Performance and Programming Environment of a Combined GPU/FPGA Desktop

Presented at the 21st ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, February 13, 2013

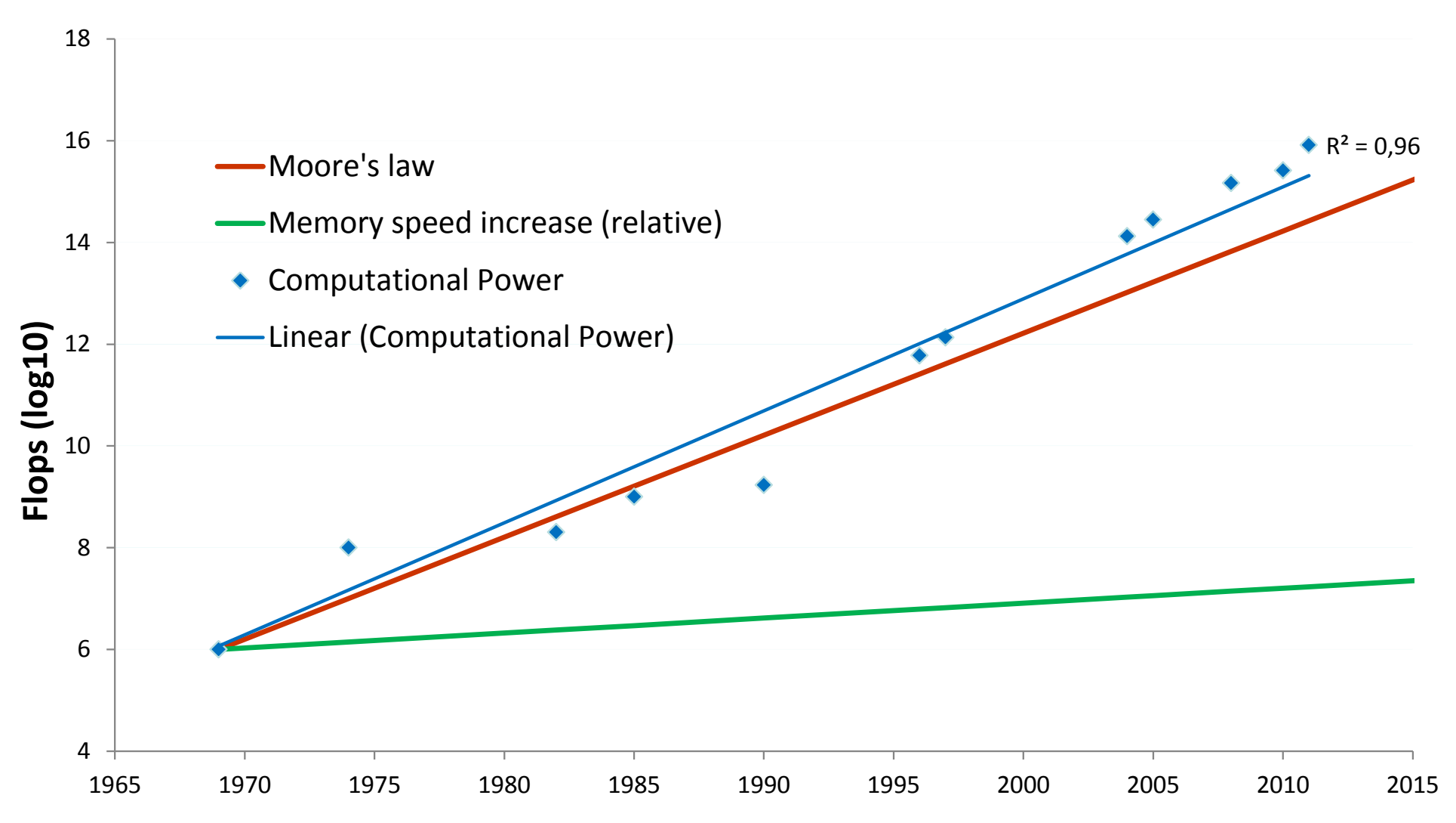
Contact Details: Bruno Da Silva, brunotiago.da.silva.gomes@ehb.be

Bruno Da Silva[1], An Braeken[1], Erik H. D'Hollander[2], Abdellah Touhafi[1], Jan G. Cornelis[3] and Jan Lemeire[3]

[1]Erasmus University College, Department IWT, Brussels,
[2]Ghent University, Department ELIS, Ghent,
[3]Vrije Universiteit Brussel, Department ETRO, Brussels

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.



