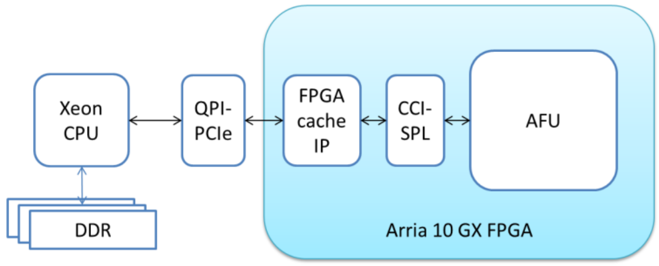
EXPLORING OPENCL ON A CPU-FPGA HETEROGENEOUS ARCHITECTURE RESEARCH PLATFORM (HARP)

Thomas Faict, Erik H. D'Hollander, Dirk Stroobandt, Bart Goossens – Ghent University

FPGA and OpenCL: do they deliver a HPC hardware accelerator?

Intel introduced the Heterogeneous Architecture Research Platform (HARPv2)[1]. The platform interconnects a CPU (Broadwell Xeon) and a Field-Programmable Gate Array (Intel Arria 10 GX) through a high speed, low latency QPI/PCI data path providing cache coherency and shared memory. To program the FPGA, a High Level Synthesis (HLS) language, OpenCL, is made available [2]. By using HLS, a faster design cycle can be achieved compared to programming in a traditional Hardware Description Language (HDL). This, however, comes at the cost of having less control over the hardware implementation. We will investigate how efficiently OpenCL can be applied on the HARP platform.

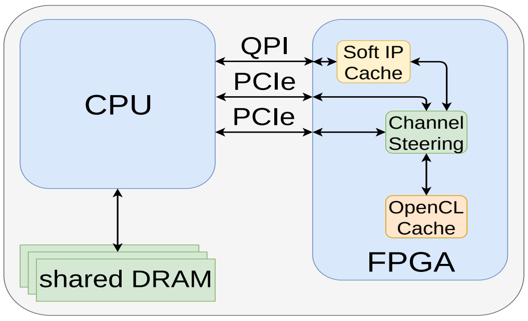
I. HARP performance enhancing features



The integrated package of a Xeon CPU and an Arria 10 GX FPGA features a Cache Interface and System Protocol layer (CCI-SPL), supporting virtual memory address translation and cache coherency. The user generated Accelerator Functional Unit has access to DDR shared memory. The integrated package provides several performance enhancements.

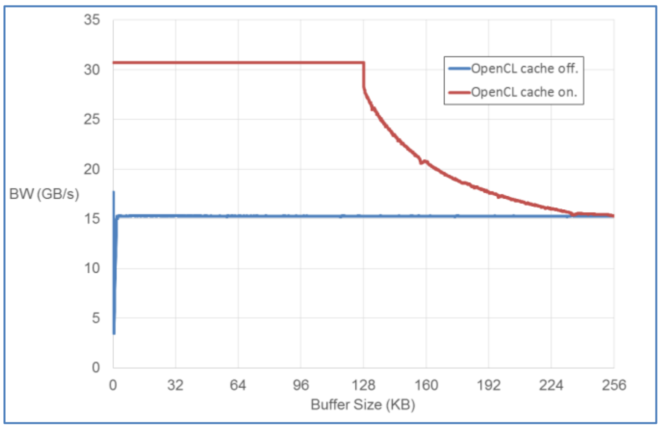
1. FPGA coherent caches

The FPGA contains two different caches



- A **Soft IP cache** of 64 KB is part of the in the standard IP core.
- An **OpenCL cache** is generated for **streaming Load-Store Units** having a data-dependent or repetitive memory access pattern.
- The OpenCL cache to memory path (QPI or PCIe) is selected by the **Channel Steering** core.
- The **caches are coherent** with the CPU Last Level Cache.

Cache efficiency of the OpenCL cache



The bandwidth **without the OpenCL cache** is 16 GB/s. The bandwidth **with the OpenCL cache** exceeds 30 GB/s. The OpenCL generated cache therefore doubles the bandwidth for sizes up to 128 KB (largest OpenCL cache).

2. FPGA - DDR memory bandwidth

Raw bandwidth

- QPI (Quick Path Interconnect): 12.80 GB/s, PCIe : 7.88 GB/s
- Combined: 1 QPI + 2 PCIe : 28.56 GB/s

Measured bandwidth

- **Bandwidth vs. Word size**
  Reading 4, 64 and 128 byte words from DRAM yields measured bandwidth between 1 and 16 GB/s. The buffer size varies between 2 and 4 MB .
- **Bandwidth vs. Unroll factor**
  The bandwidth for small word sizes can be increased by unrolling loops to coalesce data access. This increases the bandwidth up to 14 GB/s for a unroll factor > 32, accessing 128 bytes or 32 floats in parallel.

3. Shared Virtual Memory

- HARP shares virtual memory between CPU and FPGA using the SPL (System Protocol Layer) for address translation.
- OpenCL kernel shares OpenCL host memory using virtual address mapping (clSVM), avoiding the traditional data transfer to private memory (clEnqueueWriteBuffer, clEnqueueReadBuffer).

- SVM saves substantial memory- and communication-overhead.

