GUDI: A combined GPU/FPGA Desktop System for Accelerating Image Processing Applications

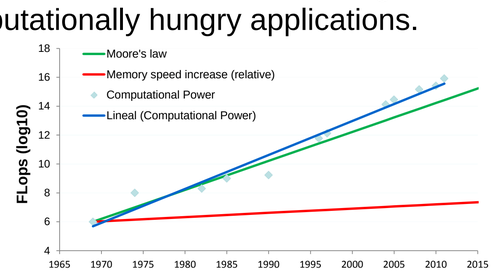
An Braeken, Jan G. Cornelis, Bruno Da Silva, Erik H. D'Hollander, Valentin Enescu, Jan Lemeire, Abdellah Touhafi

Erasmus University College, Brussels

Free University of Brussels, Ghent University

Introduction

- The performance of today's PCs exceeds many times the power of the supercomputers in the 90s, but it is not enough for many computationally hungry applications.



- Present-day solutions focus on one technology, e.g. multi-cores, grids, clusters,…
- To leverage the power of different technologies, a hybrid solution is presented, combining the power of Graphics Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs).

Objectives

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

Hybrid architecture

GPUs:

- Massive SIMD parallelism
- Well-known software tool chain

FPGAs:

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

Research platform:

GPU: Tesla C2050 NVIDIA

FPGA: Pico Computing w/ 2 Virtex-6 FPGA's

Communication link:

PCIe 2.0 x16 lanes (GPU and Pico board)

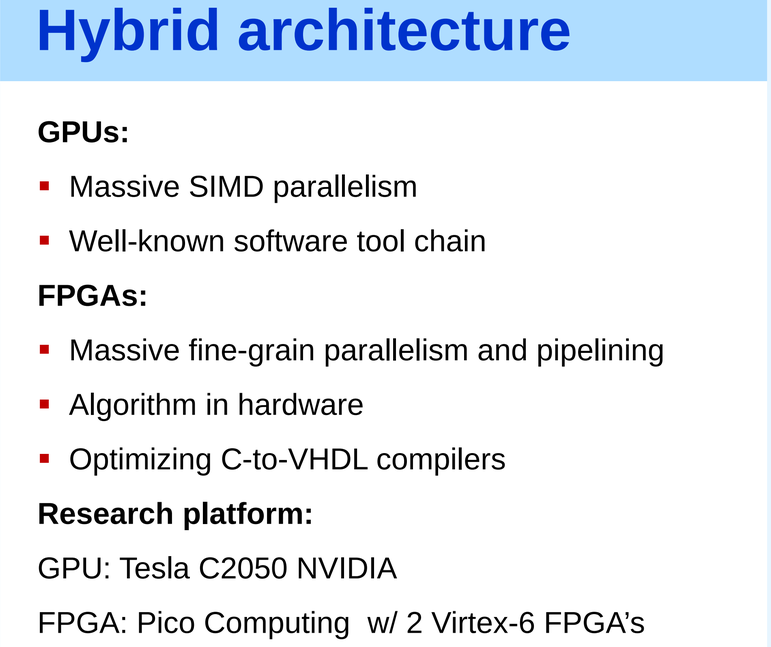
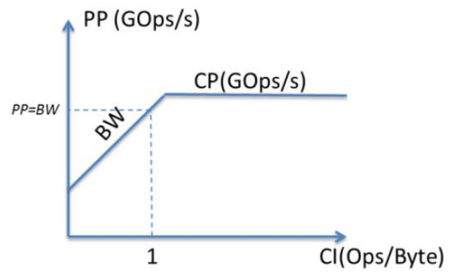


Figure 1. *Detailed architecture combining GPU and FPGA accelerators to create a high performance computing super desktop platform..*

Performance estimation

- The roofline expresses the maximum performance in function of the algorithm's computational intensity (CI), taking into account the peak compute power (PP) and the peak I/O bandwidth of the accelerator (BW).



- Superimposing the rooflines of GPU and FPGA shows the relative performance of both accelerators.

Combined toolchain

1. Identify the parts of the application to be executed on GPU, CPU and FPGA.
2. Create a C++ program with GPU, CPU and FPGA function calls
   - GPU code → GPU compiler
   - FPGA code → High-Level Synthesis (HLS) (ROCCC, VivadoHLS, ...)
3. Compile the programs, synthesize the FPGA design and generate an executable linking the CPU, GPU and FPGA binaries.
4. Load GPU, CPU code binaries and FPGA configuration binary.
5. Execute the program

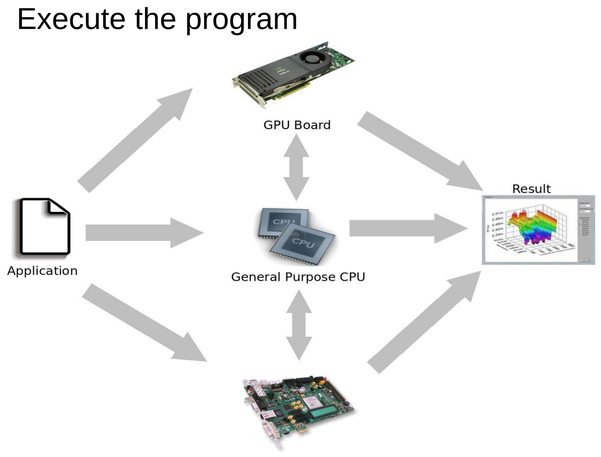
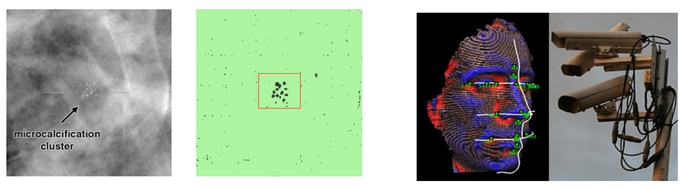


