Harnessing FPGAs potential with OpenCL

Maxime MARTELLI[1,2,3], Nicolas GAC[1], Alain MÉRIGOT[2], and Cyrille ENDERLI[3]

[1]Laboratoire des Signaux et Systèmes, CentraleSupélec, CNRS, Université Paris Sud,Université Paris-Saclay, FRANCE
[2]Laboratoire des Systèmes et Applications des Technologies de l'Information et de l'Énergie,ENS Cachan, CNRS, Université Paris Sud, Université Paris-Saclay, FRANCE
[3]Thales Systèmes Aéroportés S.A., Elancourt, FRANCE

Abstract

The work presented deals with the evaluation of FPGAs resurgence for hardware and software acceleration. We focus our attention on the tools developed by FPGAs manufacturers, in particular the Intel FPGA SDK for OpenCL, that promises a new level of hardware abstraction from the developer's perspective, allowing a software-like programming of FPGAs. Our first contribution is to propose an accurate memory benchmark, and we follow with an evaluation of different custom OpenCL implementations of one use case : the computed tomography. With some clues on memory fetching and coalescing, we then further tune designs to improve performance. Finally, a comparison is made with GPU implementations, and a preliminary conclusion is drawn on FPGAs future in the semi-conductor realm. This presentation include but is not limited to results presented in [3].

1 Introduction

With the end of Moore's Law, the semi-conductor industry seeks a reliable way to pursue the performance improvements of the last decades, and architecture-algorithm adequation is a solution for this new landscape.

With this in mind, FPGA Manufacturers like Xilinx and Intel are pushing for an FPGA resurgence, offering software suites (called SDAccel and Intel FPGA SDK for OpenCL, respectively) and FPGAs card focused on a software-like FPGA programming model. The considered use case is the back-projection algorithm used in iterative reconstruction [1]. After the raise of GPGPU, architectures designed on FPGA were put aside by the tomography reconstruction community but nowadays, new architectures and improved HLS tools may very well make FPGA competitive again as a forgotten accelerator for 3D tomography reconstruction.

2 3D Tomography algorithm

Computed Tomography relies on the analysis of a known radiation stream through the considered object to recover said physical characteristic by reversing the matter transport equation [4]. An X-ray source (Fig.1) revolves around the φ axis at $z = constant$. Radiation emitted from it is attenuated depending on the object local density, and a two-dimensional sensor array records intensity values, for each elementary φ angle. Those values are stored in a 3D matrix along (u,v,φ) in what is called a sinogram. From this sinogram, the 3D volume is reconstructed using among other processing the back-projection operator.

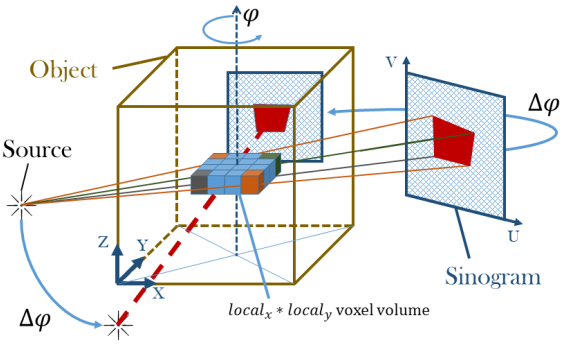

Figure 1. 3D Computed Tomography Projection.

3 Programming FPGAs with OpenCL

OpenCL is an abstract programming model. The host program is written in standard C, and handle communication between the host processor and devices function, also

called kernels. Each kernel is transformed by the Altera Offline Compiler in a sequence of logic blocks, creating elementary pipelines that are then aggregated to form a kernel pipeline.

The OpenCL model has four memory categories : global, constant, local, and private. Manual memory handling is essential for effectively improving software implementations efficiency. In order to characterize FPGA memory structure in OpenCL, we propose a custom benchmark to calculate memory latency for the four memory types which results are shown in Table 1.

Table 1. Measured memory latency on an Altera Cyclone V.

