A Hierarchical Architecture Description for Flexible Multicore System Simulation

Thomas Bruckschloegl, Oliver Oey, Michael Rueckauer, Timo Stripf, Juergen Becker
Institute for Information Processing Technologies (ITIV)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{bruckschloegl, oey, rueckauer, stripf, becker}kit.edu

***Abstract*— As processors and systems on chip in the embedded world increasingly become multicore, parallel programming remains a difficult, time-consuming and complicated task. End users who are not parallel programming experts have a need to exploit such processors and architectures, using high level programming languages, like Scilab or MATLAB. The ALMA toolset solves this problem: it takes Scilab code as input and produces parallel code for embedded multiprocessor systems on chip, using platform quasi-agnostic optimizations. The platform information is provided by an architecture description language designed for the purpose of a flexible system description as well as simulation. A hierarchical system description in combination with a parameterizable simulation environment allows fine-grained trade-offs between simulation performance and simulation accuracy**

I. Introduction

Many performance-critical applications (e. g. digital video processing, telecoms, and security applications) that need to process huge amounts of data in a short time benefit from efficient, flexible, and high performance systems on chip. Research projects such as MORPHEUS [1] and CRISP [2] have demonstrated the feasibility of such an approach and presented the benefit of heterogeneity and parallel processing on a real hardware prototype.

The European project ALMA, ***A**rchitecture oriented para**L**lelization for high performance embedded **M**ulticore systems using scil**A**b* (alma is Greek for “leap”) [3], intends to provide a full design framework for designing parallel and concurrent computing systems. The design framework relies on Scilab, an open source language for developing high-level system models. The ALMA parallel software optimization environment is combined with a SystemC simulation framework to allow an iterative optimization by using profiling information of the intermediate parallel code.

The toolset is based on an *Architecture Description Language* (ADL) that is designed for providing abstract information about the hardware structure and hardware behavior. The ADL makes use of a hierarchical description that allows for a flexible simulation of the multicore system on different abstraction levels resulting in possible fine-grain trade-offs between simulation speed and simulation accuracy.

In this paper we present the ALMA toolset enabling compilation of Scilab source code to multicore architectures. The rest of this paper is organized as follows: Section II gives an overview of the ALMA toolset including a brief description of the individual components. The ADL is introduced in Section III. In Section IV the method and use of hierarchical descriptions within the ADL is explained, followed by a description of the evaluation and simulation system in Section V. The evaluation results are presented in Section VI and a conclusion of the paper and future work is given in Section VII.

II. ALMA Toolset Overview

The ALMA toolset provides an end-to-end tool chain from Scilab [4] code to executable code on embedded multicore systems. A typical end user that will use the ALMA toolset will start the application development by implementing Scilab code and specifying an abstract description of the target architecture using the ALMA *Architecture Description Language* (ADL). The Scilab application can be augmented with additional information in a C like declaration language to provide the tool-chain with maximum sizes of variables and further data to improve the parallelization process. In this user-driven perspective, two distinct phases can be identified. The first phase includes code transformations from Scilab to an *Intermediate Representation* (IR) and optimizations based on the IR and the architecture specification. The second phase is closer to the hardware: It transforms the IR produced by the first phase to executable code for the target embedded multicore architecture. The toolset workflow is presented in Figure 1.

The first phase of the toolset operates on an intermediate representation of the Scilab source code and performs optimizations based on a multicore ADL description (see Section III). This phase consists of three big steps: The frontend, the coarse grain parallelism extraction and optimization, and the fine grain parallelism extraction.

The MatrixFrontend converts the Scilab code into sequential C code that is then transformed to the ALMA IR. Scilab an Matlab are interpreted languages that feature dynamic typing and late binding, which poses a problem for the parallelization of the code. If, for example, the size of a matrix is not known at compile-time, a more generic implementation needs to be used instead of a highly optimized parallel code. Therefore the MatrixFrontend features a type inference step which calculates the types of variables and operators at compile time and subsequently generates code that uses static data, thus omitting dynamic type checking, and generates type specific operations where possible.

The fine grain parallelism extraction works on the ALMA