Figure 2. *An algorithm is converted into a C ++ program with mixed code fragments for the three platforms, CPU, GPU and FPGA. The executable communicates with the GPUs and FPGAs using API libraries.*

Applications

Combining GPU and FPGA strengths:

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



Results

- **GPU tool chain: standard OpenCL**
- **FPGA tool chain: Using HLS tools as ROCCC and AutoESL/Vivado HLS**
- **Handwritten vs C-to-VHDL compiler**

  The C-to-VHDL compilers have outperformed handwritten code for algorithms as erosion and is highly productive, but also uses more resources.

- **Comparison of GPUs and FPGAs for image erosion.**

  The measurements show that both the GPU and FPGA excel for image processing algorithms, but both devices suffer from the limited I/O bandwidth to the host.

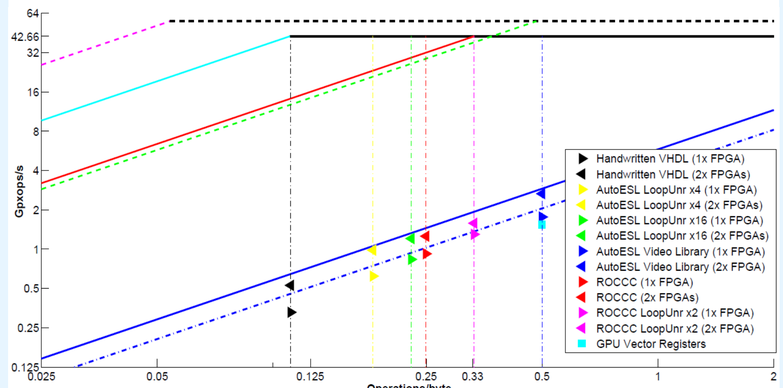


Figure 3*. Roofline models of GPU (dashed lines) and FPGA (continuous lines) with the measured performance values of the algorithm erosion3x3. The PCIe bandwidth (x16 continuous lines and x8 dashed lines) limits the performance for both technologies.*

- **Object Recognition Application (fastHOG): A real collaboration of GPU/FPGA**

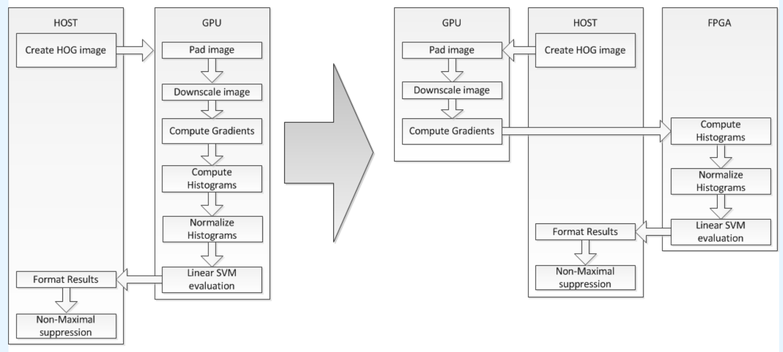


Figure 4*. The object recognition application called fastHOG and designed for GPUs is adapted to be partially executed on the FPGA. The Histogram computation and the normalization are ideal candidates for FPGAs.*

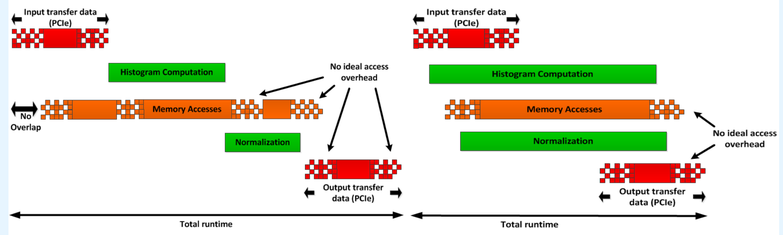


Figure 5*. Execution of HOG on GPU (left) and on FPGA (right). Thanks to operating in streaming and in pipelining mode, the HOG execution can be accelerated on the FPGA.*

Conclusions

- Combined HPC platform
- C++ based tool chain available for both platforms; FPGAs and GPUs
- High-level synthesis cuts down development time and has the potential to increase execution speed for several applications.

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.Erik H. D'Hollander, 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 100132 "A combined GP-GPU/FPGA desktop system for accelerating image processing applications (GUDI)" of the Flanders agency for Innovation by Science and Technology.

Contact details

Jan G. Cornelis, jgcornel@etro.vub.ac.be

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

Presented at the PHPC Symposium in Brussels, December 13, 2012