Network of Excellence

IST-004527 ARTIST2:
Embedded Systems Design

Activity Progress Report for Year 2

JPIA-Platform

Compilers Platform

Clusters:

Compilers and Timing Analysis

Activity Leader:

Prof. Dr. Sabine Glesner

Technical University of Berlin
http://pes.cs.tu-berlin.de/

The objective is to provide world-class code-synthesis and compiler tools for the generation of efficient machine code. Goals of the cluster include the integration of existing compiler-generation approaches allowing compilers for new architectures to be built quickly, efficiently and reliably.

One goal of the compilers sub-cluster is to achieve a tighter integration of European R&D activities by building on a carefully chosen industrial re-targetable compiler development platform that ensures interoperability.

The CoSy compiler platform provided by ACE is a state-of-the-art software system on which the common activities will build. This will reinforce Europe's leading position in the area of compilers for embedded processors.

Also, the partners aim at making existing advanced optimization algorithms available to designers of embedded systems. Finally, advanced high-level source-to-source transformations will be made available to users.

Table of Contents

1. Overview of the Activity

1.1 ARTIST2 Participants: Expertise and Roles

Prof. Dr. Reinhard Wilhelm – Saarland University (Germany)
Cluster Co-Leader, Compiler design, Static Program Analysis, Timing Analysis

Prof. Dr. Rainer Leupers – RWTH Aachen (Germany)
Cluster Co-Leader, Compilers.

Christian Bertin – ST Microelectronics (France)
Driver Applications

Prof. Dr. Peter Marwedel – Dortmund University (Germany)
Architecture-aware compilation, low-power code generation, Development of optimizations for WCET minimization.

Hans van Someren – ACE (The Netherlands)
CoSy *Lead Technical Architect. Core expertise used in ARTIST2: Software compilation techniques.*

1.2 Affiliated Participants: Expertise and Roles

Prof. Francky Catthoor – IMEC vzw. (Belgium)
Collaboration with Dortmund Uni. on high-level transformations for source code optimization.

Dr. Christian Ferdinand – AbsInt (Germany)
Program-Analysis Tools, Leading WCET estimation tool supplier.

Prof. Dr. Sabine Glesner – Technical University of Berlin (Germany)
Collaboration with ACE on development of verification methods for compilers

Dr. Markus Schordan, Prof. Andreas Krall – TU Vienna (Austria)
Collaboration with AbsInt on integration of PAG and development of program analyses
Areas of team's expertise: Development of tools for program analysis and optimization of high-level languages

1.3 Starting Date, and Expected Ending Date

September 1st, 2004 until December 2009.

1.4 Baseline

Traditionally, timing analysis tools have been designed independently of compilers. It has now turned out that proceeding along this path would result in a duplication of efforts. Flow facts are available in compilers and need to be regenerated in timing analysis tools. Timing information is available in timing analysis tools and would be useful for timing-aware optimizations in compilers. Currently, compilers use very rough approximations of timing, if they use any timing model at all. As a result, the impact of certain transformations on the run-time is frequently not known by the compilers. Hence, the user has to follow a trial-and-error approach, experimenting with different compiler options and figuring out a suitable combination of them. However, even this time-consuming process cannot really minimize the execution time since

options which might be good for some part of the code might lead to bad result for some other part of the code. A tight integration of timing models into compiler is urgently needed.

Moreover, there is a general trend in industry to replace non programmable hardware accelerators (NPAs) with flexible reconfigurable cores, which have specialized resources and instructions dedicated to a class of applications. These reconfigurable cores create new challenges for embedded development tools and especially for the compiler, and new challenge for processor architecture investigation tools.

Examples of configurable cores include Xtensa from Tensilica, ARC600 and 700 from ARC, CoreXtend from MIPS. Examples of flexible development tools are the Coware/LISATek processor and compiler designer based on CoSy Express, or the toolsets proposed by Tensilica and ARC for their core extension development.

