Loop transformations for generating scalable hardware

Harald Devos

Promoter(s): Dirk Stroobandt, Jan Van Campenhout

I. Introduction

Multimedia applications emerge on all kinds of devices, from small mobile systems up to desktop computers with varying ranges of quality, resolution, power and other requirements.

They are part of a large class of signal processing systems that often need hardware acceleration. The QoS (Quality of Service) depends on the available hardware.

Designing the hardware for a lot of different platforms is a huge task. The occurance of reconfigurable hardware (e.g. FPGAs), that can be configured with different hardware designs (e.g. each with different QoS, power and area consumption) makes the number of designs to be made even higher. Tools have to be developped to generate different scaled versions of a design.

Most of the algorithms are not only computationally intensive but also data intensive. The memory bandwidth and latency become a bottleneck. Transformations are needed to improve the spatial and temporal locality of the memory accesses. We will focus on programs where most of the execution time is spent in a small part of the code, consisting of a set of nested loops. Almost all signal processing algorithms contain such *hot* loop nests.

II. The polyhedral model

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 the polyhedral model. The iteration domain of each statement is described as a union of convex sets of points in $\mathbb{N}^N$ (= a polyhedron), where N is the number of iterators. By extending the iteration vector to a vector with dimension 2N+1 the ordering of statements can be described and manipulated. Loop transformations are reduced to linear transformations on the polyhedrons and on the ordering vectors.

This representation overcomes a lot of limitations of syntactic representations and facilitates the composition of a sequence of transformations[1]. A lot of techniques for optimizing software for (multi-)processors use this model but it is rarely used with the purpose of generating hardware.

III. Transformations

All modern FPGAs contain on-chip memory blocks that can be accessed in a short time, typically within one clock cycle. A larger but slower memory is often placed next to the FPGA. Accesses to this external memory consume more power and time and should be minimized and bundled into bursts.

H. Devos is with the Electronics and Information Systems Department, Ghent University (UGent), Gent, Belgium. He is supported by the fund for Scientific Research Flanders(F.W.O.). E-mail: Harald.Devos@UGent.be .

Transformations increasing the temporal and spatial locality should be used to maximize the reuse of the data stored in the on-chip buffers.

Processor architectures have a similar memory hierarchy and part of the optimization algorithms for software can be used here. There are however some differences between processor and FPGA/ASIC-architectures:

- A processor mostly has a vertical memory hierarchy, one memory on each of the cache levels. On a FPGA the hierarchy is more horizontal, one large external memory and a lot of on-chip memories that can be accessed in parallel.
- Software optimization transforms the data access pattern for a fixed memory structure while on a FPGA the memory hierarchy can be made application specific.
- On a processor, control and data flow are sequentially executed on the same unit. On a FPGA or ASIC control and data path can work in parallel.

We would like to investigate where the existing optimization algorithms for software can be used, with the purpose of generating hardware and where adaptations or new algorithms are needed. We also believe that transformations in the polyhedral model can be used with the goal of making scaled versions of a design.

IV. Path to hardware

An implementation of an Inverse Discrete Wavelet Transform on a FPGA board was first made by hand [2][3]. The memory bandwidth was indeed the bottleneck. The ad hoc methods used for this application should be turned into systematic techniques.

Existing techniques to create hardware starting from a software description, have difficulties with memory accesses, e.g. SPARK[4], or support only a restricted set of IO structures (e.g. ImpulseC).

Cloog[5], a library generating C or FORTRAN starting from a polyhedral representation of a SCoP (without statement information), was extended to generate a synthesizable VHDL description of a hardware controller block.

The next step consists of creating a tool to generate VHDL for the hardware implementing the statements and memories. Also the possibility of creating parallelism should be added to the tools. Once a toolchain from software to hardware exists the effects of loop transformations on hardware performance can be investigated more easily. The tool chain will also make it possible to create scaled versions of a design.

References

[1] Albert Cohen, Sylvain Girbal, David Parello, Marc Sigler, Olivier Temam, and Nicolas Vasilache, “Facilitating the search for compositions of program transformations.,” in *ACM Int. Conf. on Supercomputing (ICS’05), Boston, Massachusetts.*, June 2005.

[2] D. Stroobandt, H. Eeckhaut, H. Devos, M. Christiaens, F. Verdicchio, and P. Schelkens, “Reconfigurable hardware for a scalable wavelet video decoder and its performance requirements,” *Computer Systems: Architectures, Modeling, and Simulation*, vol. 3133, pp. 203–212, July 2004.

[3] Harald Devos, Hendrik Eeckhaut, Benjamin Schrauwen, Mark Christiaens, and Dirk Stroobandt, “Ever considered SystemC ?,” in *Proceedings of the 15th ProRISC Workshop*, Veldhoven, November 2004, pp. 358–363.

[4] Sumit Gupta, Nick Savoiu, Nikil Dutt, Rajesh Gupta, and Alex Nicolau, “Using global code motions to improve the quality of results for high-level synthesis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,*, vol. 23, no. 2, pp. 302–312, February 2004.

[5] 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*, Juan-les-Pins, september 2004, pp. 7–16.

[6] “The RESUME project: Reconfigurable Embedded Systems for Use in Scalable Multimedia Environments,” http://www.elis.UGent.be/resume.