Estimation and Partitioning for CPU-Accelerator Architectures

Tobias Kenter[1], Christian Plessl[1], Marco Platzner[1], Michael Kauschke [2]

[1] University of Paderborn
[2] Intel Labs Braunschweig

kenter@uni-paderborn.de

Summary

We developed an approach for studying the design space when interfacing reconfigurable accelerators with a CPU. We consider reconfigurable accelerators that are controlled by a CPU via a direct low-latency interface but also have direct access to the memory hierarchy. In order to investigate those, we present a framework based on the LLVM infrastructure that performs estimation of the runtime on the target architecture utilizing profiling information and code analysis. It finds the optimal hardware/software partitioning by forming a corresponding ILP problem. The process is fully automated and enables the evaluation of various architecture parameters.

1 Detail

We investigate a class of CPU-accelerator architectures, where the two compute devices have a direct interface mainly for control and are also connected by a shared memory hierarchy. The effects of different interface characteristics, cache layouts, accelerator sizes and ratios of compute capabilities of CPU to accelerator are subject to our design space exploration.

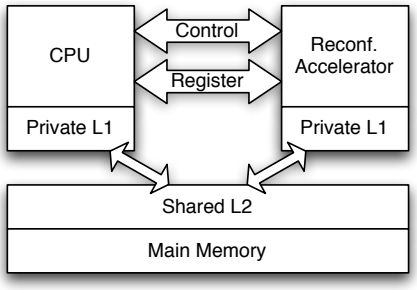

Figure 1: Example of Architecture

For each architecture configuration and code mapping, we estimate the total execution time based on analysis and profiling of the LLVM intermediate representation of the code. We consider execution times for instructions on register operands, memory access latencies and direct communication latencies.

While the architecture itself is well suited for concurrent execution on CPU and accelerator, the estimation model is currently limited to a sequential execution with the control flow alternating between the two devices. This way, we determine a lower bound of the achievable speedups in two regards: first, the dynamic power consumption of the combined CPU and accelerator depends only on their respective dynamic power consumption and not on the sum of both. Second, the presented speedups do neither depend on the existence of task level or coarse grained loop level parallelism in the benchmarks, nor on their exposition by the programmer or identification by automated tools.

Instead, the analysis focuses on the existence of kernels or parts of kernels in the benchmarks that fit the size constraints of the accelerator and promise speedups after considering control flow and data flow overheads. Both presented aspects of the sequential execution model also favor a replication of several CPU-accelerator pairs into a multicore architecture.

The execution time of in-register instructions is estimated by an average value. We have taken steps towards a calibration of these timings for different instruction classes on modern CPUs and to evaluate the estimation accuracy of the calibrated model.

In previous publications, we found mappings of the code to CPU and accelerator with a greedy heuristic [1] and incorporated information about the control flow to improve the heuristic [2]. We now compute the optimal partitioning for a given estimation using ILP solving and find that the improved heuristic leads to optimal or almost optimal solutions in most cases.

Publications

[1] Tobias Kenter, Marco Platzner, Christian Plessl, and Michael Kauschke. “Performance estimation for the exploration of CPU-accelerator architectures.” *Proc. Workshop on Architectural Research Prototyping (WARP)*, June 2010.

[2] Tobias Kenter, Christian Plessl, Marco Platzner, and Michael Kauschke. “Performance estimation framework for automated exploration of CPU-accelerator architectures.” *Proc. International symposium on Field programmable gate arrays (FPGA)*, pages 177–180. ACM, 2011.