstruction

Applications:

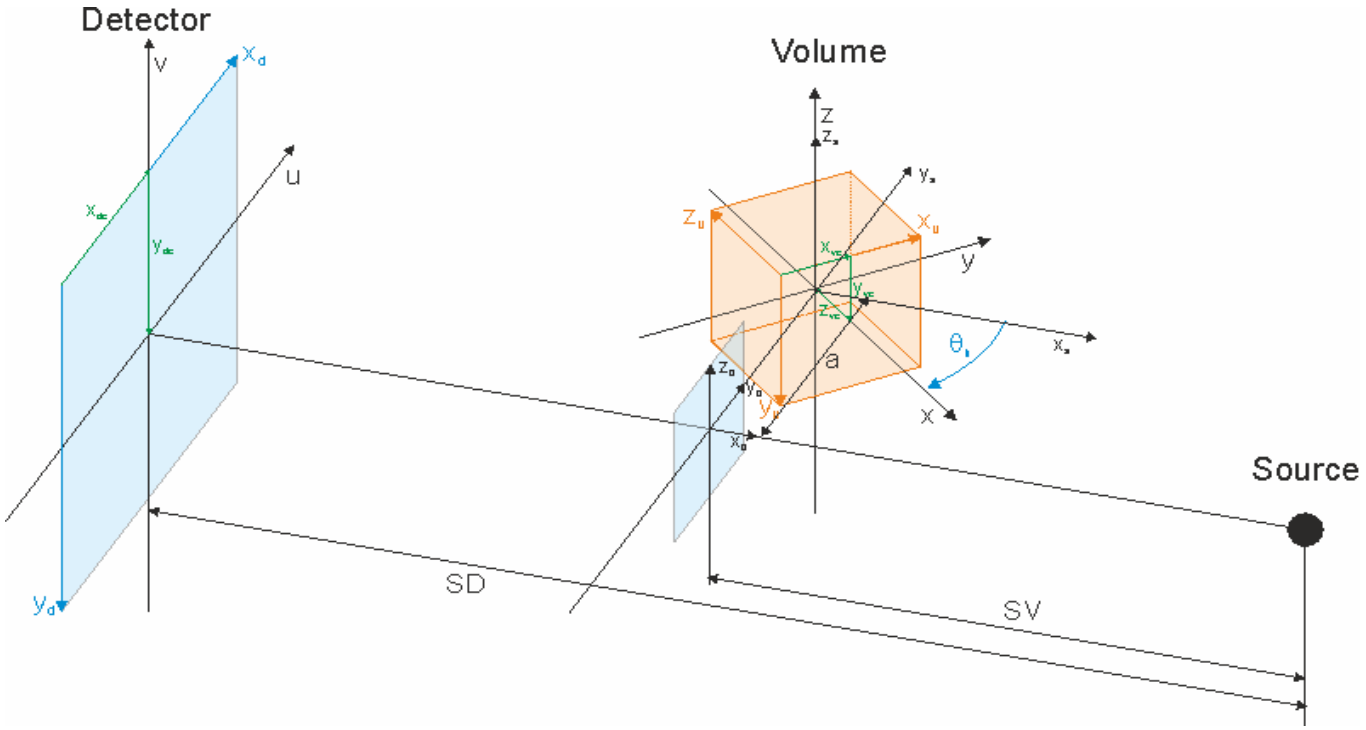
Applications in focus comprise the needs of internal and external custumors among others in the following areas:

- prefabricated parts of different materials (metal, synthetic material, etc.)
- sandwich materials (concrete, etc.)
- soil assays
- plants

For some applications newly developed algorithms based on the GSC segmentation allow the detection of defects in the material. The example in Figure 2 shows the automated defect recognition of a Mg casting. Red areas indicate a big defect volume, blue areas a low one.

