IST-004527 ARTIST2
Network of Excellence
on Embedded Systems Design

Activity Progress Report for Year 4

JPIA-Platform

Platform-based Code Optimization and Verification

Clusters:

Compilers and Timing Analysis

Activity Leader:

Prof. Dr. Sabine Glesner

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

Policy Objective (abstract)

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 many of the common activities have been built. This has reinforced Europe's leading position in the area of compilers for embedded processors.

Table of Contents

1. Overview of the Activity

1.1 ARTIST Participants and Roles

Prof. Dr. Sabine Glesner – Technical University of Berlin (Germany)
Activity Leader, Compiler Verification and Optimization.

Prof. Dr. Rainer Leupers – RWTH Aachen University (Germany)
Software for Systems on Silicon.

Prof. Dr. Peter Marwedel – Dortmund University (Germany)
Leader of cluster "Compilers and Timing Analysis", 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.*

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

1.2 Affiliated Participants and Roles

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

Dr. Stylianos Mamagkakis, Prof. Francky Catthoor – IMEC vzw. (Belgium)
Collaboration with TU Dortmund. on high-level transformations for source code optimization.

Dr. Markus Schordan, Prof. Andreas Krall – TU Vienna (Austria)
Collaboration with AbsInt on tool integration 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 September 2008.

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 run-time is frequently not known by 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 compilers and their optimizations is urgently needed.

There is a general trend in the 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 compilers, 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 cooperation between the Technical University of Berlin and ACE, verification methods and tools for compilers have been investigated.

1.5 Problem Tackled in Year 4

Based on the first approaches towards WCET-aware compiler optimization reported by TU Dortmund and AbsInt for Year 3 of Artist2, Year 4 focused on the development of more sophisticated WCET-aware optimizations. As an outcome of the current reporting period, it could be shown that combined procedure positioning and procedure cloning allows for improved WCET estimates; simultaneously, huge code size increases which were a an issue for the techniques reported during Year 3 of Artist2 are avoided now. Additionally, WCET-aware memory hierarchy exploitation using data and instruction scratchpads and using WCET-aware register allocation was studied. Furthermore, the infrastructure of the WCET-aware compiler developed at TU Dortmund was complemented by a fully automated static loop analyzer of superior quality. The work on timing-aware compiler analyses and optimizations was published on international conferences and workshops of very high quality.

ACE and TU Berlin continued their cooperation for further improving the Itanium compiler platform. At TU Berlin, we developed more compiler optimization techniques to overcome the impact of the memory wall. To this end, we consider novel speculative optimization techniques of memory accesses to reduce their effective latency. In one approach, we developed a speculative optimization which reduces the number of memory accesses caused by the use of global variables, and we could show that notable performance improvements can be obtained for the SPEC CPU2006 benchmarks. Besides, we also developed a speculative optimization which targets memory accesses in general. Using statistical machine learning techniques, we trained predictors to learn the memory dependency degree of a given pair of accesses. The result is used in our optimization to decide about the profitability of a given optimization step. Preliminary results show that the predictors can effectively and precisely predict the memory dependencies for previously unseen pairs of memory instruction.

Besides, we also considered how the correctness of compilers can be improved via formal verification. One crucial phase in the compiler is the code generation. Due to its complexity, it is highly error prone. We formalized important parts of the semantics of the intermediate representation within the compiler and of the Itanium assembler code, respectively. From that, we could prove code generation as correct for a subset of the language.

Aachen and ACE continued their cooperation. ACE productized the conditional execution engines and further development on the analysis framework around the SIMD engines has been developed by a student from Aachen.

AbsInt and Dortmund continued their work on a WCET-aware compiler. This compiler uses the AIR interface format and aiT technology to determine WCET estimates for various code possibilities and then to select the most efficient of these.

TU Vienna and AbsInt further integrated SATIrE and PAG, for performing whole-program source-code analysis of C/C++ applications and evaluated the scalability of WCET analysis methods in the presence of optimizations with the Mälerdalen Benchmark suite. In year four, SATIrE was improved and made operational. In cooperation with Saarland University, it was successfully applied to the analysis of loop bounds and function pointers.

TU Vienna also participated in the WCET Tool Challenge with TuBound, a tool built with SATIrE and PAG in 2008. This also motivated further research in annotation languages and new concepts for source code annotation-based WCET analysis. The iterative refinement technique of annotations, and the transformation of annotations according to performed compiler optimizations, was the most crucial conceptual achievement.

IMEC has continued research related with data memory hierarchy assignment optimizations. More specifically, we developed a fully automatic application analysis and transformation tool which selects static data-structures for transfer to the Scratchpad Memory (SPM) and schedules data transfers between background memory and SPM (pre-fetching) to achieve both high performance and low power consumption. Moreover, profiling and analysis tools were developed to leverage on information about the dynamic data access behavior of simultaneously executing threads in order to enable the relevant memory data transfer optimizations.

1.6 Comments From Year 3 Review

1.6.1 Reviewers' Comments

This is a quality document. No specific remarks.

1.6.2 How These Have Been Addressed

We are happy to hear the positive comment and hope that the same holds for this document.

2. Summary of Activity Progress

2.1 Previous Work in Year 1

2.1.1 Cooperation AbsInt – TU Vienna

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.

Achievements for Year 1: Our goal for Y1 was the creation of a tool, PIG, to automate most aspects of the 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 cooperation between IMEC and University of Dortmund resulted in the 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 Previous Work in Year 2

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

Optimization of Conditional Execution in CoSy

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

A prototype comprising a set of optimization engines and compilers 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 TU Berlin – ACE

TU Berlin, who joined the ARTIST2 NoE as an affiliated partner in Y2 and became a core partner in Y3, has worked 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 has been investigated.

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 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.

