Designing Domain Specific Heterogeneous Manycores from Dataflow Programs

Extended Abstract

Süleyman Savas
School of Information Technology, Halmstad University
Halmstad, Sweden
suleyman.savas@hh.se

ABSTRACT

We propose a design method to develop domain-specific heterogeneous manycores by extending simple cores with custom instructions based on application requirements. We develop tools to automate the design method and facilitate development of the manycores. The tools generate competitive hardware/software co-design in terms of performance and hardware resource usage.

ACM Reference Format:
Süleyman Savas. 2018. Designing Domain Specific Heterogeneous Manycores from Dataflow Programs: Extended Abstract. In *Proceedings of 47th International COnference on Parallel Processing (ICPP 2018).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The requirements on the performance of computers continuously increase with the emerging applications of each era. Machine learning, baseband signal processing, radar signal processing, high definition audio/video processing, financial applications as well as many other scientific applications are the performance demanding applications of the current era.

The common trend in the current multi and manycore architectures is homogeneity on the same chip where the processing units (cores) are identical. However, the target applications have a heterogeneous structure in which the tasks are different and require different hardware resources for efficient execution. Therefore they require heterogeneous architectures. Heterogeneity can be introduced in many different forms. A simple form is simple cores with different extensions to execute different tasks efficiently.

We propose a design method to develop manycore architectures which can execute a domain of applications efficiently by utilizing cores specialized on computationally intensive tasks in the applications. Specialization increases the efficiency however, decreases the flexibility. Therefore we propose usage of a number of specialized cores on the same chip to support efficient execution of a number of applications in the same domain. Additionally, the cores can still be used for general purposes, however they will not be as efficient. Instead of exploring the design space to find a suitable architecture for the target applications we propose to build the architectures based on the requirements of the applications. With this method, we aim to develop task specific extensions and integrate them into simple cores. The cores can run the control logic and delegate the computations to the accelerators.

A common feature of the target applications is having a chain of tasks processing a continuous stream of data. This computation model can be implemented easily with dataflow programming model. Therefore we use this model for implementing the target applications. As the language, we use CAL actor language [4] which is a modern dataflow language and used in standardization of MPEG.

ICPP 2018, August 13-16, 2018, Oregon, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 METHODOLOGY

The design method aims to build an efficient architecture for a specific domain of applications by integrating task-specific custom hardware into simple cores. The steps of the proposed method are as below:

- Application development
- Analysis and code generation
- Accelerator integration
- System integration

The architecture configurations are based on the requirements of the applications. Therefore, the design method starts from the application description. The application should be developed in a language with support for parallelism. Having the application divided into separate tasks/functions/actions makes it easy to identify the compute intensive blocks.

The purpose of the task-specific custom hardware is to execute the compute intensive parts (hot-spots) of the applications. In order to find these hot-spots, the application needs to be analyzed. Analysis methods are usually divided into static and dynamic analysis methods [6–8]. Static analysis can provide the number of operations and operands in an application. This information can be used to estimate execution times for applications which have constant or static behavior. However, dynamic analysis is required to obtain execution measures for applications with dynamic behavior, and to define the most frequently executed, computationally-intensive code blocks. Useful analysis information for hot-spot identification includes execution rate, number of operations and operands, complexity of the operations, and execution time.

After identifying the hot-spot of the application, hardware code should be generated for the accelerator that will execute the hot-spots. The rest of the application can be converted to a software language that is supported by the native compiler of the target core.

The target core executes the application and delegates the hot-spot execution to the generated accelerator. Thus the accelerator must be integrated to the target core. It can be connected to the data bus and be memory mapped, or it can be connected through custom interfaces, or even act as an instruction extension. The integration method can vary based on aspects such as the features of the base core, architectural requirements, and application requirements.

The prior steps of the design flow produce tiles consisting of processing core and accelerator for individual tasks in the applications. To execute the entire application, or a set of applications, the tiles need to be connected to each other to form a manycore architecture. In particular, for dataflow applications that have significant amount of core-to-core (or tile-to-tile) communication, the tiles must be connected with a proper infrastructure that supports tile-to-tile communication. The analysis step can provide configuration information to be used in this step such as communication buffer sizes, number of cores and memory requirements.