CT Reconstruction

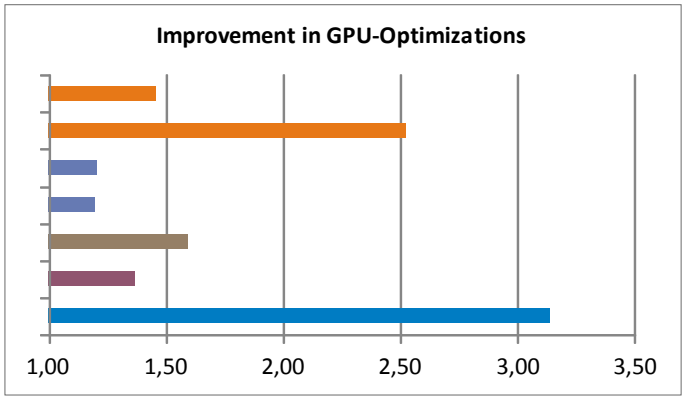
Geometric model:



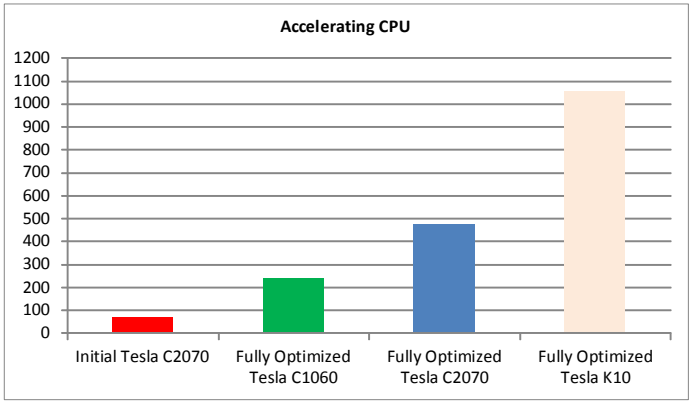
For the reconstruction of the volume (8G voxels in floating point accuraccy = 32 Gbyte dataset)the well-known Feldkamp (FDK) algorithm is used slightly modified for appearing detector parallel shifts of the rotation axis (a in Figure 4).Due the available memory on the GPU boards, the volume must be divided into volume parts, each computed seperately.

CT Reconstruction

Optimization results:



- Projection processing outside the reconstruction loop →intermediate storage on disk
- Projections in the textur memory →exploitation of HW-Interpolation
- Higher Texture Hit by appropriate Block configuration →prefer z-dimension
- Projection constant variables in constant memory →avoiding Local Memory
- Constants' pre-calculation on host parallel to kernel execution
- Hiding file I/O and host→device date transfer by asynchronous kernel execution
- Arithm. pre-computation for y-invariant terms on host
- One thread processes a column section →shared memory usage