Many applications in the embedded systems domain are both resource-restricted and safety-critical. This in turn requires compilers for embedded processors to be both efficient and correct. In a cooperation between the Technical University of Berlin and ACE, verification methods and tools for compilers are investigated.

1.5 Problem Tackled in Year2

We saw that timing models need to be integrated into compilers. This needs to be done such that a duplication of the efforts spent on timing analysis tools is avoided. Therefore, we were faced with the problem of making timing information from timing analysis tools available in compilers. We wanted to demonstrate the approach in a prototype compiler. The aiT timing analysis tool was selected since it had already demonstrated its applicability at an industrial level. The Infineon TriCore was selected for the demonstrator, due to the availability of timing models, a tight coupling between the aiT analyzer provided by AbsInt and Dortmund's compilers was established. A common data exchange format between timing analyzer and compiler was set up so that the compiler is able to pass information to the analyzer and vice versa. On top of this timing-aware compiler, new optimizations reducing worst-case timing will be developed.

Aachen and ACE have been working together on a number of topics together including an SIMD optimisation framework and conditional execution optimisation. ACE has also provided ongoing support of ST's team in relation to interprocedural optimisation.

Absint and Vienna have performed additional work on the ROSE/PAG integration in order to further explore the cluster´s secondary compiler platform.

TU Berlin and ACE have started initial discussions and CoSy platform trainings about Berlin´s future work on integration of compiler verification techniques into CoSy.

2. Summary of Activity Progress

2.1 Previous Work

2.1.1 Cooperation Absint – TU Vienna

Improving the productivity of developing embedded software requires that high-level abstractions can be used and reused. It is paramount for the next decade that these abstractions can be optimized such that they can be used in embedded systems without a significant performance impact. One of the major problems in optimizing high-level abstractions is aliasing, introduced by operations on references and pointers in programs. Our effort focuses on building and integrating existing infrastructures such that we can offer platforms for evaluating the impact of aliasing algorithms on today's languages performance that are used for embedded software.

Our goal for Year 1 was the integration of the Program Analysis Generator (PAG) of AbsInt in several platforms to share the same analysis in different infrastructures and leverage existing optimizations for evaluation. We created a tool, the PAG Interface Generator (PIG), to automate the PAG integration. The two different infrastructures which served as applications for the PAG integration by using PIG were ROSE and OCE/xDSPcore. ROSE is a source-to-source infrastructure that supports C++ (and Fortran in near future). The OCE/xDSPcore is the ATAIR open compiler infrastructure with a backend for digital signal processors.

The Input to PIG is a rule specification and PAG's syn-file. The PAG syn-file is processed to extract all information about the structure of the AST and the PIG rules are used to match with the extracted information. As output all required AST interface code is generated. Additionally we had to present the control flow information such that PAG can operate on the respective data of an infrastructure. This completed the integration of PAG.

Our goal for Y1 was the creation of a tool, PIG, to automate most aspects of a PAG integration and prove its usefulness by using it for integrating PAG in ROSE and OCE/xDSPcore. We have achieved both goals such that we can demonstrate the result by having a constant propagation analysis (as test) running in both environments.

2.1.2 Cooperation IMEC – University of Dortmund

The co-operation between IMEC and University of Dortmund resulted in alignment of the research objectives for the steering of locality-improving loop transformations at the source code level. For this purpose, the control flow complexity of a given source code should be evaluated for steering the loop-transformations and evaluating their benefits and overheads, before actually compiling the resulting code on the target platform. The requirements (the WHAT specifications) to tackle this problem were defined. Based on the WHAT specifications, Dortmund looked at high-level control flow cost estimation approaches 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.

2.2 Current Results

2.2.1 Technical Achievements / Outcomes / Difficulties encountered

2.2.1.1 Dortmund – Absint

Design of a WCET-aware C Compiler
Based on the interface language CRL2 of AbsInt's timing analysis tool aiT, a successful integration of timing analysis into the compiler infrastructure of Dortmund University was achieved. This was done by automatically translating the assembly-like contents used in compilers to aiTs CRL2 format. Additionally, the results produced by the WCET analyzer aiT were automatically collected and re-imported into the compiler infrastructure. This way, precise timing information is available within a compiler for future optimization for the very first time. In addition, a powerful mechanism was developed to attach not only WCET-related data to the compiler data structures, but also to store arbitrary information used by optimizations targeting different objectives than WCET. This approach will be useful in order to perform automated trade-offs between different optimization goals.