2.3 Previous Work in Year 3

2.3.1 Technical Achievements

SIMD & Conditional Execution Support in CoSy (Aachen, ACE)

- **SIMD retargeting**: A retargeting formalism for the SIMD optimization developed earlier has been devised. The SIMD instructions can now be described within CoSy's natural code generator description format. To achieve this, adaptations to the backend generator were necessary and improvements in data dependency analysis were devised.
- **SIMD enabling loop transformations**: Several loop transformations, such as strip mining and loop peeling, that increase the number of possible SIMD operations have been implemented in the CoSy system. They interact with the SIMD optimizer and consult the code generator description in order to assess the benefit of each transformation.
- **Conditional execution retargeting**: The retargetability of the conditional execution optimization previously done has been shown by adding support for a commercial, CoSy based, compiler. The results of this work by a student from Aachen led to several improvements in the cost calculation and to beneficiary transformations.

Transformation of Flow Facts within Optimizations of a WCET-aware Compiler (Dortmund University)

Timing analysis relies on the presence of highly precise flow facts. Flow facts represent information about the possible flow of control through a program under analysis – e.g. iteration counts of loops. Usually, such information is provided by the designer who is fully responsible for its correctness. In the context of the WCET-aware compiler developed at Dortmund in the

past years, flow facts can now be entered into the compiler within the source code to be processed by the compiler. These flow facts are analyzed and kept semantically correct during each transformation performed by the compiler. In particular, all flow facts are maintained and adjusted during all compiler optimizations, even if they heavily restructure the code. These automatically transformed flow facts are finally passed to the WCET analyzer aiT provided by AbsInt, in order to perform the actual WCET analysis. This achievement has taken away the burden from the designer to specify flow facts at the assembly code level. Instead, the designer now can annotate the source code, invoke the compiler, perform optimizations, and still obtains valid WCET results.

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

Compile-Time Decided Instruction Cache Locking Using Worst-Case Execution Paths (Dortmund University, AbsInt)

Caches are notorious for their unpredictability. It is difficult or even impossible to predict if a memory access results in a definite cache hit or miss. This unpredictability is highly undesired for real-time systems. The Worst-Case Execution Time (WCET) of software running on an embedded processor is one of the most important metrics during real-time system design. The WCET depends to a large extent on the total amount of time spent for memory accesses. In the presence of caches, WCET analysis must always assume a memory access to be a cache miss if it cannot be guaranteed that it is a hit. Hence, WCETs for cached systems are imprecise due to the overestimation caused by the caches.

Modern caches can be controlled by software. The software can load parts of its code or of its data into the cache and lock the cache afterwards. Cache locking prevents the cache's contents from being flushed by deactivating the replacement. A locked cache is highly predictable and leads to very precise WCET estimates because the uncertainty caused by the replacement strategy is eliminated completely.

In year 3 of Artist2, the lockdown of instruction caches at compile-time to minimize WCETs was explored. In contrast to the current state of the art in the area of cache locking, our techniques explicitly take the worst-case execution path into account during each step of the optimization procedure. This way, we can make sure that always those parts of the code are locked in the I-cache that lead to the highest WCET reduction. The results demonstrate that WCET reductions from 54% up to 73% can be achieved with an acceptable amount of CPU seconds required for the optimization and WCET analyses themselves.

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

Influence of Procedure Cloning on WCET Prediction (Dortmund University, AbsInt)

For the worst-case execution time analysis, especially loops are an inherent source of unpredictability and loss of precision. This is caused by the difficulty to obtain safe and tight information on the number of iterations executed by a loop in the worst case. In particular, data-dependent loops whose iteration counts depend on function parameters are extremely difficult to analyze precisely. Procedure Cloning helps by making such data-dependent loops explicit within the source code, thus making them accessible for high-precision WCET analyses.

In year 3 of Artist2, we studied the influence of standard optimizations found in ordinary compilers on the program's WCET. We present the effect of Procedure Cloning applied at the source-code level on worst-case execution time. The optimization generates specialized versions of functions being called with constant values as arguments. In standard literature, it

is used to enable further optimizations like constant propagation within functions and to reduce calling overhead.

Our work shows that Procedure Cloning for WCET minimization leads to significant improvements. Reductions of the WCET from 12% up to 95% were measured for real-life benchmarks. These results demonstrate that Procedure Cloning improves analyzability and predictability of real-time applications dramatically. In contrast, average-case performance as the criterion Procedure Cloning was developed for is reduced by only 3% at most. Our results also show that these WCET reductions only implied small overhead during WCET analysis.

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

Optimization and Verification in Compilers (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

TU Berlin has established a platform for a research compiler, using the compiler tool CoSy provided by ACE. We developed a backend specification for the Intel Itanium processor, which gave us an industrial-strength compiler for this architecture (the current SPEC benchmark suite CPU2006 is compiled). The challenges in establishing this platform have been to exploit the special features of the Itanium, e.g. predication. On top of that, first optimizations have been developed. These optimizations mainly aim at reducing the impact of the memory wall on program performance, which is drastic on modern VLIW processors like the Itanium.

Furthermore, we investigate how verification techniques can be brought into practice, in order to ensure that the code generated by the compiler is correct. We considered the scheduling phase of a compiler, which rearranges the instructions of a program in order to improve runtime performance. We formalized the scheduling phase in the theorem prover Isabelle/HOL and derived a criterion that is necessary for correctness of the scheduled code. From this criterion, we automatically generated the core of a checker (facilitated by the code generation capabilities of Isabelle) which can be used to augment any existing scheduler. By this, we discovered a bug in the scheduler of the popular GNU assembler, which lead to possibly incorrect programs.

http://www.pes.cs.tu-berlin.de/

Optimized Dynamic Memory Allocation (IMEC vzw.)