- To leverage the power of different technologies, a hybrid solution is presented, combining the power of:
  - **Graphics Processing Units (GPUs):**
    - Massive SIMD parallelism
    - Well-known software tool chain
  - **Programmable Gate Arrays (FPGAs)**.
    - Massive fine-grain parallelism and pipelining
    - Algorithm in hardware
    - Optimizing C-to-VHDL compilers

Objectives

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

Hybrid architecture

Research platform:

CPU: Quad-core Intel Xeon E5506

GPU: NVIDIA Tesla C2050

FPGA: Pico Computing w/ 2 Virtex-6 LX240

Communication link:

PCIe 2.0 x16 lanes (GPU and Pico board)

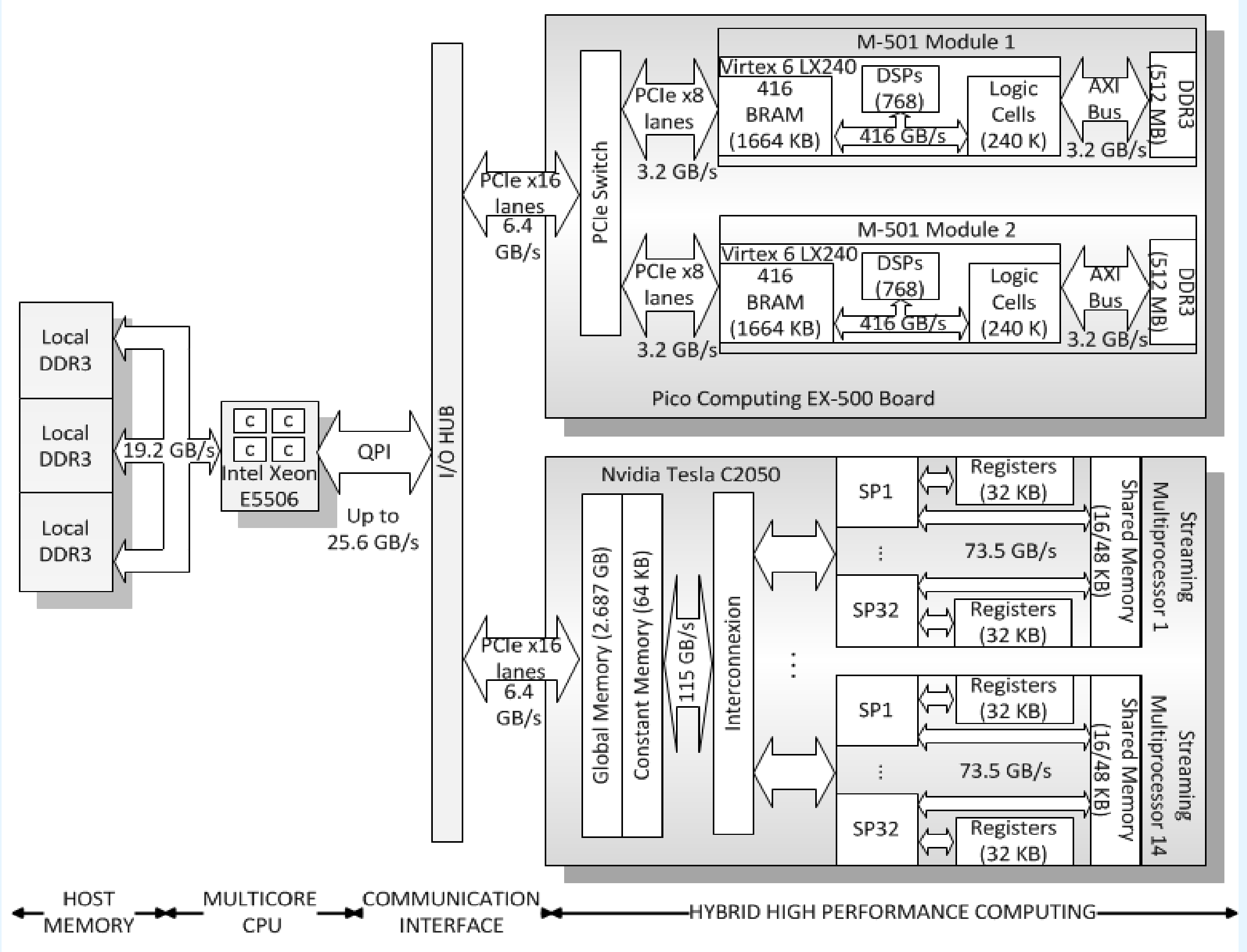


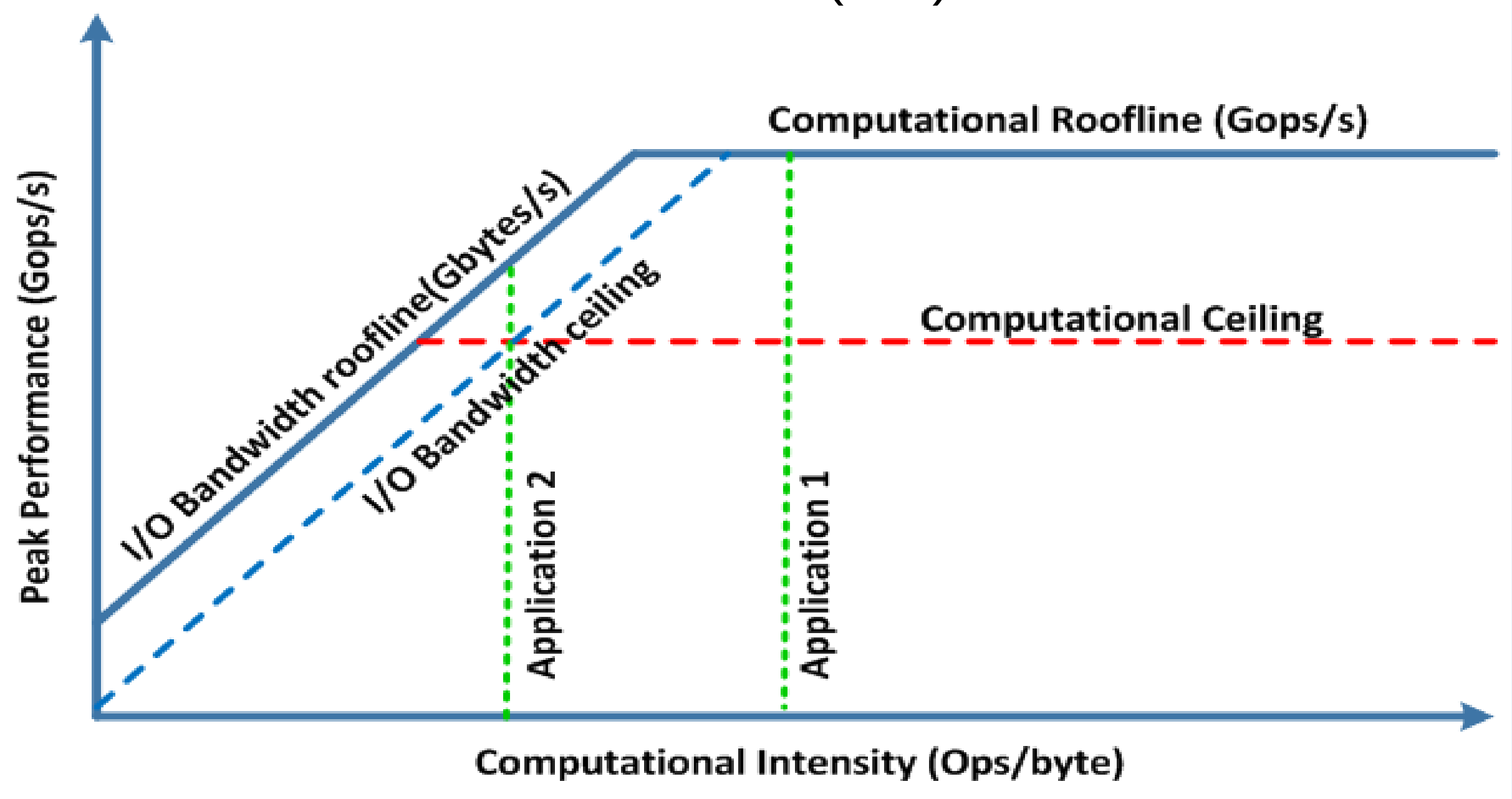
Figure 1. Detailed architecture combining GPU and FPGA accelerators to create a high performance computing super desktop platform..

