A Combined GPGPU-FPGA High-Performance Desktop

Bruno Da Silva, An Braeken, Jan Cornelis, Erik H. D'Hollander, Jan Lemeire, Abdellah Touhafi, Valentin Enescu

Erasmushogeschool Brussel (IWT Department)

Vrije Universiteit Brussel (ETRO Department), UGent (ELIS Department)

Abstract

- Computation of intensive interactive software applications on R&D desktops require a versatile hardware and software high-performance environment.
- Present-day solutions focus on one technology, e.g. GPUs, grids, multi-cores, clusters, …
- To leverage the power of different technologies, a hybrid solution is presented, combining the power of General-Purpose Graphical Processing units (GPGPUs) and Field Programmable Gate Arrays (FPGAs)

Objectives

- Build a super GPU/FPGA desktop
- Develop a combined tool chain
- Explore industrial applications

Hybrid technologies

GPGPUs:

- Massive SIMD parallelism
- Software tool chain
- Fast GPU-CPU communication

FPGAs:

- Massive fine-tuned parallelism and pipelining
- Optimizing C-to-VHDL compilers
- Algorithm in hardware

Research platform:

GPU: Tesla C2050 NVIDIA

FPGA: Pico Computing w/ 1-6 Virtex 6 FPGAs

Communication links:

PCI 2.0 express 16 lanes (GPU and Pico board)

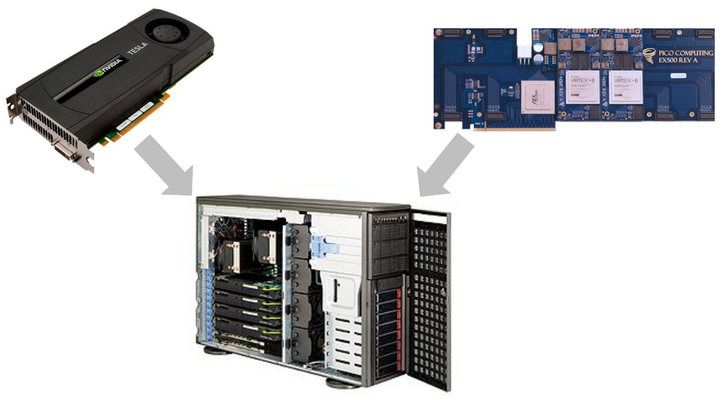


Figure 1. *A PC is extended with GPU and FPGA accelerators to create a high performance computing super desktop platform.*

Combined tool chain

Putting 2 and 2 together:

1. Algorithm: define GPU, CPU and FPGA parts
2. C-program with GPU, CPU and FPGA function calls
3. GPU code → GPU compiler
4. FPGA code → High Level Synthesis (HLS) using a C to VHDL optimizing compiler (ROCCC, AutoESL, ...)
5. FPGA design:
   - bitmap file (FPGA configuration)
   - communication support (FPGA board API)
6. Linker → executable with calls to GPU framework and FPGA co-design
7. Load GPU, CPU code binaries and FPGA configuration binary.
8. Execute program