The main technical achievement of IMEC (in the context of the collaboration with Dortmund Uni.) is the realization of multiple fine-grain dynamic memory allocation design options in software modules of the standard compiler library (e.g., *libc*), which can be parameterized and combined in many ways. Then, we extract application specific information (i.e., Software Metadata) and memory hierarchy specific information (i.e., Hardware Metadata) from the embedded system design. All this metadata information is exploited by our tools, which parameterize and combine the aforementioned dynamic memory allocation modules using energy efficiency criteria. The final result is the construction of a unique dynamic memory allocator, which is compiled with the software application and utilizes a unique combination design options. This unique design fine tunes its energy consumption management according

to the unique characteristics of the software application and the memory hierarchy of the embedded system.

Automatic Source-Code Annotation (TU Vienna, AbsInt)

Source-Code annotations allow for providing additional semantic information. Our goal was to support programmers in providing the information that the analyzer can determine automatically in a readable form, such that the need for user-defined annotations is minimized. We have created a general mechanism that allows for annotating source codes automatically with a textual annotation representing the results of arbitrary analysis results. The annotation mechanism is implemented as a source-to-source transformation at the statement level, generating analysis information either as comments or pragmas at the corresponding statement positions in the C/C++ source-code. A first application of this mechanism allows us to generate may-alias and must-alias annotations based on the results of a shape-analysis.

External Program Representation for Tool Interoperability (TU Vienna, AbsInt)

An external program representation permits to build tool chains for program analysis and transformation. We have designed and implemented an external representation for C programs. The connection has been established in both directions, for generating an external C representation as well as reading in the external representation. For the external format we have chosen Prolog syntax, because it is suitable for querying programs and specifying transformations at a high-level. This new feature has already been used in cooperation with the CoSTA project for specifying the transformation of WCET annotations according to loop optimizations performed with our LLNL-ROSE loop optimizer at the C code level.

The tools LLNL-ROSE, the ROSE loop optimizer, the Program Analysis Generator (PAG), and the generator and parser for the external program representation have been integrated within the Static Analysis Tool Integration Engine (SATIrE), which aims at offering the combined features of all tools to the user through a single interface and a set of new specialized tools. The full C++ language has been addressed, in particular virtual methods, templates, constructor/destructor calls, function pointers, etc. – only exceptions are not addressed yet. ROSE permits generating C++ code and lowered C code. The generated code can serve as input to ACE's compiler for generating optimized machine code.

http://www.complang.tuwien.ac.at/markus/satire/

2.3.2 Individual Publications Resulting from these Achievements

Aachen

- S. Kraemer, R. Leupers, G. Ascheid, H. Meyr: SoftSIMD: *Exploiting Subword Parallelism Using Source Code Transformations*, Design Automation & Test in Europe (DATE), Nice (France), Apr 2007
- H. Scharwaechter, R. Leupers, H. Meyr, J. Youn, Y. Paek: *A Code-Generator Generator for Multi-Output Instructions*, IEEE/ACM Int. Conf. on Hardware/Software Codesign and System Synthesis (CODES + ISSS), Salzburg (Austria), Oct 2007

TU Dortmund

- Paul Lokuciejewski, Heiko Falk, Martin Schwarzer and Peter Marwedel. *Tighter WCET Estimates by Procedure Cloning*. In *Proceedings of "The 7th International Workshop on Worst-Case Execution Time Analysis" (WCET)*, Pisa, Italy, July 2007.
- Heiko Falk and Peter Marwedel (Editors). *Proceedings of the 10th International Workshop on Software & Compilers for Embedded Systems (SCOPES)*, Nice, France, April 2007.

TU Berlin

- L. Gesellensetter, S. Glesner, E. Salecker: *Formal Verification with Isabelle/HOL in Practice: Finding a Bug in the GCC Scheduler*, 12th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2007), 2007.

IMEC vzw.

- S. Mamagkakis, D. Soudris, F. Catthoor: *Middleware design optimization of wireless protocols based on the exploitation of dynamic input patterns*. DATE 2007: 1036-1041
- M. Peon-Quiros, A. Bartzas, S. Mamagkakis, F. Catthoor, J. M. Mendias, D. Soudris: *Direct Memory Access Optimization in Wireless Terminals for Reduced Memory Latency and Energy Consumption*. PATMOS 2007: 373-383

TU Vienna

- Raimund Kirner, Jens Knoop, Adrian Prantl, Markus Schordan, Ingomar Wenzel: *WCET Analysis: The Annotation Language Challenge*. 7th Workshop on WCET Analysis, Pisa, Italy, July 3, 2007.
- Markus Schordan: *The Language of the Visitor Design Pattern*. Journal of Universal Computer Science (JUCS), Vol. 12, No. 7, pp. 849-867, August 2006.
- Markus Schordan: *Integrating Tools and Languages for Source-Based Static Analysis and Optimization of High-Level Abstractions*. Post-Workshop Proceedings of the GI-Fachgruppe Programmiersprachen und Rechenkonzepte, 2007.

2.3.3 Interaction and Building Excellence between Partners

Compiler teams from Aachen and ACE held numerous face-to-face discussions and design reviews relating to the joint R&D at ACE's offices in Amsterdam and Aachen. ACE gave a lecture on compiler technology on 17 January 2007 at Aachen and two Aachen students spent several months at ACE on SIMD, Conditional Execution and Loop Transformations.

Researchers from Berlin attended the CoSy Community Gathering and academic workshops that were held at ACE in October 2006, March 2007, and August 2007. Further interaction between ACE and TU Berlin has been via email with regular support questions and mentoring in the use of the CoSy platform as the subject.

A researcher from Berlin attended the ARTIST2 MOTIVES winter school in February 2007 held at Trento, Italy. Researchers from Berlin visited RWTH Aachen in February 2007 and the University of Edinburgh in August 2007 for academic exchange.

In multiple bilateral meetings during the last year, Dortmund and IMEC defined a common roadmap on research for MPSoC memory management.

