Offloading OpenMP Target Regions to FPGA Accelerators Using LLVM

Lukas Sommer*, Julian Oppermann*, Jens Korinth†, Andreas Koch*
*Embedded Systems and Applications Group, TU Darmstadt
{sommer, oppermann, koch}@esa.tu-darmstadt.de
†Computer Systems Group, TU Darmstadt
korinth@rs.tu-darmstadt.de

To provide sufficient compute power for future computational tasks, the use of heterogeneous systems, which incorporate one or more specialized accelerators, is indispensable.

However, programming such heterogeneous systems in a portable way requires additional effort. In general, it would be desirable to program a heterogeneous system with a single codebase portable across different systems with different accelerators available.

OpenMP is a well-established standard for the programming of parallel architectures, especially in the field of high-performance computing. In recent versions, the standard [1] has been extended to account for the trend towards heterogeneous systems. Directives have been introduced to denote regions of code that should be executed on a device in a heterogeneous system. Additionally, the standard defines directives and clauses that allow the programmer to specify which and how data should be mapped to the device memory. In combination, these features allow portable programming of heterogeneous systems with a single codebase.

The implementation of the OpenMP offloading features in LLVM, the Clang frontend and the corresponding OpenMP runtime extensions is under active development. Among others, the potential for the use of OpenMP offloading directives has been demonstrated for Nvidia GPUs [2].

Besides GPUs, FPGAs have received increasing attention as dedicated accelerators in heterogeneous systems, showcased i.a. by their use in the Amazon AWS F1-instances and Microsoft's Azure and Bing cloud. The goal of this work is to develop a compile-flow to map OpenMP target regions to FPGA accelerators based on LLVM and the Clang frontend.

Our work is based on the open-source ThreadPoolComposer-project (TPC) [3]. The project provides us with an automated flow for the execution of HLS-tools for code that should be mapped to FPGA-based accelerators. Additionally, TPC implements a generic top-level architecture which provides standardized host- and memory-connectivity to every hardware kernel. On the software-side, TPC comes with a portable library that defines an API to control the execution on the FPGA and to map data to/from the external memory on the FPGA-device.

Based on the TPC tool-flow, we have developed an automated flow to compile OpenMP target regions to FPGA accelerators. To this end, we have introduced a new target-triple for OpenMP device compilation in Clang. Target regions are extracted to standalone kernels, which are then processed by Vivado HLS, yielding an FPGA bitstream with the TPC top-level architecture and the accelerator kernel.

Additionally, we have implemented a plugin for `libomptarget` [4], which is part of the LLVM OpenMP runtime implementation. The plugin uses the portable TPC API and is responsible for mapping data to the FPGA memory and controlling the execution on the device.

Both parts of our implementation (Clang compile-flow and runtime library plugin) integrate seamlessly with the existing LLVM OpenMP offloading infrastructure and together allow to offload regions of code to specialized FPGA accelerator kernels from a single OpenMP codebase.

We have succesfully tested our proof-of-concept implementation by mapping different BLAS-kernels in OpenMP target regions to a heterogeneous system comprising a x86-CPU (Intel Core i7-6700K) and a Xilinx Virtex 7 FPGA. Data is transfered to/from the host using PCIe Gen3 and the BLAS computation is conducted in accelerator kernels on the FPGA.

Currently, our prototype is only using a single kernel instance on the FPGA. In future work we want to exploit parallel OpenMP device features, such as `omp distribute`, to distribute the computation across multiple kernel instances and improve the performance of our prototype.

References

[1] "OpenMP Application Programming Interface - OpenMP Standard 4.5," Nov. 2015.

[2] S. F. Antao, A. Bataev, A. C. Jacob, G.-T. Bercea, A. E. Eichenberger, G. Rokos, M. Martineau, T. Jin, G. Ozen, Z. Sura *et al.*, "Offloading support for OpenMP in Clang and LLVM," in *Proceedings of the Third Workshop on LLVM Compiler Infrastructure in HPC*. IEEE Press, 2016, pp. 1–11.

[3] J. Korinth, D. d. l. Chevallerie, and A. Koch, "An Open-Source Tool Flow for the Composition of Reconfigurable Hardware Thread Pool Architectures," in *2015 IEEE 23rd Annual International Symposium onField-Programmable Custom Computing Machines (FCCM)*, May 2015, pp. 195–198.

[4] Samuel Antao, Carlo Bertolli, Andrey Bokhanko, Alexandre Eichenberger, Hal Finkel, Sergey Ostanevich, Eric Stotzer, and Guansong Zhang, "OpenMP offload infrastructure in LLVM." [Online]. Available: https://github.com/clang-omp/OffloadingDesign

[5] L. Sommer, J. Korinth, and A. Koch, "OpenMP Device Offloading to FPGA Accelerators," in *2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2017, pp. 201–205.