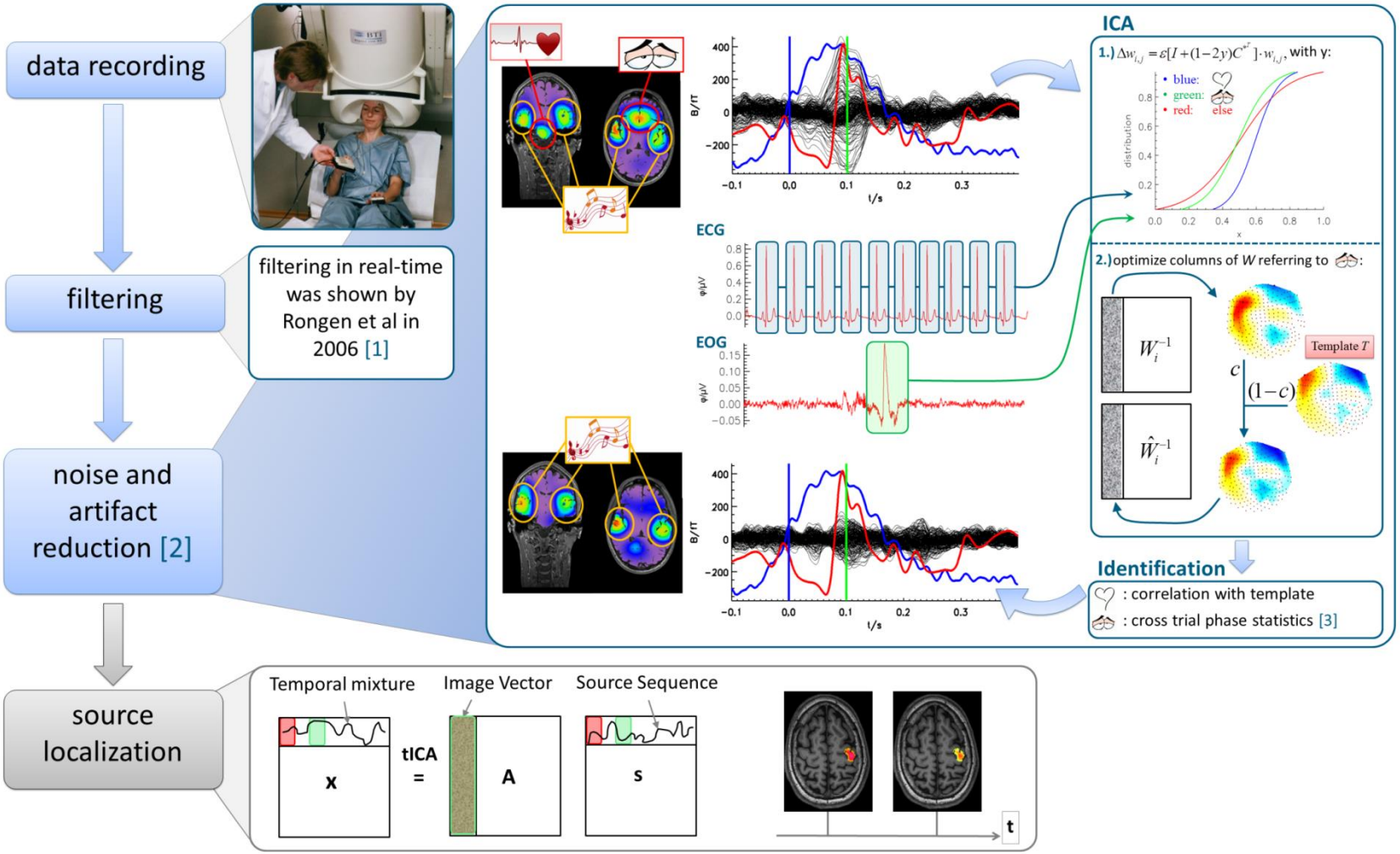
2048^3 Volume of 1800 Projections:

- CPU: 207 Hours (8,63 Days)
- Tesla C2070, non-optimized: 3,15 Hours
- Tesla C1060, optimized: 1 Hour
- Tesla C2070, optimiert: 26 Minutes
- Tesla K10, optimiert: 12 Minutes