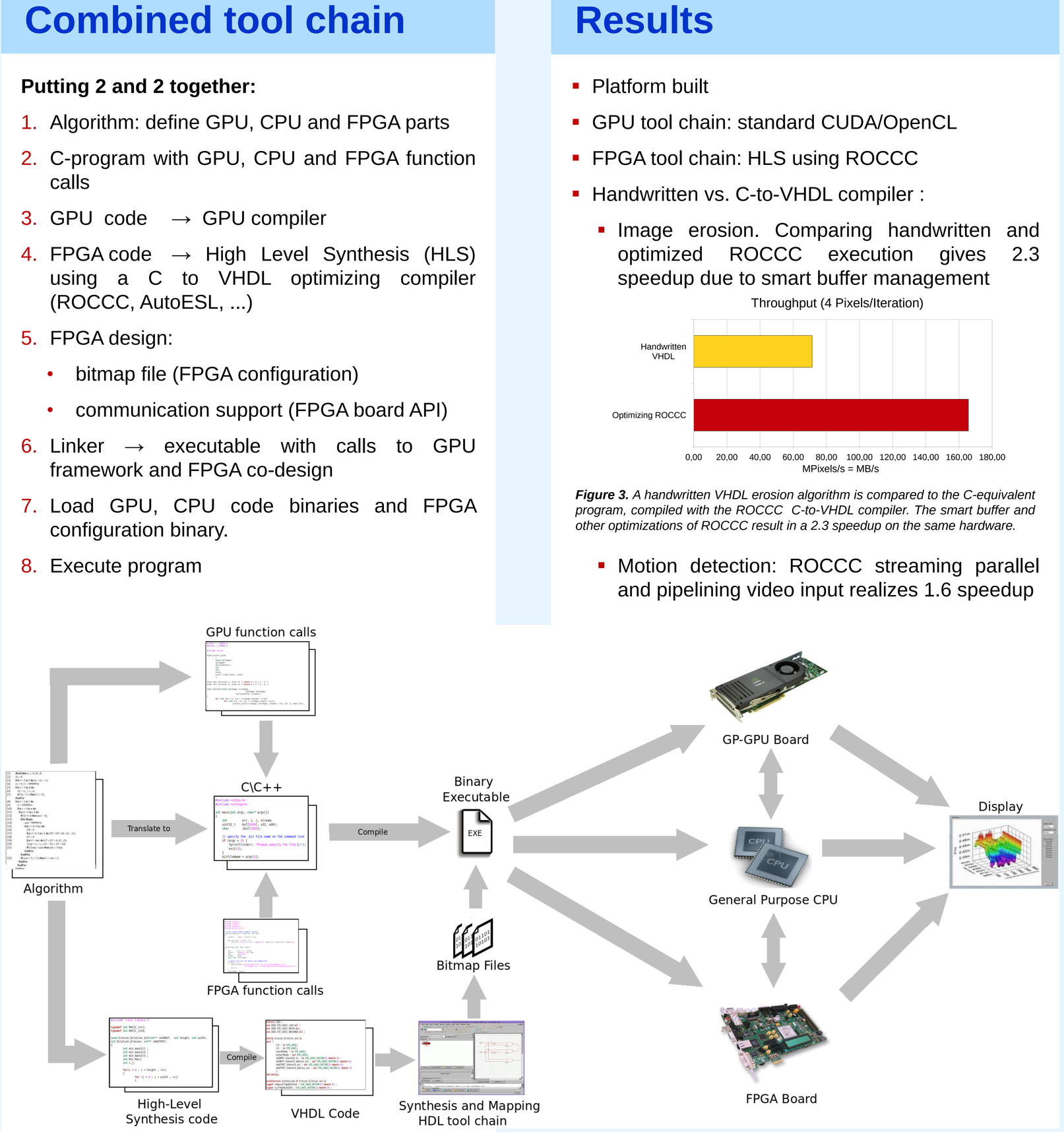


Figure 2. *An algorithm is converted into a C program with mixed code fragments for the three platforms, CPU, GPU and FPGA. The fragments are sent to the cross compilers. The FPGA code is synthesized into a bitmap file and combined with the other binaries. The executable communicates with the GPUs and FPGAs using API libraries.*

Results

- Platform built
- GPU tool chain: standard CUDA/OpenCL
- FPGA tool chain: HLS using ROCCC
- Handwritten vs. C-to-VHDL compiler :
  - Image erosion. Comparing handwritten and optimized ROCCC execution gives 2.3 speedup due to smart buffer management

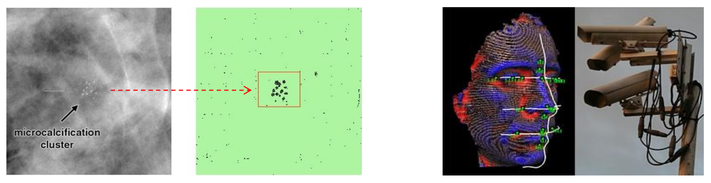
Figure 3. *A handwritten VHDL erosion algorithm is compared to the C-equivalent program, compiled with the ROCCC C-to-VHDL compiler. The smart buffer and other optimizations of ROCCC result in a 2.3 speedup on the same hardware.*

  - Motion detection: ROCCC streaming parallel and pipelining video input realizes 1.6 speedup

Applications

Combining GPU and FPGA strengths:

- Image processing + Bio-informatics
- Face recognition + Security
- Image segmentation + HMMer DB searches
- Traffic analysis + Neural network control



Conclusions

- Combined HPC platform realized
- C-based tool chains available for both platforms, FPGA and GPGPU
- High level synthesis cuts down development time and increases execution speed
- Future work: combine tool chains into efficient multi-platform programming environment

References

1. Cornelis J., Lemeire J. *Benchmarks Based on Anti-Parallel Pattern for the Evaluation of GPUs*, International Conference on Parallel Computing, Ghent, 2011
2. D'Hollander E. H., *High-Performance Computing for Low-Power Systems*, Advanced HPC Systems workshop, Cetraro, 2011

Acknowledgements

This research has been made possible thanks to a Tetra grant of the Flanders agency for Innovation by Science and Technology

Contact details

Bruno Da Silva, brunotiago.da.silva.gomes@ehb.be

Erasmushogeschool Brussel (IWT),

Nijverheidskaai 170, 1070 Brussels, Belgium

Presented at the 7th HiPEAC conference in Paris, January 23-25, 2012