3 PRELIMINARY RESULTS

We have generated tiles using the proposed method. Our design choices are as follows: CAL actor language [4] for application development, TURNUS framework [3] for analyzing the application, Cal2Many framework [5] for generating hardware/software co-design, RISC-V [9] cores (rocket core) as the base core, Chisel [2] as the hardware description language, C as the native language for RISC-V core, rocket chip generator [1] for integrating the accelerator to the core and generating a single tile.

Cal2Many framework had support for generating different software designs targeting different platforms. We added the support for generating hardware design to achieve co-design of hardware and software. This extension takes CAL application and generates hardware code (in Chisel) for the hot-spot software code (in C) for the rest of the application.

We have targeted two applications namely QR decomposition, that is usually used for avoiding matrix inversions typically in baseband processing, and Autofocus criterion calculation which is used in radar signal processing. The applications are implemented in CAL, analyzed via TURNUS and hot-spots are identified. Hardware code in Chisel is generated to execute the hot-spots and the rest of the applications are converted to C with custom instructions that can be compiled with the native compiler for rocket core. As a reference point for comparison, we have implemented the accelerators manually as well. The applications are executed 3-4 times faster with the accelerators while the performance degradation is between 0 and 4 % when the accelerator is generated instead of being manually implemented. Generated accelerators use approximately 0 to 10% more resources while achieving almost the same clock frequency when compared to manually implemented accelerators.

4 CONCLUSIONS

We proposed a method to design domain-specific manycore architectures and realized all the steps of the methodology except the last one which is integration of tiles to form a manycore. Our results show that with task specific extensions the cores can achieve 3-4x higher performance. The automatic generation of these extensions facilitates development of the tiles and manycores while achieving competitive performance and resource usage results.

REFERENCES

[1] Krste Asanovic, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, et al. 2016. The Rocket Chip Generator. (2016).

[2] Jonathan Bachrach, Huy Vo, Brian Richards, Yunsup Lee, Andrew Waterman, Rimas Avižienis, John Wawrzynek, and Krste Asanović. 2012. Chisel: constructing hardware in a scala embedded language. In *Proceedings of the 49th Annual Design Automation Conference*. ACM, 1216–1225.

[3] Simone Casale-Brunet, M Wiszniewska, Endri Bezati, Marco Mattavelli, Jörn W Janneck, and Massimo Canale. 2014. Turnus: An open-source design space exploration framework for dynamic stream programs. In *Conference on Design and Architectures for Signal and Image Processing (DASIP)*. IEEE, 1–2.

[4] Johan Eker and Jorn W. Janneck. 2012. Dataflow programming in CAL – balancing expressiveness, analyzability, and implementability. In *Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), 2012*. IEEE, 1120–1124.

[5] Essayas Gebrewahid, Mingkun Yang, Gustav Cedersjo, Zain Ul-Abdin, Veronica Gaspes, Jörn W Janneck, and Bertil Svensson. 2014. Realizing efficient execution of dataflow actors on manycores. In *Proceedings of the 12th IEEE International Conference on Embedded and Ubiquitous Computing (EUC), 2014*. IEEE, 321–328.

[6] Jorn W Janneck, Ian D Miller, and Dave B Parlour. 2008. Profiling dataflow programs. In *International Conference on Multimedia and Expo*. IEEE, 1065–1068.

[7] Małgorzata Michalska, Jani Boutellier, and Marco Mattavelli. 2015. A methodology for profiling and partitioning stream programs on many-core architectures. *Procedia Computer Science* 51 (2015), 2962–2966.

[8] Chao Wang, Xi Li, Huizhen Zhang, Aili Wang, and Xuehai Zhou. 2017. Hot spots profiling and dataflow analysis in custom dataflow computing SoftProcessors. *Journal of Systems and Software* 125 (2017), 427–438.

[9] Andrew Waterman, Yunsup Lee, David A Patterson, and Krste Asanovic. 2011. The risc-v instruction set manual, volume i: Base user-level isa. *EECS Department, UC Berkeley, Tech. Rep. UCB/EECS-2011-62* (2011).