MEG Heterogeneous Complex

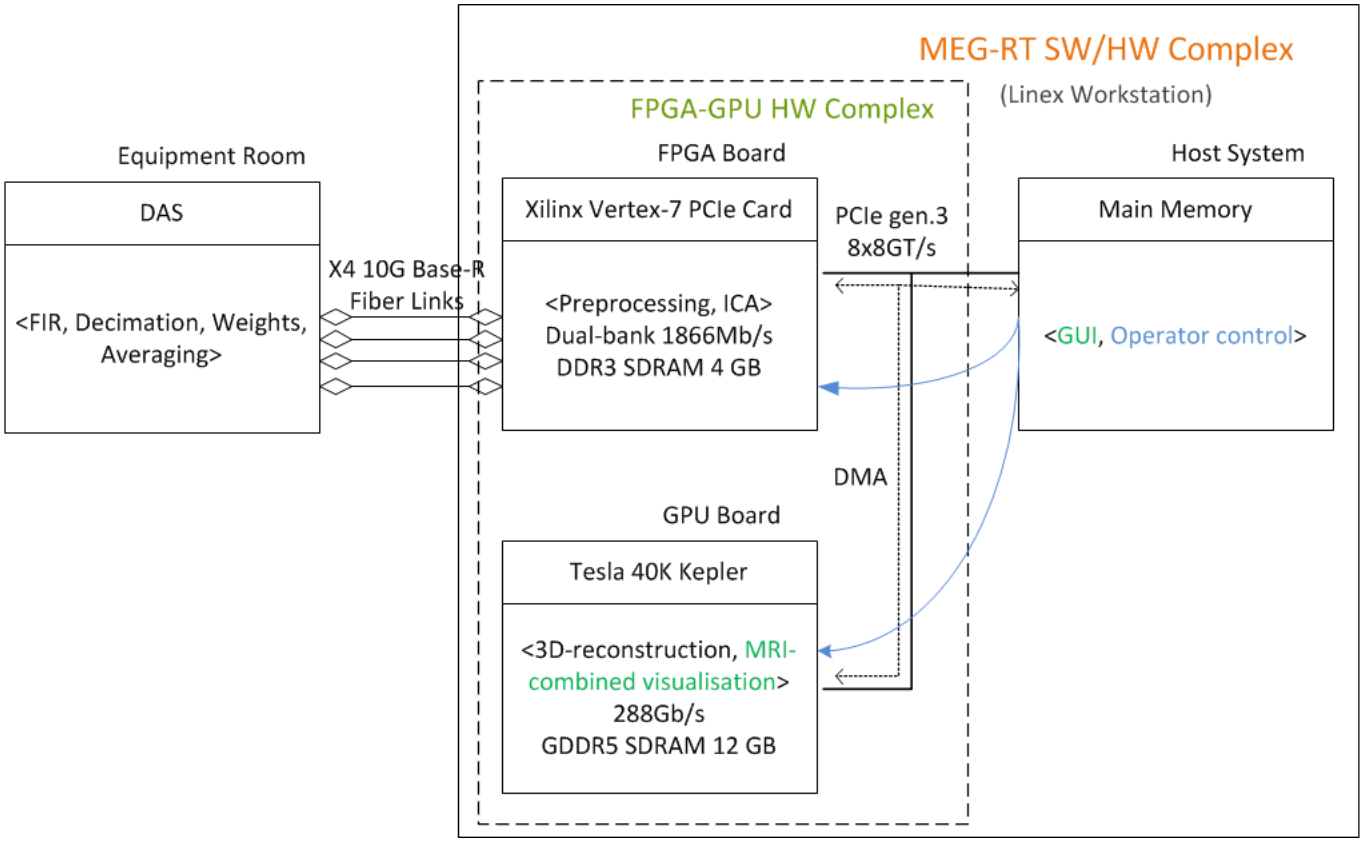
in collaboration with Institute of Neuroscience and Medicine (INM-4, Dr. J.Dammers)

Analysis pipeline:



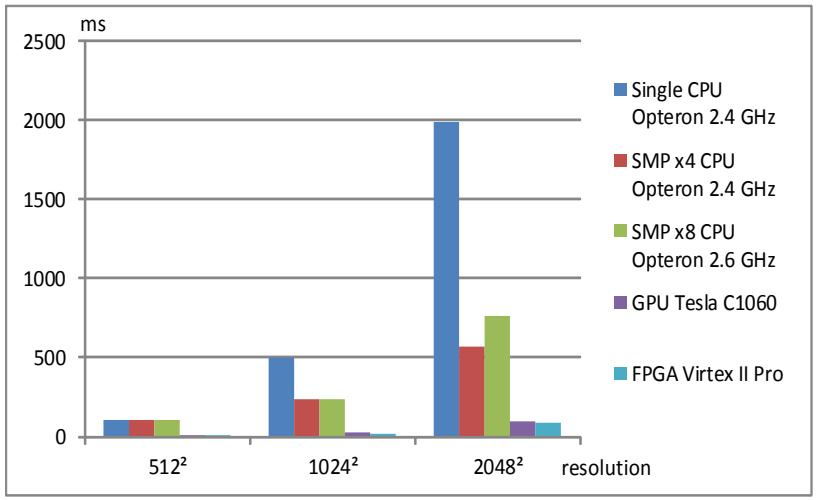
MEG Heterogeneous Complex

Complex block diagram:

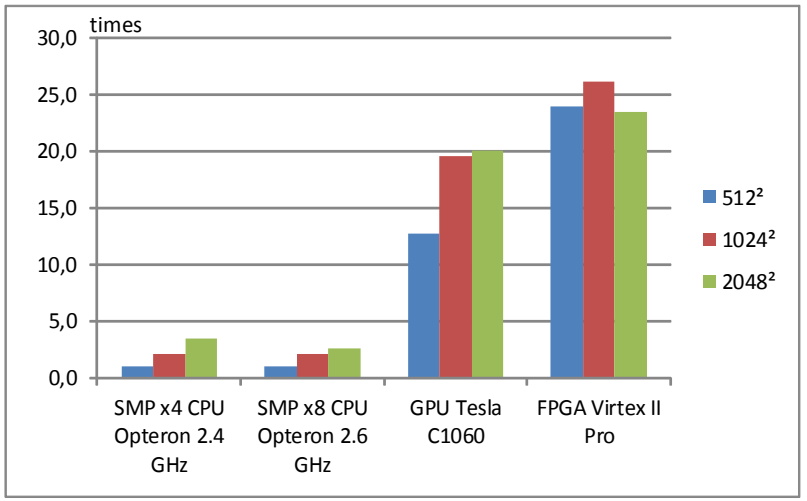


- Linux based Open Architecture HW/SW complex
- Python API to HW functions
- Single Workstation solution for Real-time operating
- Selectable HW tool set
- Dynamic HW tool chaining
- FPGA + GPU HW complex interaction with minimum Host interference

