ARTIST 2

Network of Excellence

IST-004527 ARTIST2:
Embedded Systems Design

Activity Progress Report for Year 2

JPRA-Cluster Integration

Architecture-aware Compilation

Clusters:

Compilers and Timing Analysis

(Compilers sub-cluster)

Activity Leader:

Prof. Rainer Leupers

RWTH Aachen University

http://www.iss.rwth-aachen.de/1_institut/dok/leupers.htm

Policy Objective (abstract)

The objective of this activity is to exploit the world-leading position and expertise of academic and industrial cluster partners in order to integrate and further develop the technology currently available with the partners, so as to provide a unified architecture-aware code-synthesis and compiler methodology to a variety of users, also beyond ARTIST2.

Table of Contents

1. Overview of the Activity

1.1 ARTIST2 Participants: Expertise and Roles

Team Leader: Reinhard Wilhelm (Saarland University)
co-leader, program analysis tools.

Team Leader: Rainer Leupers (RWTH Aachen)
co-leader, code optimization, retargetable compilation.

Team Leader: Christian Bertin (STMicroelectronics)
driver applications, resource-aware code generation.

Team Leader: Christian Ferdinand (AbsInt)
Program-Analysis Tool.

Peter Marwedel (Dortmund Univ.)
low power compilation.

1.2 Affiliated Participants: Expertise and Roles

Team Leader: Hans van Someren (ACE)
compiler development platform.

Team Leader: Francky Catthoor (IMEC)
high-level code optimization.

Team Leader: Markus Schordan, Andreas Krall (TU Vienna)
Program analysis, source-to-source transformation, code optimization.

1.3 Starting Date, and Expected Ending Date

September 1st, 2004 until unified methodology has been achieved.

1.4 Baseline

Members of this activity have comprehensive expertise in different areas of compilers for embedded systems. Contacts and cooperations are already partially in place.

The CoSy compiler platform, provided by ACE, is a state-of-the-art tool on which the common activities will build. For code-synthesis, ongoing cooperation on high-level transformations between IMEC and Dortmund will be extended.

Since compiler optimizations developed at Dortmund University have proven to be beneficial for Worst-Case Execution Time (WCET), a cooperation between Dortmund University and AbsInt has been established and will be extended in the future.

Further new or continued cooperations include STM-ACE (code optimization) and Aachen-ACE (SIMD instructions)

1.5 Problems Tackled in Year2

At Dortmund, during year 2, the implementation of the bit-true dataflow analysis at the C-source level was carried out in partial cooperation with Aachen. In addition, processor specific optimizations were also implemented. In the Dortmund-IMEC cooperation, source to source code optimizations have been investigated.

Aachen and ACE have been working together on a number of topics together including a SIMD optimisation framework and conditional execution optimisation. A prototype optimization engine is currently being finalized, and a joint paper has been accepted for publication. ACE has also provided ongoing support of ST's team in relation to interprocedural optimisation.

During year two, more work was done by ST, with partial support by ACE, on the retargetability of the compiler. It has been mainly devoted to the continuation of the effort to make the process fully dynamic.

Absint and TU Vienna have been working on the integration of PAG into the ROSE compiler infrastructure as well as on the evaluation of C++ code optimizations.

1.6 Comments From Previous Review

1.6.1 Reviewers' Comments

This revised document now incorporates the discussion of requirements analysis that was done by the cluster.

The revised document is accepted.

1.6.2 How These Have Been Addressed

No comments to be addressed here

2. Summary of Activity Progress

2.1 Previous Work

The main result achieved from the previous work in the co-operation between IMEC and Univ. Dortmund is a thorough alignment of the research objectives for the steering of locality-improving loop transformations at the source code level. For this purpose, it was crucial that the impact on the control flow complexity of the applied source code trafo could be evaluated at an early stage, before actually performing all the alternatives and compiling the resulting code on the target platform. The requirement (the WHAT specification) to tackle this problem had been defined. Based on a the WHAT specification, PhD research work had been initated at Univ. Dortmund to evaluate the alternative ways to come up with such a high-level control flow cost estimator that could base its estimate when only the source code is available (without performing any compilation). At IMEC complementary actions had been started to see how this estimator can be integrated in a loop transformation framework project that had been started up earlier (prior to the start of ARTIST2) and that is now being extended for these high-level estimators.