TU Vienna und AbsInt have intensified their cooperation in integrating AbsInt's program analysis generator PAG with the C++ infrastructure ROSE as part of the compiler platform. In Y3 an external program representation and an automatic analysis results annotation mechanism have been added and integrated to create the Static Analysis Tool Integration Engine, SATIrE. The cooperation with AbsInt allowed for performing tests with industrial codes and working towards a test bench for evaluating the suitability of SATIrE for real-world applications.

2.3.4 Joint Publications Resulting from these Achievements

Aachen/ACE

- 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

TU Dortmund University/AbsInt

- Heiko Falk, Sascha Plazar and Henrik Theiling. *Compile-Time Decided Instruction Cache Locking Using Worst-Case Execution Paths*. In *Proceedings of "The International Conference on Hardware/Software Codesign and System Synthesis" (CODES+ISSS)*, Salzburg, Austria, October 2007.
- Paul Lokuciejewski, Heiko Falk, Martin Schwarzer, Peter Marwedel and Henrik Theiling. *Influence of Procedure Cloning on WCET Prediction*. In *Proceedings of "The International Conference on Hardware/Software Codesign and System Synthesis" (CODES+ISSS)*, Salzburg, Austria, October 2007.

2.3.5 Keynotes, Workshops, Tutorials

Workshop : ACE Second CoSy Community Gathering (CCG'06)
Amsterdam, Netherlands – October 2006

This CoSy workshop was held to give the users of the CoSy system a platform to present their results and discuss their experiences. Amongst others, participants came from RWTH Aachen and Technical University of Berlin.

Workshop : CoSy Research Workshop
Amsterdam, Netherlands – March 2007

A CoSy workshop was held for academic partners including Universities of Amsterdam, Cambridge, Aachen, Edinburgh, Twente, Dresden, Berlin.

Workshop : CoSy Research Workshop
Amsterdam, Netherlands – August/September 2007

A CoSy workshop was held for academic partners including Universities of Edinburgh, Delft, Berlin, Amsterdam, and Imperial College London, IMEC, INESC-ID, NTHU.

Workshop : Software & Compilers for Embedded Systems (SCOPES) 2007
Nice, France – April 20, 2007

The influence of embedded systems is constantly growing. Increasingly powerful and versatile devices are developed and put on the market at a fast pace. The number of features is increasing, and so are the constraints on the systems concerning size, performance, energy dissipation and timing predictability. Since most systems today use a processor to execute an application program rather than using dedicated hardware, the requirements can not be fulfilled by hardware architects alone: Hardware and software have to work together to meet the tight constraints put on modern devices.

One of the key characteristics of embedded software is that it heavily depends on the underlying hardware. The reason of the dependency is that embedded software needs to be designed in an application specific way. To reduce the system design cost, e.g. code size, energy consumption etc., embedded software needs to be optimized exploiting the characteristics of the underlying hardware.

SCOPES focuses on the software generation process for modern embedded systems. Topics of interest include all aspects of the compilation process, starting with suitable modeling and specification techniques and programming languages for embedded systems. The emphasis of the workshop lies on code generation techniques for embedded processors. The exploitation of specialized instruction set characteristics is as important as the development of new optimizations for embedded application domains. Cost criteria for the entire code generation and optimization process include runtime, timing predictability, energy dissipation, code size and others. Since today's embedded devices frequently consist of a multi-processor system-on-chip, the scope of this workshop is not limited to single-processor systems but particularly covers compilation techniques for MPSoC architectures.

In addition, this workshop intends to put a spotlight on the interactions between compilers and other components in the embedded system design process. This includes compiler support for e.g. architecture exploration during HW/SW codesign or interactions between operating systems and compilation techniques. Finally, techniques for compiler aided profiling, measurement, debugging and validation of embedded software are also covered by this workshop, because stability of embedded software is mandatory.

SCOPES 2007 is the 10th workshop in a series of workshops initially called "International Workshop on Code Generation for Embedded Processors". The name SCOPES has been used since the 4th workshop. The scope of the workshop remains software for embedded systems with emphasis on code generation (compilers) for embedded processors.

SCOPES 2007 was organized by Heiko Falk and Peter Marwedel from Dortmund University and was held as DATE Friday Workshop.

http://www.scopesconf.org/scopes-07/

Workshop : Compiler Optimization Meets Compiler Verification (COCV'07)
Braga, Portugal – 25 March 2007

COCV provides a forum for researchers and practitioners working on optimizing and verifying compilation, and on related fields such as translation validation, certifying compilation and embedded systems with a special emphasis on hardware verification, formal synthesis methods, correctness aspects in HW/SW co-design, formal verification of hardware/software systems, and practical and industrial applications of formal techniques for exchanging their latest findings, and for plumbing the mutual impact of these fields on each other. By encouraging discussions and co-operations across different, yet related fields, the workshop strives for bridging the gap between the communities, and for stimulating synergies and cross-fertilizations among them.

COCV'07 is the 6th workshop in a series of workshops held annually since 2002. COCV'07 was organized by Sabine Glesner (TU Berlin), Jens Knoop (TU Vienna) and Rolf Drechsler (University of Bremen) and was held as a satellite event of ETAPS'07.

http://pes.cs.tu-berlin.de/cocv2007/

Exhibition : OpenCoSy Stand
Design, Automation and Test in Europe (DATE)
Nice, France – 16-20 April, 2007

Aachen presented the results of its research at DATE in Nice. A specially organized OpenCoSy stand for academic users of CoSy – www.opencosy.org/announcements. This stand proved very attractive to attendees over the course of the week with the results obtaining an unusually high level of visibility for such projects. Also represented on the stand were University of Amsterdam, TU Delft, Leiden University and Edinburgh University.

Workshop : Dagstuhl Seminar 08161 "Scalable Program Analysis"
Schloss Dagstuhl, Germany – 13.04.08 – 18.04.08.