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.

Performance estimation

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


Computational Roofline (Gops/s)
Computational Ceiling
I/O Bandwidth roofline(Gbytes/s)
I/O Bandwidth ceiling
Application 2
Application 1
Peak Performance (Gops/s)
Computational Intensity (Ops/byte)

$$Peak\ Performance = Min\,(\,CI\ \times\ BW, CP\,)$$

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

Tool chain

Programming steps:

1. Identify the parts of the application to be accelerated by the GPU and/or the FPGA.
2. Create a C/C++ program to be executed by the CPU with GPU and FPGA function calls.
   - GPU code → GPU compiler
   - FPGA code → High-Level Synthesis (HLS) (ROCCC, Vivado HLS, ...)
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

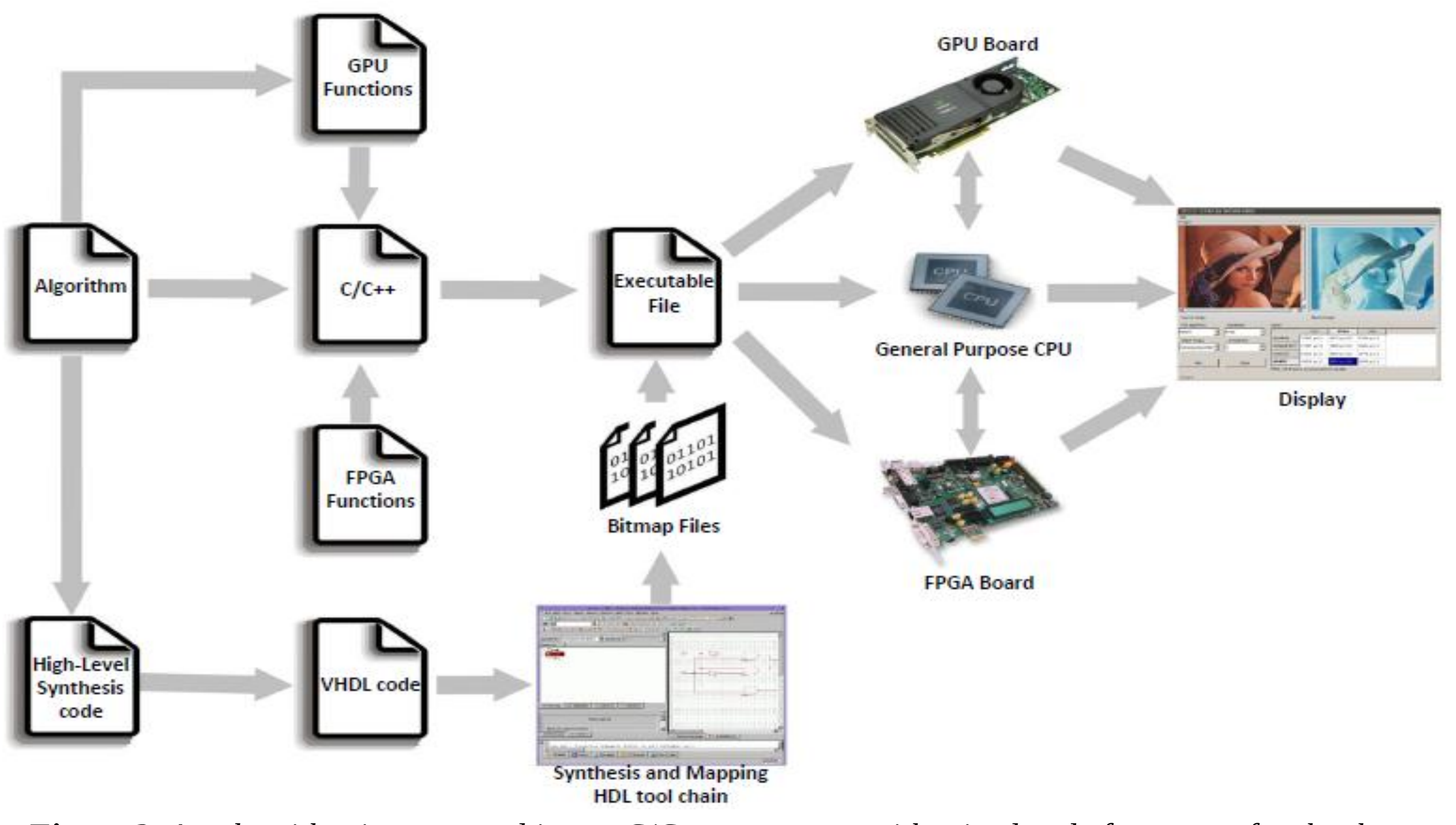


Figure 2. An algorithm is converted into a C/C ++ program with mixed code fragments for the three platforms, CPU, GPU and FPGA. The executable communicates with the GPUs and FPGA using API libraries.

Conclusions

- Combined High-Performance Computing platform
- C/C++ based tool chain available for both platforms; FPGAs and GPUs
- The I/O bandwidth has a significantly impact over the final performance.
- High-level synthesis cuts down development time, making FPGAs an alternative for market solutions.

