High-level Synthesis for Data-intensive Applications

Harald Devos and Dirk Stroobandt*
Electronics and Information Systems Department (ELIS)
Ghent University, Belgium
Harald.Devos@elis.UGent.be

1. INTRODUCTION

Most high-level synthesis tools focus on exploiting parallelism. However, for data-intensive applications the memory bandwidth or latency may become a bottleneck and improving data access patterns becomes as important as exploiting parallelism. To minimize the number of accesses to off-chip memory, a memory hierarchy is needed to reuse data in on-chip memories. The resulting performance heavily depends on the locality of the data accesses. Loop transformations are a means to improve this locality but also may have a large impact on the loop control complexity and thus on the control hardware. A close integration of loop transformations and hardware generation is needed to tackle this problem.

We present a methodology, supported by tools, to explore loop transformation variants and study their impact on both data access patterns and generated control hardware. The construction of an application-specific memory system is supported, which offers more than solely a reduction of bandwidth requirements.

2. LOOP TRANSFORMATIONS

A part of a program where the control flow is independent of the processed data is called a SCoP (Static Control Part), and can be represented in a polyhedral model [6]. Loop transformations are now reduced to matrix and vector operations. This representation overcomes a lot of limitations of syntactic representations and facilitates the composition of a long sequence of transformations [6].

For guiding the composition of loop transformation sequences we make use of SLO (Suggestions for Locality Optimizations) [2, 4]. This tool investigates the data reuses within a program execution and gives hints for potential optimizing loop transformations.

3. LOOP CONTROLLER VARIANTS

We have extended CLooG [1], a library generating software code from a polyhedral representation of a SCoP (without statement information), towards CLooGVHDL generating synthesizable VHDL descriptions of hardware controller blocks.

*This research is supported by the I.W.T. grant 060068 and the F.W.O. grant G.0475.05.

DAC HLS workshop '08 Anaheim, California USA

CLooG has a number of code generation optimization options which also have a large influence on the generated hardware controllers. Next to this, the generated architecture offers several options to trade off area, clock frequency and cycle count of an implementation as demonstrated in [3].

4. APPLICATION-SPECIFIC MEMORY SYSTEM

For processors, optimizations are done for a fixed memory hierarchy, while on FPGAs or ASICs the memory structure can be made application-specific. Furthermore, buffer memories not only reduce the number of off-chip accesses or hide the external memory latency but can also augment the on-chip bandwidth through parallel access of multiple buffers or simplify address expressions by remapping data [5]. The final target, i.e. fully automatic generation of the optimal memory system for a given application, with data mapping and scheduling of burst transfers, will probably not be reached in the near future. However, many useful techniques have been developed in the context of software but are not integrated yet with HLS tools. In the mean time, we offer a step-by-step methodology to construct such a memory system directed by the user. Special care is taken of the reusability of design modules and the simplification of addresses to improve the performance.

5. REFERENCES

[1] C. Bastoul. Code generation in the polyhedral model is easier than you think. In *PACT'13 IEEE International Conference on Parallel Architecture and Compilation Techniques*, pages 7–16, Juan-les-Pins, September 2004.

[2] K. Beyls. SLO – Suggestions for Locality Optimizations. `http://slo.sourceforge.net/`.

[3] H. Devos. *Loop Transformations for the Optimized Generation of Reconfigurable Hardware*. PhD thesis, Ghent University, February 2008.

[4] H. Devos, K. Beyls, M. Christiaens, J. Van Campenhout, E. H. D'Hollander, and D. Stroobandt. Finding and applying loop transformations for generating optimized FPGA implementations. *Trans. on High Performance Embedded Architectures and Compilers I, LNCS*, 4050:159–178, 2007.

[5] H. Devos, D. Stroobandt, and J. Van Campenhout. Building an application-specific memory hierarchy on FPGAs. In *2nd HiPEAC Workshop on Reconfigurable Computing*, pages 53–62, Göteborg, Sweden, January 2008.

[6] S. Girbal, N. Vasilache, C. Bastoul, A. Cohen, D. Parello, M. Sigler, and O. Temam. Semi-automatic composition of loop transformations for deep parallelism and memory hierarchies. *Int. J. Parallel Program.*, 34(3):261–317, 2006.