A cooperation was established between University of Dortmund and RWTH Aachen University, with the objective to integrate SIMD optimizations developed by the former partner into the LISATek Embedded Processor Designer based tool chain, developed by the latter partner. First, a complete tool chain, including Cosy-based C compiler, assembler and linker, was generated for the Philips Trimedia architecture. The Assembler Optimizer API, providing programming interface for developing post-pass optimizations, was extended based upon the specifications from University of Dortmund. The extensions paved the way for the integration of the bit-true dataflow analysis and the SIMD optimizations.

For the Aachen-Dortmund cooperation, after the generation of all software tools and the definition of the interfaces, the integration of the SIMD optimizations was tackled next. For this purpose, a bit-true representation of integer and register values, a semantic based control flow analysis, and a bit-true dataflow graph representation was implemented and integrated into the tools generated by LISATek model. Finally, the bit-true dataflow analysis and two proof-of-concept SIMD optimizations were integrated. Only a very limited set of additional architectural properties needs to be setup in the case of retargeting the tools to a new architecture.

During year one, ST worked on adapting the FlexCC retargetable compiler technology to deal with reconfigurable processors. Focus was on:

- the definition of user-defined built-in functions.
- the dynamic access to the processor model in from the compiler,
- the reconfiguration features for an 'extension'.
- the binary tool reconfiguration toolkit based on the LISA language.

2.2 Current Results

2.2.1 Technical Achievements / Outcomes / Difficulties encountered

The main technical outcome of the Dortmund-IMEC collaboration has been an agreement on the basic guidelines for the source to source transformations regarding static and dynamic optimizations (at design time and at run time respectively). These optimizations will target the loop transformations and memory assignment of statically and dynamically allocated data in complex memory hierarchies. The collaboration is mainly based on synchronized, individual work of each of the two partners and aims on common work through PhD research.

Dortmund-Aachen cooperation on Bit-True Data Flow Optimizations as a Processor specific Source-Level Transformation: During the last reporting period, it was found bit-true dataflow analysis and the corresponding SIMD optimizations implemented at the C source-level serve greater benefit than as a post-pass analysis into the LISATek tools, as they can be easily retargeted for different processor architecture. Currently, the data flow analysis along with three different optimizations [Fal06] is implemented for TI C6x DSP. The first optimization detects and optimizes saturated arithmetic operations. The second optimization looks for SIMD instructions for packed parallel arithmetic. Third, is strength reduction optimization which determines the number of unused bits in variables and then reduces them to the smallest data type.

ACE - Aachen Cooperation:

SIMD Support in CoSy
Aachen and ACE are working on an extensible framework to allow compiler writers to target SIMD hardware more quickly and efficiently. At present, hand optimised point solutions tend to be used in industry which are not conducive to retargeting. Such an approach is essential as part of an overall solution to support SIMD hardware generated from architectural descriptions

One-line Description of the Outcome
Work in progress – extensions to data dependency analysis, support for interprocedural pointer alignment analysis and code generation description extensions to enable SIMD retargetability.

Difficulty : One-line Description of the Difficulty Encountered
No-one has successfully been able to find a formalism or generate tools which facilitate generic retargeting of these algorithms.

Optimisation of Conditional Execution in CoSy
A dynamic programming algorithm is being implemented and tested on a number of different architectures to validate its behaviour with real world code and current high-end industrial processors.

One-line Description of the Outcome
A prototype comprising a set of optimisation engines and compiler has been constructed.

Difficulty : One-line Description of the Difficulty Encountered
As per the SIMD work, retargettability is the issue.

AbsInt - TU Vienna Cooperation:

Integration of PAG in the ROSE C++ Infrastructure and Evaluation of C++ programming styles

The ROSE-PAG integration achieved in Y1 for C was substantially extended to cover full C++ (only exlcuding Exceptions). This includes handling of templates, virtual methods, short-circuit evaluation in conditions, resolving overloaded functions, C++ namespaces, constructor and destructor calls. Based on that infrastructure an initial version of a tool for whole-program analysis (WHOPA) was implemented for performing high-level evaluation of different generic programming styles suitable for embedded systems.

Evaluation of C++ optimizations

The high-level analysis of the C++ evaluation cases shows a significant difference in code size after template instantiation and in-lining (which is crucial for the relevant codes to permit whole program optimization). The evaluation cases, although performing the same operations on test data, show a difference by a factor of six in over all codes size including library codes.