Organizers: Florian Martin (AbsInt), Hanne Riis Nielson (Technical University of Denmark), Claudio Riva (NOKIA Research Center - Helsinki), Markus Schordan (TU Vienna).

The application for the seminar has been accepted in 2007.
http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=2008161/

2.4 Final Results

2.4.1 Technical Achievements

Design of a Static Loop Analyzer (TU Dortmund)

Knowledge about the loop iteration counts is mandatory for a large number of different program analyses (e.g. loop transformations like loop unrolling or loop tiling). Furthermore, it is impossible to derive WCET information statically without knowing iteration counts of loops. In the past, this information was collected and annotated manually which was a very time consuming and error-prone job. By integrating a loop analyzer into the existing WCET-aware compiler framework developed at TU Dortmund, it is now possible to produce WCET-optimized code fully automatically.

Static loop bound analysis has the same complexity as the halting problem. Because of this, it is impossible to get exact results in an acceptable amount of time for general problems. The only way to do loop analysis is to reduce complexity by applying approximations. This is the reason why our loop analyzer is based on an enhanced version of the classical paradigm of Abstract Interpretation which is a mathematical framework to obtain sound approximations of the original problem.

The main disadvantage of conventional Abstract Interpretation is that it is still a very time consuming analysis technique if the results should not be too pessimistic. It analyses loops by evaluating every iteration on its own. By integrating a new static polytope-based loop evaluation inside the original Abstract Interpretation, it was possible to reduce the required analysis time without getting inferior results. If a loop is reached during fixed-point iteration of

Abstract Interpretation, it is checked whether the new polytope-based loop analysis is applicable. If so, the loop body is analyzed only once and the iterative calculation of Abstract Interpretation can be skipped. If polyhedral analysis is not applicable, the loop is analyzed conventionally.

To analyze loops with the new polytope-based technique, several constraints concerning the loop's header and body have to be met. These constraints require that the body of the loop is not too complex. This is achieved by integrating another technique called Program Slicing. Program Slicing determines all expressions that are superfluous for the current analysis so that their evaluation can be skipped.

To demonstrate the applicability of the developed loop analyzer, we tested it using 96 benchmarks containing more than 700 loops. These benchmarks were taken from the benchmark suites MRTC, DSPStone, MiBench, UTDSP and MediaBench. Related to these benchmarks, 99.15% of all loops could be analyzed successfully. Furthermore, 95.62% of all loops could be analyzed without any overapproximation thus leading to exact results. Concerning the running time of our analysis, we could accelerate the time consumed by conventional Abstract Interpretation up to a factor of 100 for some benchmarks it the combination of Program Slicing and polyhedral loop analysis was enabled.

Another fact showing that our loop analyzer is also applicable to real-world problems is that it was the only tool which was able to answer all questions related to flow fact during the WCET tool challenge 2008, organized by the Timing Analysis cluster of Artist2.

http://ls12-www.cs.tu-dortmund.de

WCET-aware Procedure Positioning and Cloning (TU Dortmund, AbsInt)

In year 3 of Artist2, we studied the influence of the standard compiler optimization Procedure Cloning on the worst-case execution time and could show that WCET reductions of up to 95% could be achieved. The main drawback of this optimization is its heavy code size increase. In year 4 we continued our work on Procedure Cloning and extended the optimization by WCET concepts. In a first extension, the novel WCET-driven Cloning focuses on the optimization of those functions that promise the highest WCET improvement. We achieve WCET reductions of 64.2%, while restricting the average code size increase to 22.6% which is a significant improvement compared to the average code size increase of more than 80% in the previous work. In a further extension, we combined our WCET-driven Procedure Cloning with a smart positioning of the newly created function clones. By placing the clones and their callers contiguously in memory, cache conflict misses can be reduced. Results show that the WCET can be reduced by 7% on average when Cloning is combined with a sophisticated positioning of the cloned functions.

Moreover, the compiler optimization Procedure Positioning was exploited for a WCET reduction. Procedure Positioning is a well known optimization aiming at the improvement of the instruction cache behavior. In the past, this technique was based on profiling information to improve the average-case performance of the program. We exploited the ideas behind Procedure Positioning for an effective WCET minimization. Our optimization operates on a call graph which is based on WCET information to find the function candidates for a contiguous memory allocation that promise the highest WCET reduction. Thus, our WCET-centric call graph is more reliable than previous approaches since it is valid for all program and all input data. We developed two types of the WCET-aware Positioning, an effective greedy and a fast heuristic approach. Results on real-world benchmarks show that WCET reductions of 10% on average could be achieved while the average-case execution time (ACET) was decreased by 2% on average. This emphasizes the importance of compiler optimizations tailored towards an

effective WCET reduction which can not be accomplished with standard ACET optimizations that take decisions based on other cost models.

http://ls12-www.cs.tu-dortmund.de

WCET- and Memory Architecture-aware Compilation (TU Dortmund, AbsInt)

In year 4 of Artist2, we studied the influence of scratchpad memory allocation techniques on worst-case execution times. Here, we developed integer linear programming models in order to decide which parts of a program's code or data should be moved onto the highly predictable scratchpad memory. In contrast to the current state of the art in this area, we developed integrated models explicitly taking the critical path of a program defining its WCET into account. The optimizations developed during year 4 of Artist2 pre-compute the scratchpad contents during compilation time. During runtime of the optimized programs, the scratchpad contents remain unchanged. The results demonstrate that average WCET reductions of 11.6% can be achieved by simply moving parts of the global data of 30 representative benchmarks onto the scratchpad. First experiments with scratchpad memory allocation of program code show that WCET reductions of more than 50% can be achieved for several benchmarks. Unlike previously published approaches, our ILP formulations scale well so that our techniques only require a few CPU seconds to solve the optimization problem.

