Microdisk Cavity FDTD Simulation on FPGA using OpenCL

Tobias Kenter
kenter@uni-paderborn.de

Christian Plessl
christian.plessl@uni-paderborn.de

Paderborn Center for Parallel Computing and Department of Computer Science
Paderborn University, 33098 Paderborn, Germany

ABSTRACT

As computational sciences strive for higher compute performance to simulate new physical phenomena, FPGA-based accelerators are emerging as potential target platform. First solutions on such platforms have been presented, however they are designed with languages and formalisms that are unfamiliar to HPC developers. In this work we investigate one possible solution to close this gap by using the software-centric OpenCL-based SDAccel tool flow. We generate an FDTD solver for computational nanophotonics and present highly competitive performance results using FPGAs.

1. APPLICATION

Our application solves Maxwell's equations to compute the propagation of light in metal with a nanostructure using the finite-difference time domain (FDTD) method introduced by Yee [3]. FDTD is an prototypical stencil application working on a regular grid that uses a weighted sum of the values of the electric and magnetic fields in the direct neighborhood of the grid point to compute the values for the next time step. FDTD is more complex than the widely used Laplace stencil because two different, interleaved vector fields with different local dependencies are required. The nanophotonic device in our simulation is a so-called microdisc cavity, that is a circular hole (vacuum) surrounded by perfect metal. Figure 1 illustrates modes in the microdisc cavity observed after 144.000 simulation steps. This is a well-suited structure for a non-synthetic 2-dimensional benchmark because it is used in actual nanophotonic devices and the solution of the problem can be analytically determined. Also, the use of different materials requires to handle different, material-dependent stencils for geometric domains, which breaks the simple control flow and complicates vectorization.

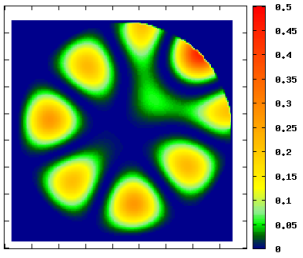

Figure 1: Visualization of simulation result: integrated energy of h-field.

2. IMPLEMENTATION

For our FPGA implementation, we started with a non-parallelized software implementation using single precision floating point and first wrapped the unaltered main loop into an OpenCL kernel. The Xilinx SDAccel tool flow internally first uses high-level synthesis (HLS) to generate a hardware description and then proceeds with the more time-consuming synthesis, place and route steps to implement the design on FPGA. Within a few hours, we had the first design up and running in hardware — however, around three orders of magnitude slower than on CPU.

The optimization of the design was much more challenging. We first aimed for two design aspects, prefetching of all data in bursts to local on-chip buffers (using BRAM resources), and pipelining of the computations within each row of the grid, such that with an initiation interval of two, a new result value can be produced in every second cycle, once the pipeline is filled. We had to explore many different code patterns to achieve these properties, but by relying on HLS reports as main source for optimization, we achieved reasonably fast design iterations. Towards our design goals, we decoupled the burst memory transfers from the computations by wrapping them into individual input and output kernels that are connected to the main compute kernel via pipe constructs. In order to adapt the computations to the data stream arriving for the h-field update, we retimed the update expression to be evaluated one row later than the loop indices would imply. Also, we moved any accesses to local memory out of conditional statements and left only selection operations inside if and else-blocks.

After this approach allowed for the intended pipelining and achieved performance in the same order of magnitude as a single-threaded CPU implementation, the main computation kernel was additionally unrolled in the simulation time domain to compute several simulation time steps in a pipelined way. With this temporal unrolling, the computational intensity of the overall design increases, as data does not leave the FPGA between pipeline stages. Instead, it is passed through local memory. We did not manage to use one big block of local memory with several disjoint partitions and memory ports, as the HLS analysis could not determine that each partition was exclusively written to by one pipeline stage and read from by exactly one other stage and thus created overly cautious loop schedules. As workaround, with the help of preprocessor macros, we instantiated and used individual local memories for each pipeline stage. Figure 2 illustrates the overall OpenCL-based FPGA-design.