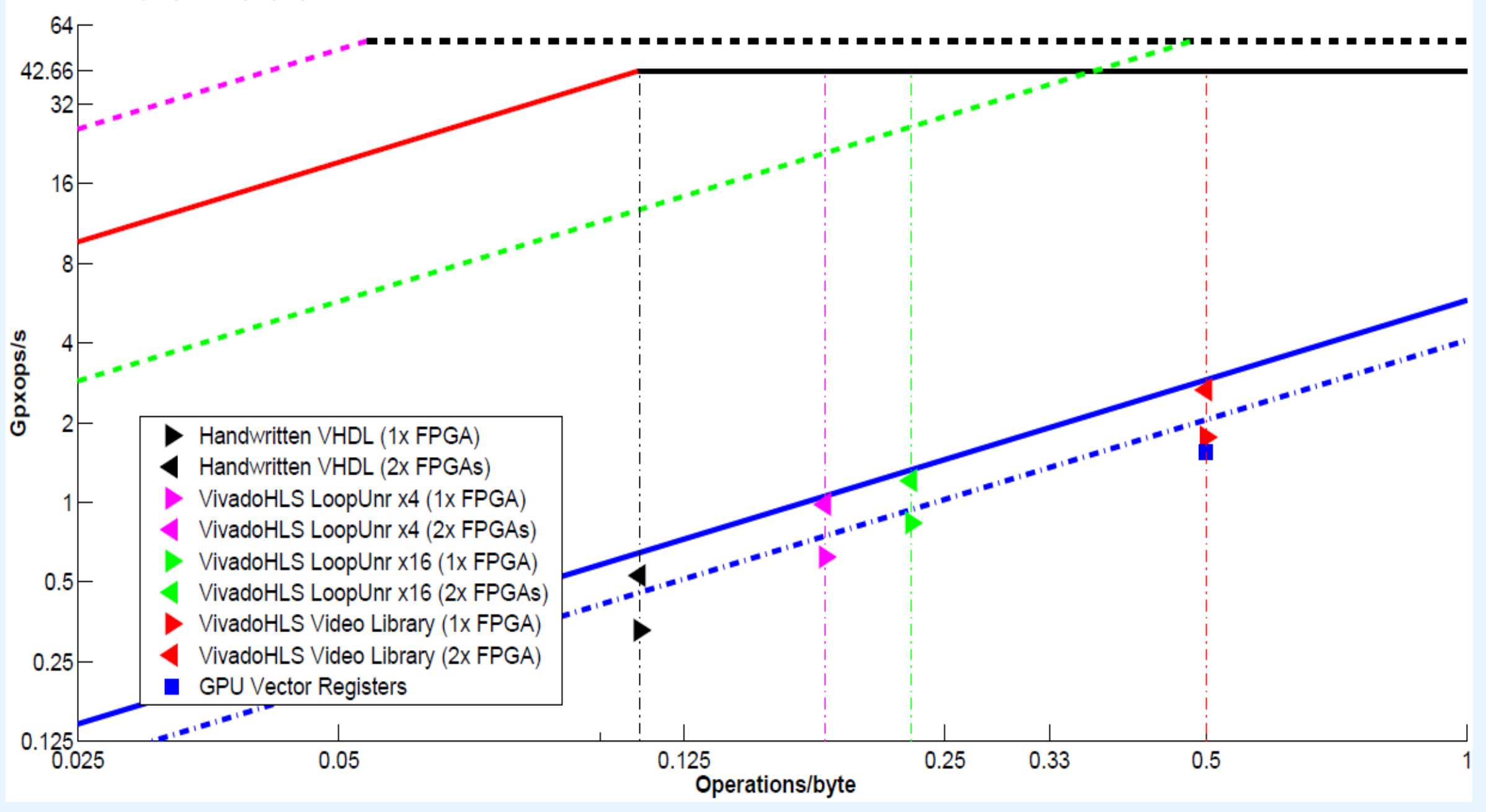
Comparison of GPU/FPGA

- **Handwritten vs C-to-VHDL compiler**

  The C-to-VHDL compilers are highly productive and outperform handwritten code for algorithms such as erosion, but commonly use more resources.

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