In addition to scratchpad memory allocation, we investigated WCET-aware register allocation. Within the compiler community, register allocation is considered the most important optimization since it leads to an optimized use of processor registers which are by far the most efficient memories of an entire system. Traditionally, register allocation relies on graph coloring approaches which apply some simple heuristics to decide where so-called spill code (i.e. load/store instructions swapping registers in and out to main memory) has to be inserted. Since these traditional techniques are timing-unaware, they may lead to spill code generation along a program's critical path defining the WCET. By making a graph coloring register allocator WCET-aware, large average WCET reductions of 27.3% were measured for a total of 28 representative benchmarks. It is worthwhile mentioning that still huge gains can be achieved even in such well-established areas like register allocation.

http://ls12-www.cs.tu-dortmund.de

Optimization and Verification in Compilers (TU Berlin, ACE)

TU Berlin works on the verification as well as on the development of optimizing compiler transformations and machine code generation. 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

TU Berlin has established a platform for a research compiler, using the compiler tool CoSy provided by ACE. We developed a backend specification for the Intel Itanium processor, which gave us an industrial-strength compiler for this architecture (the current SPEC benchmark suite CPU2006 can be compiled). We investigated the development of novel compiler optimizations to mitigate the impact of the memory wall on program performance, which is drastic on modern VLIW processors like the Itanium. One result lies in the development of speculative compiler optimization (Speculative register promotion for global variables), which reduces the number of loads induced by the use of global variables and thereby leads to a better memory performance. We could show that this optimization leads to notable performance improvement for the SPEC CPU2006 benchmark suite. Second, we also considered how memory accesses in general can be optimized. To this end, we developed a speculative optimization, which targets all kinds of memory accesses. It requires precise information about the dependencies

amongst memory accesses, to decide whether or not a given optimization is beneficial. However, state-of-the-art alias analyses are too imprecise for that aim. As a consequence, we propose to use statistical machine learning techniques to yield predictors, which can act as heuristics to determine the dependency amongst memory accesses. In our experiments, we could show that with the trained predictors, the dependencies could be efficiently and precisely predicted for unseen programs.

Furthermore, we investigated how the code generator phase, which is a crucial compiler optimization, can be verified formally. We considered code generators that are based on bottom-up rewriting, which is the most common technique in current compilers. The machine code is generated from the intermediate representation by applying a set of rules that specify how a construct on the abstract level can be mapped to the machine level. The correctness of these rules is necessary for the code generator to be correct. We developed a formal model for a subset of the compiler intermediate representation from the CoSy-Compiler and for a subset of the Itanium assembler. On top of this, we specified code generator rules and proved them correct. Besides the correctness proofs, we developed proof engineering techniques that help to handle the size of formalizations for complete rule sets. We investigated which strategies known from software engineering can be applied in the area of formal verification, too. In addition, we investigated methods and have preliminary results concerning proof automation. We could show correctness for a subset of the rules used in the specification for the Itanium. Besides, our results make the verification of complete code generator specifications more realistic.

http://www.pes.cs.tu-berlin.de/

Source-To-Source WCET Analysis (TU Vienna, AbsInt)

TU Vienna and AbsInt work on scalable source-level analysis and annotation-based timing analysis methods. The presentation of analysis results and the iterative enhancement by the user with expert knowledge about the timing behavior of a given system is incorporated by allowing round-trip engineering of timing analyses. The infrastructure SATIrE allows building analyzers that take source-code annotations as additional input as well as automatically generate output as annotations. The iterative application of this approach increases productivity, by requiring the user only to annotate the timing relevant information that is not automatically computed. The integration of PAG was fundamental in investigating scalability and precision of analyses. The automatic annotation of programs with PAG analysis results is possible because of its combination with the LLNL-ROSE C/C++ backend in SATIrE. The importance of user-readable analysis results as annotations has also fostered the use of PAG and SATIrE in teaching program analysis at several universities in Y3 and Y4.

For timing analysis various supporting analyses are necessary. In Y4 SATIrE was enhanced with a Steensgaard-style points-to analysis. This analysis partitions a program's objects (variables and dynamically allocated memory regions) into equivalence classes, and models which classes may contain pointers to which other classes. Members of structures are treated as individual objects unless accesses through pointers of incompatible type make it necessary to collapse structure fields. The analysis runs in almost linear time in the size of the program, allowing it to scale to very large input programs.

Further more, TuBound, was created with SATIrE. It performs an inter-procedural context-sensitive interval analysis with PAG and computes loop bounds for specific loop patterns in Prolog. For loops where a loop bound cannot be established, annotations can be provided by the user. The implemented algorithm for loop bounds was evaluated with the Mälerdalen Benchmark suite. TuBound also participated in this year's WCET Tool Challenge 2008.

http://www.complang.tuwien.ac.at/markus/satire/

Retargetable Code Optimizations (RWTH Aachen, ACE)

The cooperation between ACE and Aachen on retargetable code optimizations has been continued. The major problems left were complete integration of the conditional execution prototype into CoSy and further development on the analysis framework necessary to facilitate compilation for SIMD architectures.

The conditional execution engines have been extended by a strong retargeting formalism. Several compiler passes and a few extensions to the backend description used in CoSy have been devised to achieve this goal. This work has by now been productized in the CoSy release.

Work continues in two directions. Improved loop analysis infrastructure is developed by at ACE by a student from Aachen. The goal of this project is to deliver the information necessary for advanced optimizations like vectorization. Also a cooperation to make efficient code generation for clustered VLIW processors available is being planned.