Source Code Transformation for WCET-Optimization
The influence of the loop nest splitting source code optimization on the worst-case execution time (WCET) was examined. Loop nest splitting minimizes the number of executed if-statements in loop nests of embedded applications. It identifies iterations of a loop nest where all if-statements are satisfied and splits the loop nest such that if-statements are not executed at all for large parts of the loop's iteration space. Especially loops and if-statements of high-level languages are an inherent source of unpredictability and loss of precision for WCET analysis. As a consequence, the optimization achieves a significantly more homogeneous control flow structure. Additionally, the precision of the optimization algorithms led to the generation of very accurate high-level flow facts. All together, considerable reductions of WCET were achieved by the source code optimization.

http://ls12-www.cs.uni-dortmund.de/research/C2C

2.2.1.2 ACE - Aachen

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.

A prototype comprising a set of optimisation engines and compiler has been constructed.

No-one has successfully been able to find a formalism or generate tools which facilitate generic retargeting of these algorithms.

2.2.1.3 Absint – TU Vienna

Extension of the ROSE-PAG integration from C to C++ and Implementation of Alias Analysis.
The ROSE-PAG integration achieved in Y1 for C was substantially extended to cover full C++ (only excluding Exceptions). This includes handling of templates, virtual methods, short-circuit evaluation in conditions, resolving overloaded functions, C++ name spaces, constructor and destructor calls. An intra-procedural shape analysis, published by our cluster partner Reinhard Wilhelm, was implemented using PAG. We extended the analysis to an inter-procedural shape analysis. The results of the analysis can be written to an external file and visualized using the tool AiSee.

Infrastructure for high-level specification of C++ program analyses

With the integration of PAG in ROSE, an infrastructure is available that permits using a high-level language for specifying an abstract interpretation of C++ programs. ROSE uses the EDG front end for parsing C++ and offers a powerful interface for accessing and transforming the abstract syntax tree (AST). The decorated AST offers the full type information of C++ input program and the PAG-ROSE integration permits using this type information in the PAG specification (e.g. for virtual method resolution).

Difficulty : Handling of the wide range of programming constructs of a general-purpose language

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 at a minimum. Our approach is grammar based and permits the generation of the used design patterns, glue code (between ROSE and PAG), and implementations of the required interfaces.

2.2.1.4 Dortmund – IMEC

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.

2.2.1.5 TU Berlin - ACE

TU Berlin works on the verification as well as on the development of optimizing compiler transformations and machine code generation. Especially in safety-critical applications in the embedded domain, compiler transformations must be both optimizing and correct. Hence, verification is necessary to ensure that transformations indeed preserve program semantics during compilation. Within ARTIST2, the focus is on the development of automated checkers that, for a particular compiler run with its source and target program, make sure that both programs are indeed semantically equivalent. As a starting point, the verification and development of checkers for loop transformations based on unimodular transformations is investigated.

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)
- Heiko Falk, Paul Lokuciejewski and Henrik Theiling. *Design of a WCET-aware C Compiler*. In *Proceedings of "The 6th International Workshop on Worst-Case Execution Time Analysis" (WCET)*, Dresden, Germany, July 2006.
- Heiko Falk and Martin Schwarzer. *Loop Nest Splitting for WCET-Optimization and Predictability Improvement*. In *Proceedings of "The 6th International Workshop on Worst-Case Execution Time Analysis" (WCET)*, Dresden, Germany, July 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.

