Opportunities for deferring application partitioning and accelerator synthesis to runtime
– Extended Abstract –

Tobias Kenter, Gavin Vaz, Heinrich Riebler and Christian Plessl
Paderborn University, Germany

1 Static hardware/software design flows

Design flows for reconfigurable computing systems typically follow a static hardware/software codesign approach. In this flow, the identification of computationally expensive kernels that are potential targets to be offloaded to reconfigurable accelerators are identified by the developer with application profiling tools. Typically, these functions correspond to frequently executed loops in the application. The selection of the most suitable kernels for acceleration by FPGAs and the translation of these kernels to application-specific accelerators implemented on the FPGA also involves the decision and frequently also substantial manual work by the developer even if high-level synthesis tools are used. This static partitioning and design flow is proven and well understood, but it originates from the era of hardware/software codesign for ASICs and inherits at least two major assumptions from this era that no longer hold for reconfigurable devices.

1. Accelerator design and production is assumed to be very time consuming (ASIC fabrication) and costly. Hence, static design flows spend a lot of effort on taking good decisions guided by a developer at design time to make sure to get highly efficient, fast and small ASIC designs.
2. The application workload is assumed to be static and known in advance, because ASICS have a fixed function determined at design time.

2 FPGAs change the previous assumptions

FPGAs enable orders of magnitude faster design cycles because they can gain application-specific functionality just by programming them. Also, reconfiguration allows for reusing the available FPGA resources at virtually no additional cost except for additional configuration storage. Reconfiguration obviates the need to know the exact workload and usage pattern at the system design time. Instead, new functionality can be deployed whenever new applications or usage patterns emerge. Further, reconfiguration of FPGAs even offers the opportunity to build computing systems that adapt themselves to the concrete needs of the application that actually occur at runtime instead of relying on conservative decisions that have been determined from estimations at design time. To this end, the methods for application partitioning and accelerator generation need to depart from the static codesign approach.

3 Deferring application partitioning and accelerator generation to runtime

In our research, we have explored approaches to enable accelerator generation at or near application runtime and different techniques to manage actual offloading decisions inside an application or on the system level.

We have demonstrated how instruction-programmable [1] and structurally defined [2] overlay architectures implemented on top of FPGAs can serve as target for fast and automated tools chains to extract kernel loops and generate accelerator configurations. In [3], we have quantified the overheads of the instruction-programmable vector processor overlay compared to individually customized kernel implementations. Besides the general feasibility of fast compilation or synthesis for overlays, all three publications underline that the concrete acceleration potential depends both on the respective kernel pattern and arithmetic intensity, but also on loop trip counts and data transfers. In [4], we have explored the potential of automatically embedding individual offloading decisions into the application code. By analyzing the runtime behavior of popular benchmarks, we have demonstrated that for many loops, a well informed offloading decision that is not possible at compile time can be taken directly prior to the loop execution.

The EU FP7 project SAVE aims to integrate such concepts into a comprehensive framework for heterogeneous platforms. In [5], we demonstrate the integration of just-in-time (JIT) compilation and offloading into a common runtime system with an orchestrator that monitors application progress and tries to match it to given throughput goals. To this end, the decision process is moved from the scope of the application to the central orchestrator that jointly manages all system resources. In [6], we have also explored such a system-level decision mechanism through a heterogeneous scheduler that tries to maximize system throughput for a set of tasks without real-time constraints. The results underline the need for runtime migration mechanisms in order to fully utilize heterogeneous resources, because otherwise individual resource assignment decisions can not be adapted to changing system states.

In summary, our work demonstrates the manifold potential of deferring previously static codesign steps to application runtime. Further research is needed on concrete decision mechanisms and the interplay of application-internal and system-wide decisions. Accelerator generation has been demonstrated for different kernel patterns and target architectures, but it remains a research challenge to increase the scope of supported application classes and to reduce the performance gap to fully customized kernels.

Acknowledgement

This work was partially supported by the European Union Seventh Framework Programme under grant agreement no. 610996 (SAVE) and the German Research Foundation (DFG) within the Collaborative Research Centre “On-The-Fly Computing” (SFB 901).

References

[1] Tobias Kenter, Gavin Vaz, and Christian Plessl. Partitioning and vectorizing binary applications for a reconfigurable vector computer. In *Proc. Int. Symp. on Reconfigurable Computing: Architectures, Tools, and Applications (ARC)*. Springer, April 2014.

[2] Lukas Funke. An LLVM based toolchain for transparent acceleration of digital image processing applications using FPGA overlay architectures. Master’s thesis, Paderborn University, Department of Computer Science, 2015.

[3] Tobias Kenter, Henning Schmitz, and Christian Plessl. Exploring trade-offs between specialized kernels and a reusable overlay in a stereo matching case study. *Int. Journal of Reconfigurable Computing (IJRC)*, 2015.

[4] Gavin Vaz, Heinrich Riebler, Tobias Kenter, and Christian Plessl. Potential and methods for embedding dynamic offloading decisions into application code. *Computers and Electrical Engineering*, June 2016.

[5] Gianluca C. Durelli, Marco D. Santambrogio, Gavin Vaz, Christian Plessl, Heinrich Riebler, Ettore M. G. Trainiti, and Cristiana Bolchini. Using just-in-time code generation for transparent resource management in heterogeneous systems. In *Proc. Int. Forum on Research and Technologies for Society and Industry (RTSI)*, September 2016.

[6] Achim Lösch, Tobias Beisel, Tobias Kenter, Christian Plessl, and Marco Platzner. Performance-centric scheduling with task migration for a heterogeneous compute node in the data center. In *Proc. Design, Automation and Test in Europe Conf. (DATE)*. EDA Consortium, March 2016.