The generated assembly code was measured for different optimization levels and additionally evaluated by hand. Our early results show that a certain class of generic programming styles can be automatically optimized such that we obtain similar assembly codes as with low-level C or assembler programming; this indicates that not only C but even very high-level C++ techniques might become suitable for programming embedded systems in near future.

Difficulty : Handling of wide range of programming constructs in evaluating the impact of optimizations

C++ has such a rich set of programming constructs that research prototypes of analyses often only consider subsets of C/C++. Our goal was to create an infrastructure that permits performing research on real-world programs. In particular, the interface between PAG and ROSE required a careful design, such that we can maintain updates of ROSE and PAG, but keep the required changes in existing analysis specifications to a minimum. The evaluation tool WHOPA is based on this infrastructure and was adapted and extended throughout Y2 to permit investigations of the impact of optimizations and different programming styles and constructs in C++.

STMicroelectronics: Compiler for reconfigurable processors

During year two, more work was achieved on the retargetability of the compiler. It has been mainly devoted to the continuation of the effort to make the process fully dynamic. It includes:

- the completion of the definition of the scope of the possible extensions,
- the definition and implementation of the modification to be made in the compiler toolchain to make its reconfiguration fully dynamic,
- the completion of the work on the model to be built to allow the configuration of each toolchain component. The scope of the reconfiguration toolkit that was already in place to configure assembler and linker, was thus enlarged,
- the definition of the interface and tools needed to allow end users with different levels of expertise to defined their own extensions. The implementation of the end-user graphical interface has been outsourced to an external company (Coware/LISATek).

Longer term perspectives have been anticipated: for instance, we already know that SIMD instructions is a very demanded features in most recent applications and SOC. Some of the choices were made to ease automatic vectorization of SIMD extensions code in the future.

2.2.2 Publications Resulting from these Achievements

D. Atienza, S. Mamagkakis, F. Poletti, J. M. Mendias, F. Catthoor, L. Benini, D. Soudris, "*Efficient system-level prototyping of power-aware dynamic memory managers for embedded systems"*. Elsevier Integration journal 39(2): 113-130 (2006)

S. Mamagkakis, D. Atienza, C. Poucet, F. Catthoor, D. Soudris, "*Energy-efficient dynamic memory allocators at the middleware level of embedded systems"*, 6th Annual ACM Conference on Embedded Software (EMSOFT'06), Seoul, S. Korea, Oct. 2006

N. Genko, D.Atienza, J. Mendias, R. Hermida, G. De Micheli and F. Catthoor, "*A Complete Network-on-Chip Emulation Framework*," DATE, International Conference on Design and Test Europe, 2005, pp. 246-251.

M. Hohenauer, C. Schumacher, R. Leupers, G. Ascheid, H. Meyr, H. van Someren: Retargetable Code Optimization with SIMD Instructions, IEEE/ACM Int. Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS), Seoul (Korea), Oct 2006

[Fal06] H. Falk, J. Wager and A. Schaefer: Use of a Bit-true Data Flow Analysis for Processor-Specific Source Code Optimization. Submitted to ESTIMedia 2006.

[Sch06] Markus Schordan. The Language of the Visitor Design Pattern, 10th Brazilian Symposium on Programming Languages (SBLP'06) 2006. Proceedings of the 10th Brazilian Symposium on Programming Languages, pp. 235-248, ISBN 85-7669-071-3, ANAIS, 2006.

[QSVY06] Daniel Quinlan, Markus Schordan, Richard Vuduc, and Qing Yi. Annotating User-Defined Abstractions for Optimization. Workshop on Performance Optimization for High-Level Languages and Libraries (POHLL'06) in conjunction with 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2006), Rhodes Island, Greece, April 2006. Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2006), Workshop on Performance Optimization for High-Level Languages and Libraries, CDROM, IEEE Computer Society, 2006.

[SQ05] Markus Schordan and Daniel Quinlan. Specifying Transformation Sequences as Computation on Program Fragments with an Abstract Attribute Grammar. Fifth IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'05). Budapest, Hungary, September 2005. Proceedings of the Fifth IEEE International Workshop on Source Code Analysis and Manipulation, pp. 97-106, IEEE, ISBN 0-7695-2292-0, 2005.