ARTIST Interchange Representation and Attribute Database: AIR (AbsInt)

Work on the AIR format continued. The format was extended and adapted to the needs of the partners. The attribute database was extended by new attributes as required.

http://www.theiling.de/absint/attrdb.fcgi

Analyses on C source code (TU Vienna, AbsInt, Usaar)

AbsInt and TU Wien continued working on the Static Analysis Tool Integration Engine, SATIrE, which connects LLNL-Rose with AbsInt's Program Analysis Generator (PAG). In year four, SATIrE was improved and made operational. In cooperation with Saarland University, it was successfully applied to the analysis of loop bounds and function pointers.

http://www.complang.tuwien.ac.at/markus/satire/

2.4.2 Individual Publications Resulting from these Achievements

TU Dortmund

- Paul Lokuciejewski, Heiko Falk, Peter Marwedel: *WCET-driven Cache-based Procedure Positioning Optimizations*, Proceedings of the 20th Euromicro Conference on Real-Time Systems (ECRTS), Prague, Czech Republic, July, 2008.
- Heiko Falk (Editor), Proceedings of the 11th International Workshop on Software & Compilers for Embedded Systems (SCOPES), Munich, Germany, March 2008.
- Paul Lokuciejewski, Fatih Gedikli, Peter Marwedel: *Accelerating WCET-driven optimizations by the Cold Path Paradigm*, Submitted to Design Automation & Test in Europe (DATE), 2009.
- Paul Lokuciejewski, Daniel Cordes, Heiko Falk, Peter Marwedel: *A Fast and Precise Static Loop Analysis based on Abstract Interpretation, Program Slicing and Polytope Models*, Submitted to the International Symposium on Code Generation and Optimization (CGO), 2009.

- Sascha Plazar, Paul Lokuciejewski, Peter Marwedel: *A Retargetable Framework for Multi-objective WCET-aware High-level Compiler Optimizations*, Submitted to the IEEE Real-Time Systems Symposium (RTSS), 2009.
- Peter Marwedel, Heiko Falk, Sascha Plazar, Robert Pyka, Lars Wehmeyer: *Automatic mapping to tightly-coupled memories and cache locking (presentation)*, 4th HiPEAC Industrial Workshop on Compilers and Architectures, Cambridge / UK, November 2007.

TU Berlin

- L. Gesellensetter, S. Glesner, E. Salecker: *Formal Verification with Isabelle/HOL in Practice: Finding a Bug in the GCC Scheduler*, Formal Methods for Industrial Critical Systems (FMICS'07), Revised Selected Papers, Springer LNCS 4916, 2008.
- L. Gesellensetter, S. Glesner: *Interprocedural Speculative Optimization of Memory Accesses to Global Variables*, Euro-Par 2008, Springer LNCS 5168, 2008.

TU Vienna

- A. Prantl, M. Schordan, J. Knoop: *TuBound - A Conceptually New Tool for Worst-Case Execution Time Analysis*. To appear in Post-Workshop Proceedings of the 8th International Workshop on Worst-Case Execution Time Analysis (WCET 2008), Prague, Czech Republic, July 1, 2008.
- R. Kirner, A. Kadlec, P. Puschner, A. Prantl, M. Schordan, J. Knoop: *Towards a Common WCET Annotation Language: Essential Ingredients*. To appear in Post-Workshop Proceedings of the 8th International Workshop on Worst-Case Execution Time Analysis (WCET 2008), Prague, Czech Republic, July 1, 2008.
- Markus Schordan: *Source-To-Source Analysis with SATIrE - an Example Revisited.* In Proceedings of Dagstuhl Seminar 08161: Scalable Program Analysis, 17 pages, Germany, Dagstuhl, April 2008.

AbsInt

- C. Ferdinand, R. Heckmann, M. Jersak, F. Martin, K. Richter: Integrating System-Level and Code-Level Timing Analysis for Dependable System Development. 4th European Congress ERTS - Embedded Real Time Software, January 29, 30, 31, February 1, 2008, Toulouse, France.
- C. Ferdinand, R. Heckmann, M. Jersak, F. Martin, K. Richter: Integrating System-Level and Code-Level Timing Analysis for Dependable System Development. 4th European Congress ERTS - Embedded Real Time Software, January 29, 30, 31, February 1, 2008, Toulouse, France.

IMEC

- R. Baert, E. de Greef, E. Brockmeye: *An automatic scratch pad memory management tool and MPEG-4 encoder case study*. In Proceedings of the 45th Annual Conference on Design Automation (Anaheim, California, June 08 - 13, 2008). DAC '08. ACM, 201-204, 2008.
- A. Bartzas, M. Peon-Quiros, S. Mamagkakis, F. Catthoor, D. Soudris, J.M. Mendias: *Enabling run-time memory data transfer optimizations at the system level with*

automated extraction of embedded software metadata information. In Proceedings of the 2008 Conference on Asia and South Pacific Design Automation (Seoul, Korea, January 21 - 24, 2008). ASP-DAC '08. IEEE, 434-439, 2008.

2.4.3 Interaction and Building Excellence between Partners

Interaction between RWTH Aachen and ACE

Aachen and ACE continue to maintain their strong cooperation. The main indicators are information exchange by e-mail, joint publications and ongoing student exchanges, which sends students from Aachen for internships or thesis writing to work at ACE offices.

Interaction between TU Berlin and ACE

Researchers from Berlin attended the CoSy Community Gathering that was held at ACE in September 2008. Further interaction between ACE and TU Berlin has been via email with regular support questions and mentoring in the use of the CoSy platform as the subject.

Interaction between AbsInt and TU Vienna:

The interactions between AbsInt and TU Vienna have been on a daily basis since Y1. After a meeting in Y2, AbsInt and TU Vienna organized again a project-meeting in Saarbrücken in April 2008. This allowed for discussing the various aspects of the integration of PAG in SATIrE and possible connections to aiT in more detail. In particular, plans for further cooperations beyond ARTIST2 were sketched and have become concrete in the continued cooperation in the ALL-TIMES project.

Interaction between AbsInt and Dortmund:

AbsInt is supporting Dortmund in its effort to build a WCET-aware compiler by integrating aiT technology into the compiler framework.