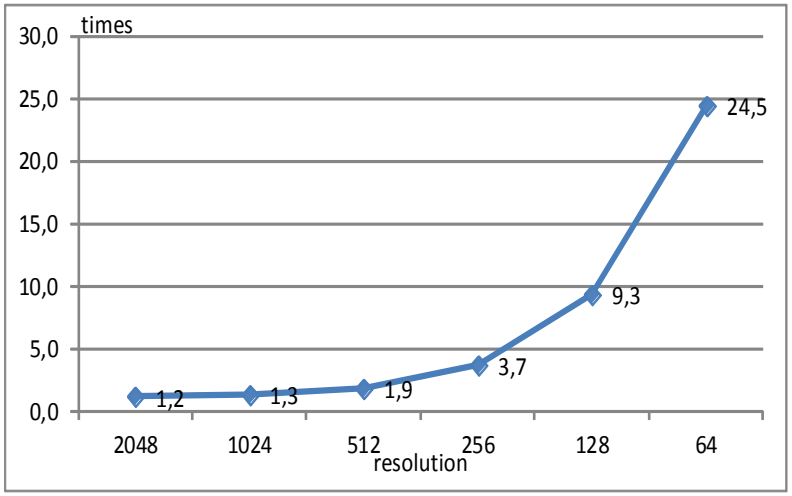
Comparison FPGA vs GPU: Performance



Elapse Time

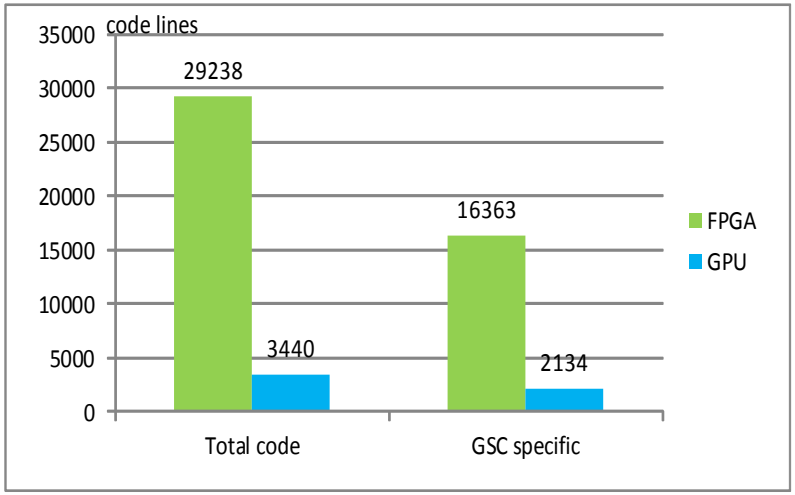


Performance Acceleration Ratios

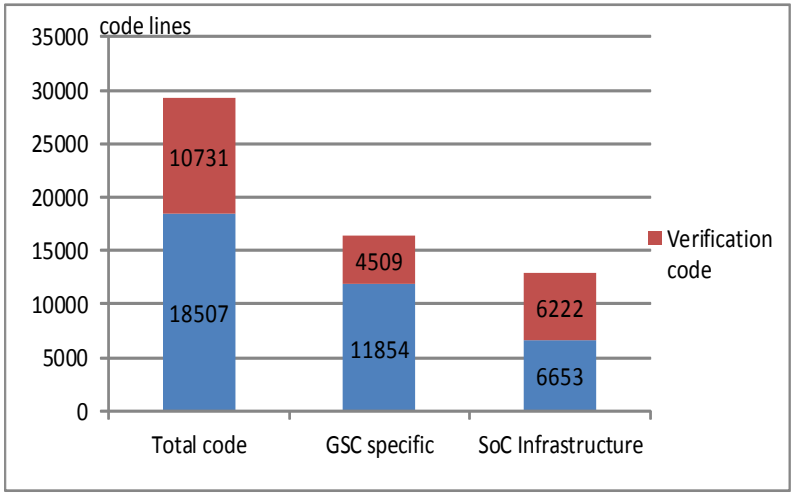


GPU to FPGA processing time ratios

Comparison FPGA vs GPU : Efforts



Efforts for FPGA and GPU solutions

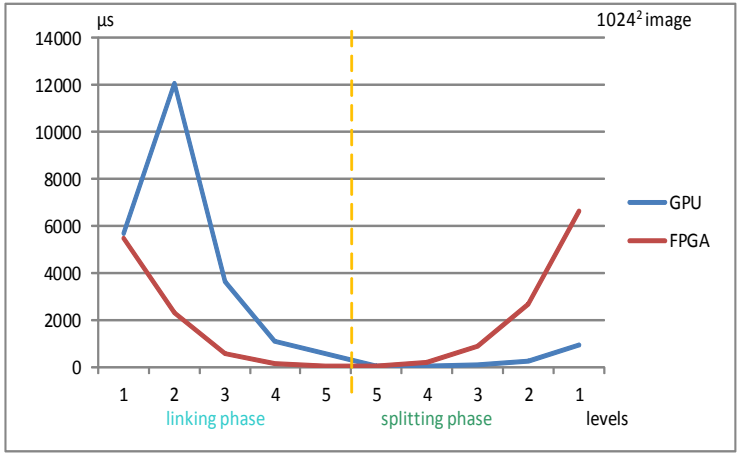


FPGA code structure

Conclusions (1)

Flexibility

- FPGA
  - freedom for the organization of the computation
  - application specific buffering and data scheduling schemes
    - more effective bandwidth utilization
    - more elastic to data volumes
  - more flexible in data organization due to fine grained parallelism
  - lack of complex hardwired functional blocks

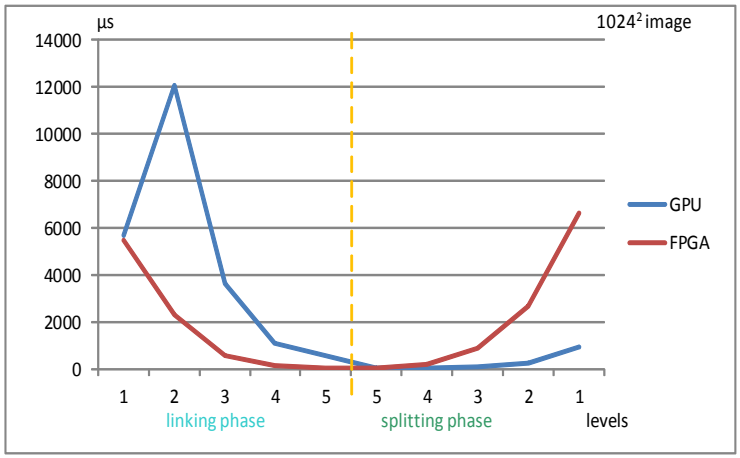


FPGA and GPU combined profile

Conclusions (2)