2.2.3 Keynotes, Workshops, Tutorials

Hans van Someren delivered a lecture for students at Aachen on compiler technology – 21st June 2006.

A CoSy workshop was held in Amsterdam 3rd-5th May 2006 for academic partners including Dresden, Berlin, Bologna, University of Amsterdam, Delft and Leiden.

Rainer Leupers has given tutorials related to ARTIST2 activities at SBAC (Rio, Sep 2005), AICCSA (Dubai, Mar 2006), HiPEAC summer school (L´Aquila, Jul 2006), and MPSoC (Colorado, Aug 2006).

Peter Marwedel has organized a compiler workshop at Nokia (Düsseldorf, Jun 2006) with invited speakers including Heiko Falk, Andreas Krall, and Rainer Leupers from ARTIST2.

Tutorial: The ROSE Source-To-Source Translator
14th International Conference on Parallel Architectures and Compilation Techniques, (PACT'05), 2005
Saint Louis*, MI, U.S.A – September 2005*

Organization: Markus Schordan (TU-Vienna), together with Daniel J. Quinlan, Bronis R. de Supinski, Qing Yi, Richard Vuduc (Lawrence Livermore National Laboratory)

The tutorial was organized as a hands-on tutorial, separated in five sessions. Each session was organized such that participants got an overview of one component in ROSE, followed by a demo, and some hands-on exercises. Participants brought their own laptops and connected to our server for implementing the exercises. The tutorial covered the ROSE C++ intermediate representation, grammar based program analysis and data-flow analysis (PAG), pre-defined loop optimizations, visualization of the intermediate representation and attached analysis results.

3. Future Work and Evolution

3.1 Problem to be Tackled over the next 18 months (Sept 2006 – Feb 2008)

The activity will focus on further integration and refinement of architecture aware compilation technologies based on the compiler platform(s). It is foreseen to retain the existing "mini-cluster" structure outlined in the Compiler and Timing Analysis Cluster report and its collaboration scheme.

For the next 18 months during the collaboration between Dortmund Uni and IMEC vzw, it is planned to develop source-to-source optimization techniques mainly for statically allocated data, dealing with arrays inside loop bodies, but will also support dynamically allocated data like linked lists. The techniques will exploit certain properties of scratchpad or cache based memory hierarchies and will ensure data coherency in the memory subsystem through integration of timinig analysis and compilers.

ACE and Aachen's work on SIMD will continue to improve its ability to translate a wider class of loops into SIMD instructions. The SIMD and conditional execution work has to be integrated into CoSy solving various practical retargeting issues.

AbsInt – TU Vienna Cooperation: Perform whole-program source-code analysis of C/C++ applications and provide techniques and evaluation data for scalable analysis. The goal is to enable library centric development of embedded systems codes.

3.2 Current and Future Milestones

See also current 18 months workprogram. Please note that compiler platform activity milestones can hardly be separated from architecture aware compilation activity milestones, as both activities are interwoven. With increased use of the compiler platform at the different partners´ sites, it is expected that more and more activities will concentrate on building architecture aware compilation modules on top of that platform.

Year 1: Initial definition of common compiler platform

This milestone has been achieved by selection of ACE´s CoSy platform as the primary platform for most cluster partners.

Year 2: Initial implementation of the platform

This milestone has been achieved by installing and adopting the platform at the partners´ sites (partially after some setup meetings and training) for teaching and research purposes (e.g. for projects related to the architecture aware compilation activity). Examples: Aachen is using CoSy presently for development of SIMD and conditional instruction based code optimization in close cooperation with ACE. Likewise, Berlin (new affiliate partner) is using CoSy for research on new compiler verification technologies.

For **Year 3** and **Year 4**, the compiler cluster envisions a status where more and more new technologies (e.g. code optimization, verification) have been integrated into the common platform, which would lay a solid basis for self-sustained cooperations after the ARTIST2 funding period.

3.3 Indicators for Integration

It is expected that the integrated activities will lead to powerful prototypes of compiler techniques that will receive considerable interest for the industrial partners and their respective customers. It is therefore anticipated that further refinements and transfer of new techniques to industry will be carried out between partners also beyond the ARTIST funding period.

3.4 Main Funding

Main sources of funding are European Integrated Projects and STREPs, German Research Foundation (DFG), industrial sponsorship, and basic university funding