- 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
- Daniel Quinlan, Markus Schordan, Richard Vuduc, and Qing Yi. *Annotating User-Defined Abstractions for Optimization*. In *Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2006), Workshop on Performance Optimization for High-Level Languages and Libraries (POHLL'06)*, CDROM, IEEE Computer Society, 2006.
- Markus Schordan. *The Language of the Visitor Design Pattern*, In *Proceedings of the 10th Brazilian Symposium on Programming Languages (SBLP'06)*, pp. 235-248, ISBN 85-7669-071-3, ANAIS, 2006.
- Markus Schordan and Daniel Quinlan. *Specifying Transformation Sequences as Computation on Program Fragments with an Abstract Attribute Grammar*. In *Proceedings of the Fifth IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'05)*, pp. 97-106, IEEE, ISBN 0-7695-2292-0, 2005.

2.2.3 Keynotes, Workshops, Tutorials

Keynote : Lecture on Compiler Technology
Aachen, Germany – *June 2006*

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

Workshop : CoSy workshop
Amsterdam, Netherlands – May 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.

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.
http://www.llnl.gov/casc/rose/

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 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.

3.1.1 University of Dortmund

The development of WCET-aware compiler optimizations will be performed by Dortmund University. On the one hand, this will include optimizations exclusively focussing on WCET as objective function, like e.g. exploitation of memory hierarchies for WCET minimization. On the other hand, the mechanisms provided by Dortmund’s WCET-aware compiler developed during ARTIST Year2 for multi-objective optimization (e.g. trading off WCET vs. code size) will be used. It is intended to set up a cooperation with ARTIST2 core partners of the timing analysis platform (Mälardalen, Vienna) in order to integrate flow analysis techniques into Dortmund’s compiler.

3.1.2 IMEC

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.

3.1.3 Absint

Recent developments in the field of register allocation allow for seperation of its subproblems thereby enabling better solution techniques. In this context Saarland University investigates coalescing, a compiler back-end optimization improving performance and reducing code size.

The solution technique relies on the strict separation of the subtasks and uses the general purpose optimization technique Integer Linear Programming (ILP) to derive (near-)optimal results. These are important, because in embedded systems the size of software is direclty related to production costs. To make this ILP-approach feasible, several optimizations were implemented improving the performance of the solving process to reasonable times. First experiments look promising and finial results in terms of speed up or code compaction of the compiled code will be available soon. This work is partially supported by the German Research Foundation (DFG) through the Graduiertenkolleg 623.

3.1.4 Technical University of Berlin

Berlin’s work, in cooperation with ACE, will focus on transformation of loops and their verification. It is planned to develop checkers for general unimodular loop transformations to ensure that, in individual compilations, the source and target programs are semantically equivalent.

3.1.5 ACE – Aachen

ACE will continue to support partners in using the CoSy compiler platform. Aachen will support and partially coordinate these activities and will provide partners with further support on the CoSy Express and LISATek technology. Furthermore, ACE and Aachen will continue their tight cooperation on platform based code optimization engines.

3.1.6 AbsInt – TU Vienna Cooperation

TU Vienna will further enhance the ROSE-PAG connection for performing whole-program source-code analysis of C/C++ applications and provide evaluation data on the scalability of an analysis. The goal is to provide enabling technology for library centric development of embedded systems codes. AbsInt and TU Vienna will continue their cooperation on automating the integration of PAG in existing platforms and in developing program analyses with PAG.

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 this activity will lead to a world-leading compiler platform prototype whose capabilities includes many existing and newly developed techniques which are so far largely separated due to heterogeneous compiler platforms in use by the different partners. High-level transformations will be available to the partners.

aiT and the compiler from Dortmund University have been tightly integrated. This combination of tools is used daily at Dortmund University. It provides the basis for numerous optimizations enabled by this integration. AbsInt is able to access the combined tools as well.

3.4 Main Funding

Main sources of funding are:

Large national project proposal to DFG, AVACS.

Several partners participate in a STREP proposal, PRESS.

ASTEC support by VINNOVA.

INRIA; CNRS, university funding.

Resources of the University of Dortmund

Resources of the Technical University of Berlin, also DFG funding