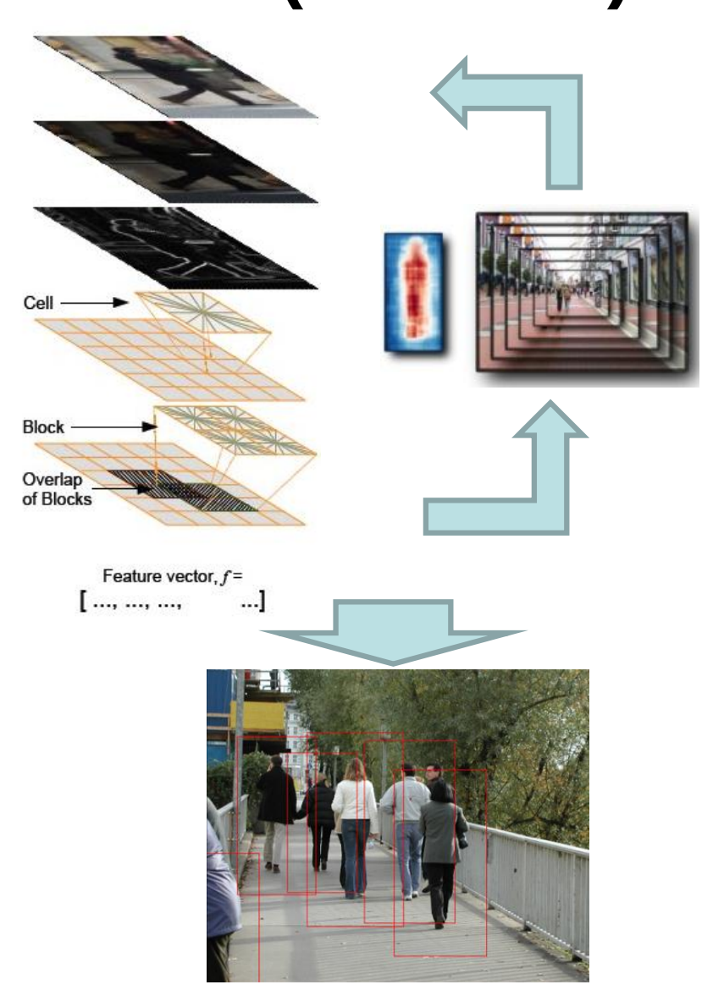
  The measurements depicted on the superimposed roofline models of GPU (dashed lines) and FPGA (continuous lines) show that both GPU and FPGA excel for image processing algorithms. However, the PCIe bandwidth (x16 continuous lines and x8 dashed lines) limits the overall performance of both devices.