II. Can OpenCL exploit HARP's performance enhancing features?

Case study: Guided Image Filter

Guided image filtering [3] is an image smoothing technique in which the properties of two different images are combined. The guided image filter takes two images, a filtering image I and a guidance image G, as input and smooths the input image by incorporating the guidance image. The filter operates on a window of rxr neighboring pixels, where r is called the radius parameter.

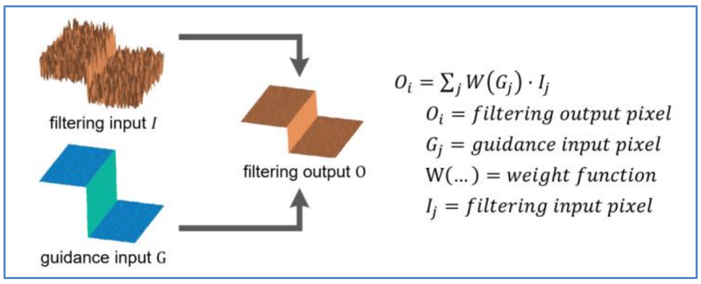
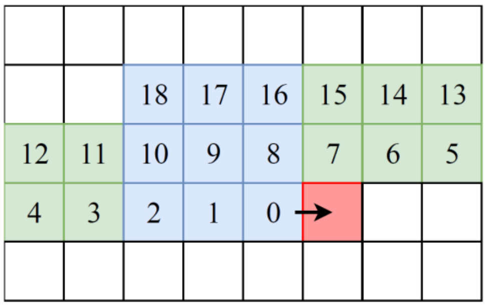

filtering input I
guidance input G
filtering output O
$O_i = \sum_j W(G_j) \cdot I_j$
O_i = filtering output pixel
G_j = guidance input pixel
$W(\ldots)$ = weight function
I_j = filtering input pixel

Figure reproduced from Kaiming He et al.[3].

- I, G, O = input, guiding and output images (1080x1920 RGB pixels)
- $\sum_j$ = window of (2r+1)(2r+1) pixels with r = Radius of filter

Sliding window operation



The figure shows a line buffer for sliding window operations on an image of size MxN with M=3, N=8 and radius r=1. The line buffer size = (N+1)*2*r+1. At each clock cycle, a new pixel is loaded in the line buffer and all values are shifted one place. The sliding window of size (2*r+1)(2*r+1) is used to calculate output pixel $O_i = \sum_j W(G_j) I_j$

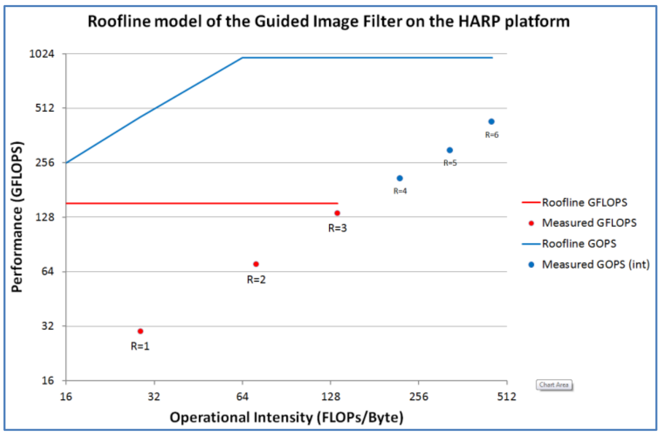
Optimizations

The code has been ported to OpenCL with the following characteristics:

1. Streaming data input and output, using line buffers for I, G and O.
2. Unrolled sliding window loops to pipeline (2r+1)(2r+1) iterations.
3. Vectorization for SIMD operations.
4. Resource saving fixed point operation for radii r > 3.
5. Shared virtual memory to avoid explicit data buffering.

Roofline model

The roofline model [4] shows the maximum performance subject to hardware and bandwidth constraints. The maximum performance is determined by the resource constrained problem size. The maximum problem sizes for floating point and fixed point operations are for radius r=3 and r=6 respectively.



Results

The maximum GFLOPS roof is approached for radius r=3.

- Larger problems use fixed point ops (r=6).
- Measured performance is 135 GFLOPS (floating point), 430 GOPS (fixed point), 70 fps (frames per second).
- OpenCL cache is very effective.
- Guided Image Filter algorithm remains I/O bound.

Profile of the performance features

- OpenCL coherent cache impact: 63-90% hit rate.
- Bandwidth channels: Read 3..3 GB/S, Write: 2.7 GB/s.
- Shared vs Private memory: 74 frames/s vs 41 frames/s.

III. Conclusion

OpenCL on the HARP is able to capitalize on the coherent cache, shared virtual memory and high bandwidth channels. The guided image filter case study provides throughput of HD (1920x1080) 24-bit color images at 74 fps.

References

[1] Intel, “Hardware Accelerator Research Program,” https://software.intel.com/en-us/hardware-accelerator-research-program [Accessed: 15-Jan-2019].

[2] Intel, “Intel® FPGA SDK for OpenCL™ Standard Edition Programming Guide,” p. 180, Sep. 2018.

[3] K. He, J. Sun, and X. Tang, “Guided Image Filtering,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 6, pp. 1397–1409, Jun. 2013.

[4] Samuel Williams, Andrew Waterman, and David Patterson, “Roofline: an insightful visual performance model for multicore architectures,” Communications of the ACM, vol. 52, no. 4, pp. 65–76, 2009.