LoopInvader: A Compiler for Tightly Coupled Processor Arrays

Alexandru Tanase, Michael Witterauf, Éricles Sousa, Vahid Lari, Frank Hannig, and Jürgen Teich
Hardware/Software Co-Design, Department of Computer Science
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

Continuous technology miniaturization allows to build massively parallel embedded computer architectures within a single silicon chip. Programming that leverages the abundant parallelism in such architectures, however, is very difficult, tedious, and error-prone. Thus, compiler support is paramount. We therefore present LoopInvader, a loop compiler for a particular class of massively parallel processor arrays: Tightly coupled processor arrays (TCPAs) [2].

TCPAs consist of a two-dimensional array of VLIW processing elements (PEs) and several peripheral components that enable zero-overhead loops. In particular, a global controller (GC) generates synchronized control signals that govern the control flow of the PEs, removing control overhead from the loops; address generators (AG) produce the necessary addresses for feeding the PEs with data from reconfigurable buffers, removing addressing overhead. Moreover, the PEs are connected to their neighbors via a circuit-switched interconnection network that is reconfigurable at runtime to optimally accommodate the running application.

Figure 1 depicts an overview of our high-level programming methodology. We describe programs in a domain-specific functional language called PAULA that is based on *dynamic piecewise linear/regular algorithms (DPLA)* [3], a mathematical representation of loop programs. For the parallelization and mapping of such algorithms onto TCPAs we use *symbolic partitioning* techniques [5] in the polyhedral model: Instead of using fixed tile sizes, our symbolic partitioning technique is able to keep the size of the input data and the number of PEs symbolic until runtime. This provides applications more flexibility and is important in resource-aware computing paradigms such as *invasive computing* [4]. Other approaches are both time-consuming (e. g., dynamic recompilation) and costly (e. g., pre-compiling multiple variants) on embedded systems.

After mapping, the compiler generates a configuration stream comprising assembly code for the PEs, interconnect configuration, address generator configuration and global controller configuration [1]. Because the PEs offer only small instruction memories, we developed an approach to generate code that is independent of the problem size [1]. This is achieved by finding processors and program blocks within processors that share the same code and appropriately combining it into loops.

As the PEs are interconnected by a circuit-switched interconnect, the compiler also generates all necessary configuration information. For preserving a given schedule of instructions, code for the GC is generated such that the repetitive execution of each unique program block does not cause any extra cycles.

[1]Because the compiler is a work in progress, only static code generation is demonstrated.

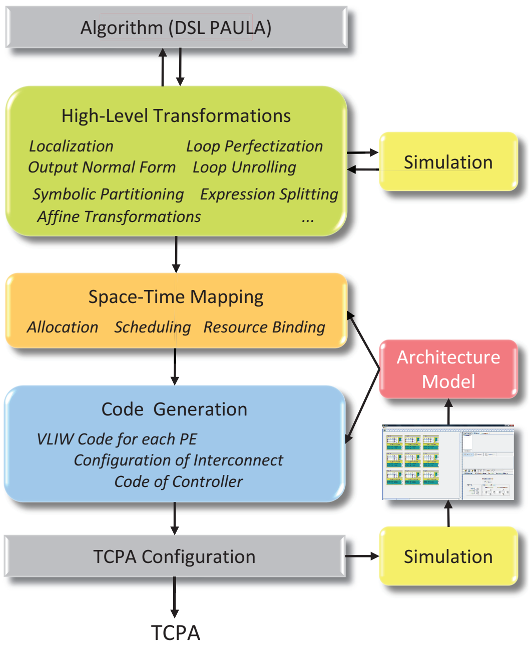

Fig. 1. Overview of LoopInvader, a compiler framework for TCPAs [2].

This orchestration enables the execution of nested loop programs with *zero-overhead loop*, not only for innermost loops but also for all static conditions in arbitrary multi-dimensional data flow.

Finally, the generated configuration stream can be loaded either onto an actual TCPA synthesized on an FPGA or into a cycle-accurate simulator that we developed in tandem with the architecture and compiler for testing and verification.

Acknowledgment

This work was supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Invasive Computing" (SFB/TR 89).

References

[1] S. Boppu, F. Hannig, and J. Teich. Loop program mapping and compact code generation for programmable hardware accelerators. In *Proceedings of the 24th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 10–17. IEEE.

[2] F. Hannig, V. Lari, S. Boppu, A. Tanase, and O. Reiche. Invasive tightly-coupled processor arrays: A domain-specific architecture/compiler co-design approach. *ACM Transactions on Embedded Computing Systems (TECS)*, 13(4s):133:1–133:29, Apr. 2014.

[3] F. Hannig and J. Teich. Dynamic piecewise linear/regular algorithms. In *Parallel Computing in Electrical Engineering, 2004. PARELEC 2004. International Conference on*, pages 79–84. IEEE, 2004.

[4] J. Teich. Invasive algorithms and architectures. *it - Information Technology*, 50(5):300–310, 2008.

[5] J. Teich, A. Tanase, and F. Hannig. Symbolic parallelization of loop programs for massively parallel processor arrays. In *Proceedings of the 24th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 1–9. IEEE, 2013.