Harnessing FPGAs potential with OpenCL
PhD Student: Maxime MARTELLI
Thesis: Hardware architecture for very fast sampling generation. Application to radar and electromagnetic listening systems certification
Supervisor: Alain MERIGOT. Co-supervisors: Nicolas GAC, Cyrille ENDERLI

Industrial Context
1. 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 adequacy is a solution for this new landscape.
2. Manufacturers like Intel and Xilinx are pushing for an FPGA resurgence, offering software suites and FPGAs card focused on a software-like FPGA programming model[1].

General Thesis Objectives
1. Evaluate and use heterogeneous architectures to accelerate algorithms, specifically RADAR processing.
2. Enabling software-like programming for FPGA implementations with architecture and algorithm adequacy in mind.
3. Definition of a methodology for implementing acceleration for general algorithms depending on their inherent specificity.

First use case: 3D X-ray Computed Tomography Reconstruction

Using OpenCL on FPGAs
There are two OpenCL kernel programming models: NDRange (NDR) that is Data Parallelism and Single Work Item (SWI), that enables task parallelism, OpenCL allows FPGAs programming, and one contribution of this work is to evaluate its effectiveness compared to VHDL and GPU implementations [2].

Memory Benchmark
For benchmarking purposes, we implemented a custom routine program to measure the mean latency of each memory type on an Altera Cyclone V FPGA.

<table>
<thead>
<tr>
<th>Memory structure</th>
<th>Kernel Frequency (MHz)</th>
<th>Mean latency (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>137.4</td>
<td>164</td>
</tr>
<tr>
<td>Constant</td>
<td>150.4</td>
<td>45</td>
</tr>
<tr>
<td>Local</td>
<td>137.1</td>
<td>12</td>
</tr>
<tr>
<td>Private</td>
<td>161.3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: Memory structure latency on an Altera Cyclone V.

Implemented OpenCL optimizations
- SWI Naive: CPU like naive version.
- SWI+SRP: Shift Register Pattern (FIFO queue) to reduce logic utilization.
- NDR+Naive: CPU like naive version (shared local memory).
- NDR+2CU: Kernel replication of the NDR+Naive version.
- NDR+MF: Prefetching mechanism bound to the algorithm specificity. The goal is to access the pattern needed by a group of voxels in one call.

FPGA obtained speedup and GPU comparison

<table>
<thead>
<tr>
<th>Device</th>
<th>NET (ms)</th>
<th>Energy (mWh)</th>
<th>Efficiency (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titan X Pascal</td>
<td>12</td>
<td>0.83</td>
<td>12.16</td>
</tr>
<tr>
<td>Jetson TK2</td>
<td>253</td>
<td>1.054</td>
<td>19.6</td>
</tr>
<tr>
<td>Intel Arria 10</td>
<td>991</td>
<td>0.63</td>
<td>1.02</td>
</tr>
</tbody>
</table>

Even with the extrapolated results on the improvements obtained on an Arria 10, the FPGA has a much longer execution time than the GPUs, mostly due to the back-projection algorithm being adapted to SIMD architectures.

Conclusion
1. With an overall 8.74 speedup [3] between naive and optimized kernels, there is room for more improvements closely related to memory handling.
2. With OpenCL, FPGAs have a good pipeline efficiency. But, the algorithm being well suited for SIMD execution, FPGAs frequency and limited pipeline replication explains their limited performance compared to GPUs.

Current works and perspectives
1. Accelerating radar processing algorithms on FPGAs.
2. Benchmarking FPGAs using OpenCL with 1D/2D filtering algorithms.
3. Comparing various implementations on heterogeneous architectures (CUDA, OpenACC, OpenMP, OpenCL).

References