Computationally-Intensive Subproblems in FPGA-Based Accelerator Design

Project Manager
Lukas Sommer

Researchers
Dr.-Ing. Julian Oppermann

Principal Investigator
Prof. Dr.-Ing. Andreas Koch

Project Term
2019 - 2020

Clusters
Lichtenberg Cluster Darmstadt

Additional Software
Gurobi 8.1

Institute
Department of Computer Science

University
Technische Universität Darmstadt

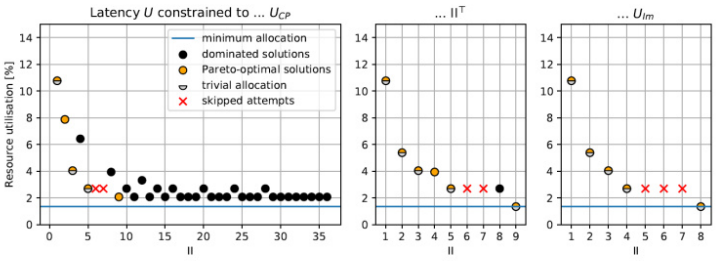


Figure 1: Results of a scheduler-driven design space exploration under different latency constraints.

Introduction

Field-Programmable Gate Arrays (FPGA) contain programmable logic elements that can accommodate application-specific accelerator modules. We research high-level synthesis (HLS) tools, e.g. translating from C to a hardware design, in order to make FPGAs accessible to non-hardware-experts. However, with the great flexibility also comes the challenge that our tools have to select designs from a vast search space. The processing performance of the Lichtenberg computer helps the HLS tools to find better hardware designs.

Methods

High-level synthesis is the automatic construction of an application-specific hardware architecture based on a behavioural description of an algorithm, e.g. in form of C code. From the many large-scale combinatorial problems that arise in high-level synthesis targeting FPGAs, we continued to focus on improvements to the state-of-the-art modulo scheduling approaches in the past year. In general, scheduling is one of the most important and time-consuming parts in high-level synthesis. Modulo scheduling, in particular, is a technique to enable loop pipelining in HLS-generated accelerators: Consider a loop (e.g. a for-loop in C). Normally, the iterations of this loop would be executed back-to-back. However, if the operations can be arranged so that dependences between the iterations still hold and resource constraints are fulfilled, subsequent iterations can be executed in a overlapping manner, therefore improving the throughput and resource utilisation in the accelerator. Modulo scheduling in practice is usually done by using heuristic algorithms that may produce non-optimal results. We proposed a novel, exact formulation of the problem using integerlinear programming (ILP). We use a commercial ILP solver that is highly tuned internally, but appears like a blackbox to our scheduler.

Therefore, in order to show that our approach is competitive or even better than existing heuristics and ILP-based approaches, we needed to performan extensive experimental campaign on a diverse set of problem instances to obtain sound and reliable data. Our research benefitted mostly from access to the large pool of identical multicore machinesin the MPI2 section. The HHLR allowed us to perform reliable wall-clock time measurements when scheduling our benchmark set of scheduling problems with different scheduling algorithms and configurations. We currently use Gurubi 8.1 as ILP solver, which can perform parts of the internal branch-and-bound algorithm in parallel. In addition, as each instance can be scheduled individually, these experiments scale extremely well on the HHLR when using jobarrays.

Results

In the past periods, we were able to show (using experimental data obtained on the HHLR) that ILP-based modulo scheduling can be practical in an HLS setting. We also developed a graph preprocessing technique that closes the gap (considering the average scheduling runtime) between a priori state-of-the-art and our approach. However, this research emphasised the distinct strengths and weaknesses of the schedulers. Our long-term goal is therefore to develop an oracle that chooses a priori the most promising scheduler for a given instances. We developed a parametrisable problem generator to create a much more diverse set of scheduling problems, and plan to use the HHLR to learn the oracle in the future. In the now ending period, we extended the scope of our scheduling technique. Instead of computing a schedule for fixed (externally given) number of operators (i.e. single-purpose functional units instantiated by the HLS tool), we can now model scheduling and operator allocation in a single, multi-objective optimisation problem. This research laid the theoretical groundwork needed to perform scheduler-driven design-space exploration.

Discussion

Our work on exact scheduling coincides with a reignited interest in the HLS community in scheduling approaches, and we believe we made meaningful contribution to the field, for which the HHLR was an indispensable tool. In the next project period, we will apply the scheduler-driven design space-exploration technique to a commercial HLS tool in order to investigate its practicability. We also plan to take on a new field of research, and bring our cutting-edge research on the design of accelerators for the inference in so-called Sum-Product Networks (SPN), a relatively new and powerful class of machine learning networks, to the HHLR. In that context, we will investigate tradeoffs between resource consumption and throughput as well as the design of low-energy accelerators for AI-on-the-edge use cases. For this purpose, we will use the automatic design-space exploration feature of our open-source framework TaPaSCo, and later experiment with the aforementioned scheduler-driven exploration for this application.

Reference

Oppermann, J.; Reuter-Oppermann, M.; Sommer, L.; Koch, A.; Sinnen, O. 2019. Exact and Practical Modulo Scheduling for High-Level Synthesis. ACM Trans. Reconfigurable Technol. Syst. 12, 2, Article 8 (May 2019), 26 pages. https://doi.org/10.1145/3317670

Last Update: 2022-04-23 17:13