Memory type	Kernel Frequency (MHz)	Latency (cycles)
Global	137.36	164
Constant	150.4	45
Local	137.11	12
Private	161.31	1

4 OpenCL 3D Back-projection implementations

There are two OpenCL kernel categories : data parallelism (NDRange) and task parallelism(single work-item - SWI) . Data parallelism is the simultaneous execution on multiple cores of the same function across the elements of a dataset whereas task parallelism is the simultaneous execution on multiple cores of many different functions across the same or different datasets.

A first optimization is to improve streaming throughput for SWI kernels. By default, when a kernel needs to access an array, it allocates memory resources for efficient reads and writes. When an array access pattern matches with a streaming pattern, implementation can be modified to integrate a Shift-Register Pattern and compilers will implement it as a cascade of flip flops, sharing the same clock. Because the main challenge of the back-projection algorithm was to efficiently access the sinogram array, a second optimization consist in designing a custom prefetching algorithm suited for data parallelism, with local memory sharing.

5 Results and discussion

5.1 OpenCL optimization for FPGA

In Table 2, the CPU-like version is the sequential algorithm with no optimization, and the GPU-like version is the SIMD algorithm. SRP(SWI) implements a shift-register pattern, and MF(NDRange) is the memory fetching kernel.

We achieved to port a CPU-like code on a FPGA, with an overall speedup of 8.74 between the naive and the best optimized kernel on a Cyclone V chip.

Table 2. Normalized Execution Time [NET] (to the logic utilization) of various kernel optimizations on the Cyclone V SoC.

Kernel version	NET(s)
CPU-like (SWI)	109.2
SRP(SWI)	24.3
GPU-like(NDRange)	17.7
MF(NDRange)	12.5

5.2 GPU versus FPGA, consumption and performance

For an adequate comparison to high-end GPUs, we extrapolated the results obtained from the DE1-SoC to an SX660 Arria 10 FPGA. As shown in Table 3, in terms of raw performance, FPGAs are merely comparable to GPUs. FPGA with VHDL is known for its efficiency and low energy consumption and here, this characteristic is preserved even with OpenCL. Indeed, its characteristic in VHDL is to be able to provide a fine-grained architecture fitted to the chosen algorithm, and to obtain a highly efficient design like in [2].

Table 3. Efficiency (Cycles needed for one voxel update per core) between GPUs and FPGAs. (256^4 voxel computation)

Device	NET (ms)	Energy (mWh)	Efficiency (cycles)
Titan X Pascal	12	0.83	12.16
Jetson TX2	253	1.054	19.6
Intel Arria 10	991	0.63	1.02

6 Current and future work

Our first lead is to characterize general algorithms to choose which suits best the FPGAs, and from there, to build a comprehensive roadmap on how and what to accelerate on FPGAs with HLS tools.

7 Conclusions and perspectives

Some hardware knowledge is required to fully harness the power of OpenCL on FPGAs, but our optimized FPGA HLS design to compete with hand-coded VHDL implementation in much lesser coding time (2 months in OpenCL vs more than a year for a 3D-cache memory fetching algorithm with VHDL [2]). Overall, OpenCL tools do a fine job constructing an adequate architecture for the algorithm, but eventually the lack of raw power from FPGAs compared to GPUs can be a strong liability for massively parallels algorithms.

References

[1] P. E.Kinahan et al. Emission tomography : the fundamentals of PET and SPECT, chapter Analytic image reconstruction methods. Elsevier Academic Press. 2004.

[2] N. Gac, S. Mancini, M. Desvignes, and D. Houzet. High Speed 3D Tomography on CPU, GPU, and FPGA. *EURASIP journal on Embedded Systems*, 2008.

[3] M. Martelli, N. Gac, A. Mérigot, and C. Enderli. 3D Tomography parallelization on FPGAs via HLS tools. In *DASIP*, Dresden, Germany, Sept. 2017.

[4] M. Thurston, M. M. Nadrljanski, et al. Computed tomography - radiology reference article. *Radiopedia*.