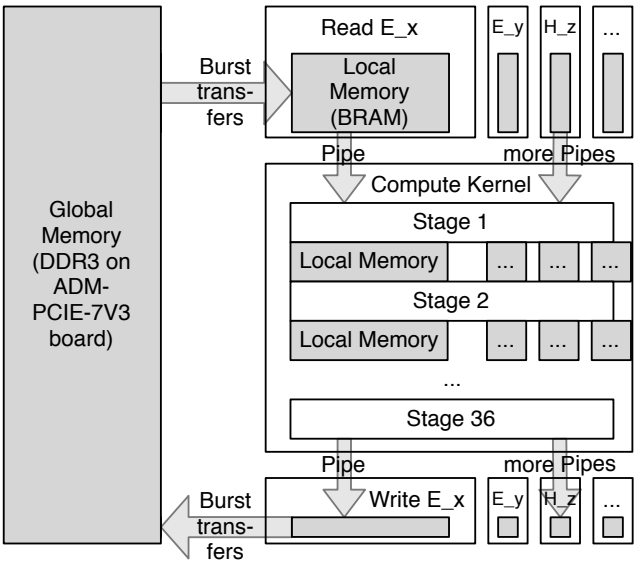

Figure 2: Illustration of final SDAccel-based design. Eight read and write kernels decouple the compute kernel from all off-chip memory accesses. Adjacent pipeline stages inside the compute kernel are coupled through individual local buffers.

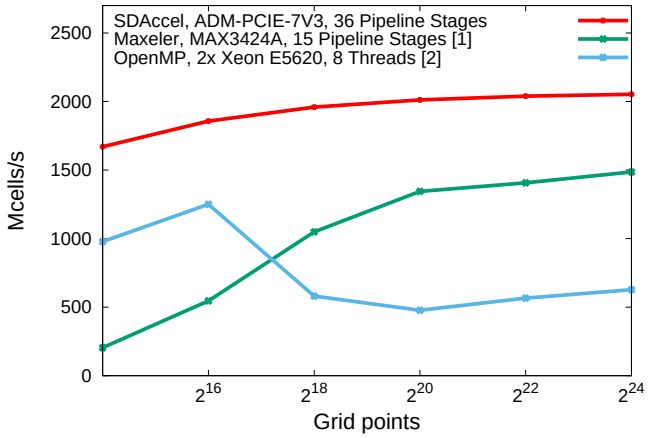

Figure 3: Throughput comparison of SDAccel-based design with results from [1] and [2]. All designs using single precision floating point.

3. RESULTS

We performed the hardware generation of the SDAccel (version 2016.1) tool flow for several designs with different unrolling factors. The tool aims at a clock frequency of 200MHz and automatically lowers the frequency of the resulting design when timing is not met. Only designs with up to 8 pipeline stages met the frequency target of 200MHz, whereas the largest successfully implemented design contains 36 pipeline stages and runs at 140MHz. As critical resource, it uses 79.4% of the available logic slices. The synthesis flow for this design completed in around 9 hours and was the only one of around a dozen instances that reached 140Mhz.

Figure 3 presents the throughput of this design for different problem sizes and compares it to two alternative implementations of the same microdisc cavity FDTD simulation. The OpenCL-based SDAccel design outperforms for all investigated grid sizes both our conceptually very similar implementation on a Maxeler FPGA platform [1] and the previous OpenMP-based cache-optimized 8-core software implementation [2]. With a throughput of 2053 MCell updates per second for the largest tested input size, the SDAccel-based design reaches 81% of the peak compute throughput of 36 pipeline stages with an initiation interval of 2, running at 140MHz. In contrast, the Maxeler design contains 15 pipeline stages with an initiation interval of 1, running at 100MHz. It contains much deeper compute pipelines and buffers, which involve reduced efficiency for small simulation grids, but allow it to come much closer to its theoretical peak compute throughput for large simulation grids.

In the SDAccel design, the buffer size was intentionally limited to ensure functional correctness also for small grid sizes by preventing updated data to still be present in the compute pipeline when it is supposed to be read in from global memory for the next iteration. With a different synchronization mechanism or with a design explicitly synthesized for large simulation grids, performance closer to the computational peak throughput might be possible at the cost of additional BRAM usage.

4. CONCLUSION

We conclude that SDAccel can generate competitive FPGA designs, both in terms of absolute performance and compared to alternative FPGA design flows. Achieving this performance is currently only possible if the OpenCL code is manually transformed with plenty of hardware-specific knowledge and exhibits patterns that are amenable to HLS tools. Hence, the optimized OpenCL is much more complex than a functionally equivalent OpenCL baseline.

Acknowledgments

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre “On-The-Fly Computing” (SFB 901) and the Xilinx University Program (XUP).

5. REFERENCES

[1] H. Giefers, C. Plessl, and J. Förstner. Accelerating finite difference time domain simulations with reconfigurable dataflow computers. *SIGARCH Computer Architecture News*, (5):65–70, June 2014.

[2] B. Meyer, J. Schumacher, C. Plessl, and J. Förstner. Convey vector personalities – FPGA acceleration with an OpenMP-like programming effort? In *Proc. Int. Conf. on Field Programmable Logic and Applications (FPL)*. IEEE Computer Society, Aug. 2012.

[3] K. Yee. Numerical solution of inital boundary value problems involving Maxewell’s equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14:302–307, May 1966.