Loop Transformations for the Optimized Generation of Reconfigurable Hardware

Harald Devos, Ghent University

Abstract—**Current high-level design environments offer little support to implement data-intensive applications on heterogeneous-memory systems; they rather focus on parallelism. This thesis addresses the memory hierarchy problem to high-level transformations of loop structures and the integration of these transformations in the hardware design flow. This work focuses on three aspects: (1) the composition of long transformation sequences, (2) the search for bounds on polynomials and quasi-polynomials over discrete domains for static program analysis and finally (3) the generation of hardware descriptions from a polyhedral representation, used to describe and apply loop transformations.**

I. Problem Definition

Multimedia applications are an example of applications that are not only computation-intensive but also data-intensive, which means a large amount of memory is needed. To implement data-intensive applications on FPGAs (Field Programmable Gate Arrays) off-chip memory is needed. The accesses to this memory are slower (bandwidth and latency) than accesses to on-chip memory and are a potential bottleneck.

A memory hierarchy should be constructed to decrease the number of off-chip transactions. A data element that is copied to an on-chip buffer can be reused several times without external memory accesses. The buffers have a limited size and cannot contain a copy of all data at the same time. Therefore, the different accesses to a data element should be close together in time, i.e. exhibit a good temporal locality. Loop transformations are a means to improve the data locality by changing the execution order of computations and data accesses. This technique is commonly used for software optimizations, in particular optimization of the cache behavior. Current high-level synthesis environments for hardware design lack support to implement data-intensive applications on heterogeneous memory systems. They focus rather on parallelism than on locality.

II. Sequences of loop transformations

We restrict ourselves to loop nests in which all loop bounds are linear expressions in some integer parameters and the iterators of surrounding loops. As a result, the iteration domain of a program statement, i.e. the set of all possible values of the iteration vector (the vector composed of the iterators of all surrounding loops), can be described as a $\mathbb{Z}$-polytope, i.e. the intersection of an integer lattice and a rational polytope. The program representation based on this property is called the polyhedral model.

In a polyhedral representation loop transformations are described as operations on matrices and vectors. This eases the composition of transformation sequences in comparison with a textual or abstract syntax tree representation. In this thesis the influence of the order of transformation steps is studied together with the possibility to build long transformation sequences by combining short sequences.

H. Devos is with the Electronics and Information Systems Department, Ghent University, Belgium. E-mail: Harald.Devos@UGent.be, URL: http://www.elis.UGent.be/~hmdevos.

III. Bounds on quasi-polynomials

Static program analysis involves the estimation or computation of program properties without execution. This can be used to guide the choice of loop transformations to apply. For programs that can be represented in the polyhedral model, many analysis problems can be reduced to counting the number of integer points in a polytope or the number of integer solutions to so-called Presburger formulas, i.e. systems of inequalities connected with logical operators and quantifiers. The solution of such a counting problem can be expressed as a (piecewise) Ehrhart quasi-polynomial. In some cases an upper bound on a quasi-polynomial over a discrete domain is needed. If, for example, the number of live data elements is expressed as a function of the point of execution of a program, then the minimal memory requirement of that program is the maximum over all points of execution.

Several novel methods to find bounds on quasi-polynomials are presented in this thesis and compared with existing methods. The presented methods take advantage of the fact that for large domains or small degrees the continuous-domain extrema of polynomials are a good approximation of the discrete-domain extrema. For small domains the straightforward method of evaluating the (quasi-)polynomial in each point is still faster, but this solution does not scale for larger domains. With a simple selection mechanism that introduces almost no overhead, hybrid methods can be constructed that combine the strengths of different methods.

IV. Hardware generation

Loop transformations not only influence the data transfers but also the control complexity of an implementation. The impact on the hardware performance can typically only be quantified after refinement to a synthesizable level. This hinders an exploration of the loop transformation space. Therefore, it would be beneficial to integrate loop transformations in high-level synthesis tools. This thesis presents a hardware architecture, which allows to generate hardware starting from a polyhedral representation. Different trade-offs between area and clock speed are investigated. Our code generator, CLooGVHDL, has been tested by generating variants of an inverse discrete wavelet transform. The results outperform those of the commercial high-level synthesis tool Impulse C and are competitive to those of the Celoxica Handel-C compiler.

References

[1] Harald Devos, *Loop Transformations for the Optimized Generation of Reconfigurable Hardware*, Ph.D. thesis, Ghent University, February 2008.