Combination of GPU/FPGA

- **Pedestrian Recognition Application (fastHOG)**

The pedestrian recognition application fastHOG, originally designed for GPUs, is composed of the Histogram Oriented Gradients (HOG) and Support Vector Machines (SVM) components which are executed several times on the downscaled images.



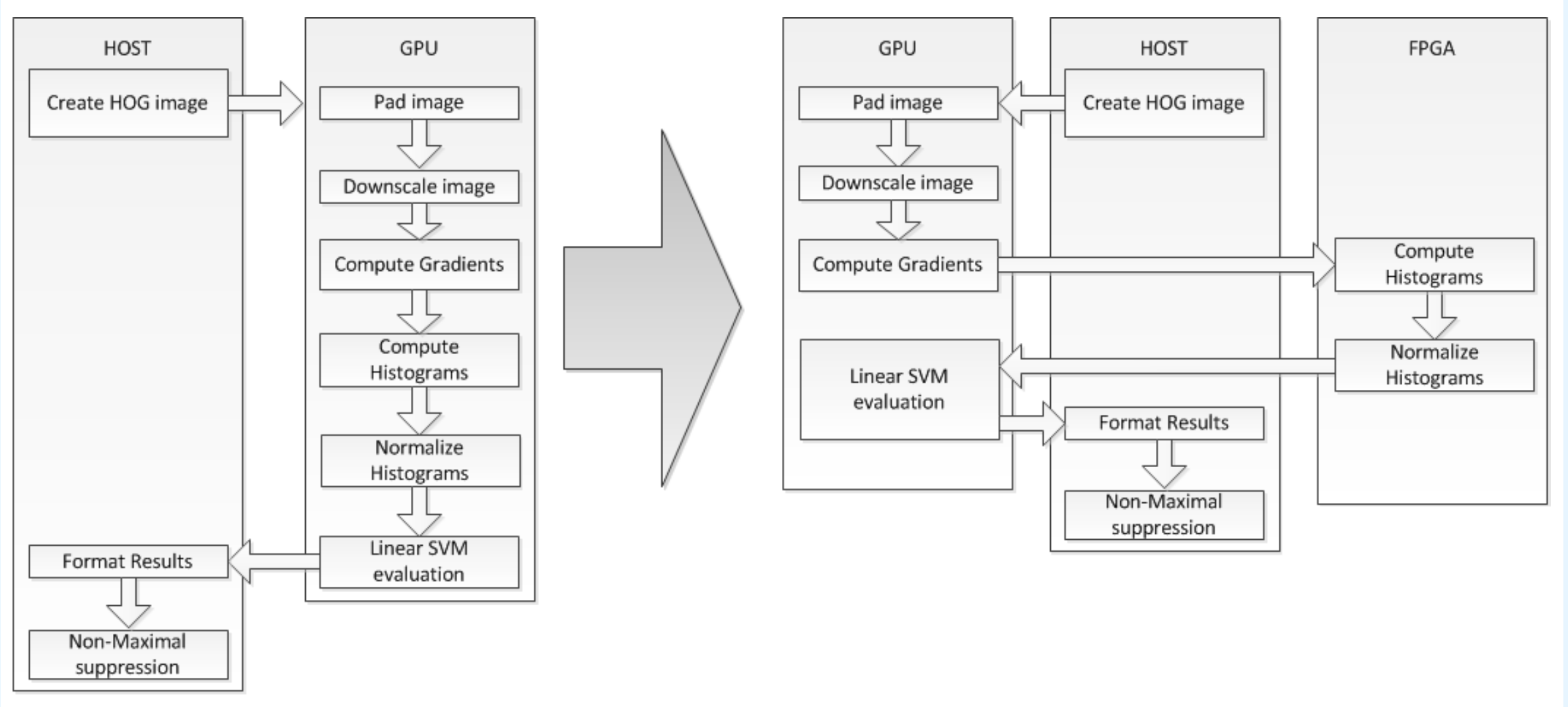
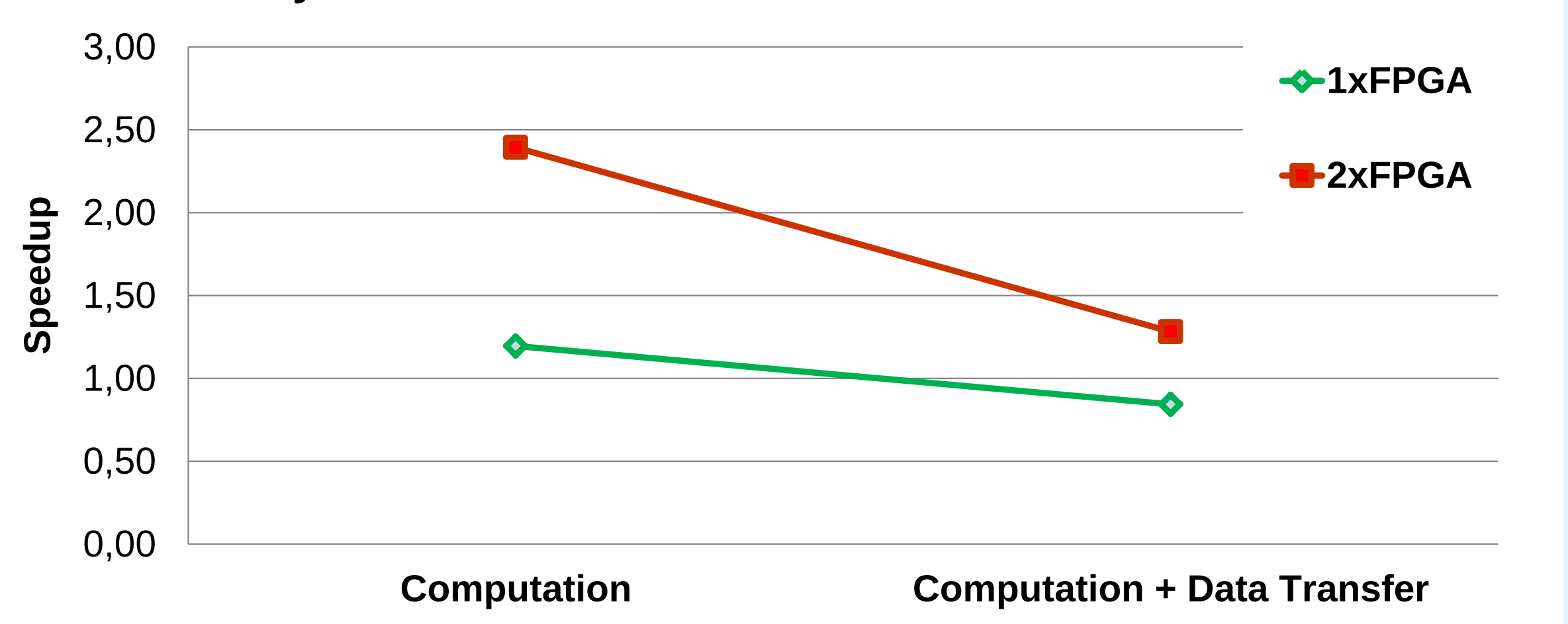


Figure 3. The application is adapted to be partially executed on the FPGA. The Histogram computation and the normalization are ideal candidates for FPGAs.

The HOG computation on the FPGA is faster than on the GPU. However, the speedup combining GPU/FPGA is bounded by the PCIe bandwidth due to the data transfer.



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