Flexibility

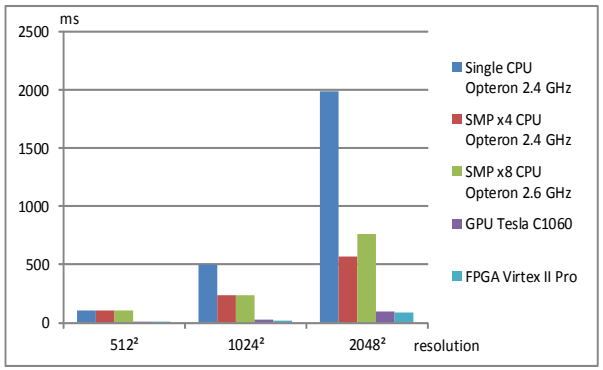
- GPU
  - adaptation of the algorithms to the target architecture
  - sensitive for complex code execution flow
    $\Rightarrow$ branching minimization
  - effective for huge amount of data
  - ISA imposes restrictions on data compaction
  - sensitive to temporary data in on-chip memory
    $\Rightarrow$ memory reuse techniques
  - sensitive to external data layout



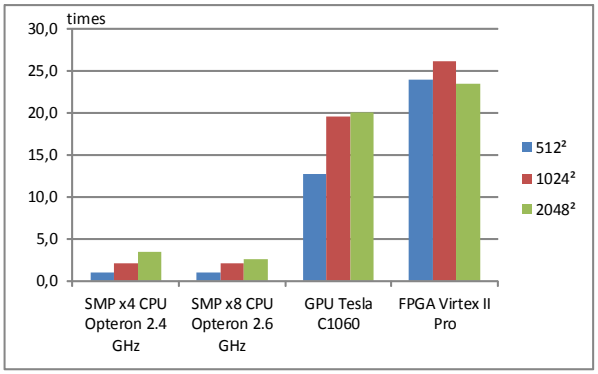
FPGA and GPU combined profile

Summary

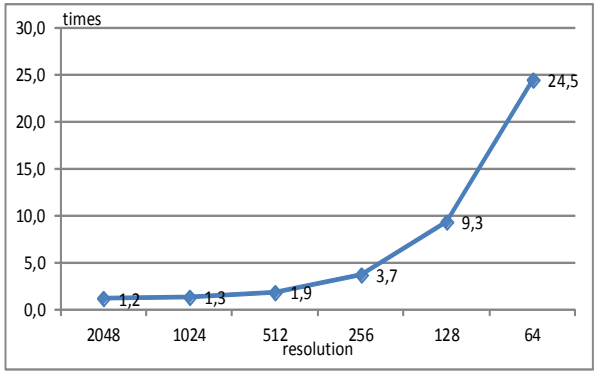
Performance



Elapse Time in ms

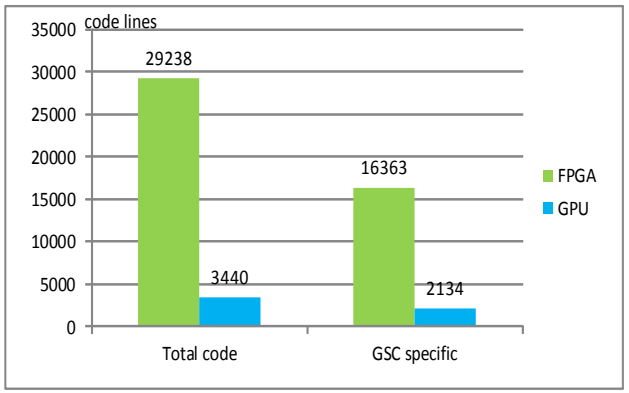


Performance Acceleration Ratios

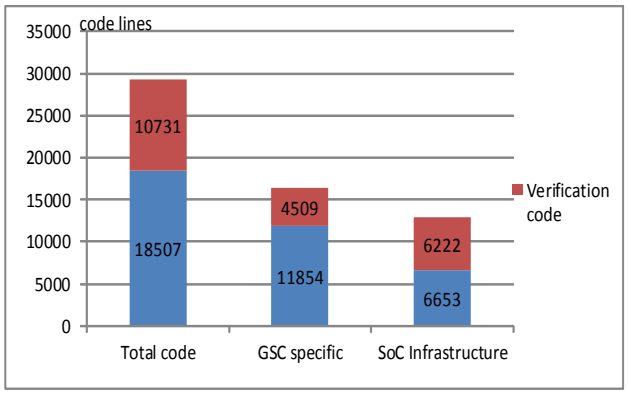


GPU to FPGA processing time ratios

Efforts



Efforts for FPGA and GPU solutions



FPGA code structure