Interaction between AbsInt and USaar:

AbsInt and USaar are cooperating on the integration of code synthesis, compilers, and timing analysis, on timing predictability, on scheduling, and on analysis of multi-core architectures.

Interaction between AbsInt, TU Vienna, and USaar:

AbsInt and Vienna are jointly improving the SATIrE interface between ROSE and PAG. They are supporting USaar in using SATIrE for the implementation of a loop bound analysis and a function pointer analysis on C/C++ level.

2.4.4 Joint Publications Resulting from these Achievements

TU Dortmund / AbsInt

- Paul Lokuciejewski, Heiko Falk, Peter Marwedel, Henrik Theiling: *WCET-Driven, Code-Size Critical Procedure Cloning*, Proceedings of the 11th International Workshop on Software & Compilers for Embedded Systems (SCOPES), Munich, Germany, March, 2008.
- Niklas Holsti, Jan Gustafsson, Guillem Bernat (eds.), Clément Ballabriga, Armelle Bonenfant, Roman Bourgade, Hugues Cassé, Daniel Cordes, Albrecht Kadlec, Raimund Kirner, Jens Knoop, Paul Lokuciejewski, Nicholas Merriam, Marianne de Michiel, Adrian Prantl, Bernhard Rieder, Christine Rochange, Pascal Sainrat, Markus Schordan, *WCET TOOL CHALLENGE 2008: REPORT*, to appear.

Aachen, ACE

- M. Hohenauer, F. Engel, R. Leupers, G. Ascheid, H. Meyr, RWTH Aachen University; G. Bette, ACE; B. Singh, NXP Semiconductors Eindhoven: Retargetable Code Optimization for Predicated Execution,. In DATE, Munich, Germany, March 2008.

AbsInt, USaar

- Christian Ferdinand, Florian Martin, Christoph Cullmann, Marc Schlickling, Ingmar Stein, Stephan Thesing, and Reinhold Heckmann: *New Developments in WCET Analysis*. In Thomas Reps, Mooly Sagiv, and Jörg Bauer, editors, Program Analysis and Compilation, Theory and Practice. Essays Dedicated to Reinhard Wilhelm on the Occasion of His 60th Birthday (Lecture Notes in Computer Science 4444), pages 12-52. Berlin, Springer, 2007.

2.4.5 Keynotes, Workshops, Tutorials

Workshop : Software & Compilers for Embedded Systems (SCOPES) 2008
Munich, Germany – March 13-14, 2008

The influence of embedded systems is constantly growing. Increasingly powerful and versatile devices are developed and put on the market at a fast pace. The number of features is increasing, and so are the constraints on the systems concerning size, performance, energy dissipation and timing predictability. Since most systems today use a processor to execute an application program rather than using dedicated hardware, the requirements can not be fulfilled by hardware architects alone: Hardware and software have to work together to meet the tight constraints put on modern devices.

One of the key characteristics of embedded software is that it heavily depends on the underlying hardware. The reason of the dependency is that embedded software needs to be designed in an application specific way. To reduce the system design cost, e.g. code size, energy consumption etc., embedded software needs to be optimized exploiting the characteristics of the underlying hardware.

SCOPES focuses on the software generation process for modern embedded systems. Topics of interest include all aspects of the compilation process, starting with suitable modeling and specification techniques and programming languages for embedded systems. The emphasis of the workshop lies on code generation techniques for embedded processors. The exploitation of specialized instruction set characteristics is as important as the development of new optimizations for embedded application domains. Cost criteria for the entire code generation and optimization process include runtime, timing predictability, energy dissipation, code size and others. Since today's embedded devices frequently consist of a multi-processor system-on-chip, the scope of this workshop is not limited to single-processor systems but particularly covers compilation techniques for MPSoC architectures.

In addition, this workshop intends to put a spotlight on the interactions between compilers and other components in the embedded system design process. This includes compiler support for e.g. architecture exploration during HW/SW codesign or interactions between operating systems and compilation techniques. Finally, techniques for compiler aided profiling, measurement, debugging and validation of embedded software are also covered by this workshop, because stability of embedded software is mandatory.

SCOPES 2008 is the 11th workshop in a series of workshops initially called "International Workshop on Code Generation for Embedded Processors". The name SCOPES has been used since the 4th workshop. The scope of the workshop remains software for embedded systems with emphasis on code generation (compilers) for embedded processors.

SCOPES 2008 was organized by Heiko Falk from TU Dortmund and was held as DATE Friday Workshop.

http://www.scopesconf.org/scopes-08

Workshop : Dagstuhl Seminar 08161 "Scalable Program Analysis"
Schloss Dagstuhl*, Germany – April, 2008*

Organizers: Florian Martin (AbsInt), Hanne Riis Nielson (Technical University of Denmark), Claudio Riva (NOKIA Research Center - Helsinki), Markus Schordan (TU Vienna).

The goal of the seminar was to bring together researchers from academia and industry to discuss the strengths and weaknesses of state-of-the-art program analysis technology for industrial-sized software. To achieve that goal the seminar gathered 38 participants from 9 companies and 23 academic/research institutions.

The seminar showed how broad the field of program analysis has grown over the years. Traditionally used in optimizing compilers, program analysis has turned into a major discipline with techniques and commercial tools supporting understanding, maintaining, and engineering of software. It often turned out that an in-depth discussion of scalability requires further investigations. The scalability of analysis techniques is a major issue as the size of

software systems is rapidly growing and the automatic analysis of those systems is becoming yet more important in future. Many questions were raised about scalability - to address the scale of today's systems, analyses will have to be run as parallel programs in future, posing themselves as problem of being scalable on multiple cores, but also whether it can be applied to multiple parts of a system which may differ in structural properties of the code or even in used programming languages.

Raising the awareness about the many faces of scalability is the major achievement of the seminar. As the seminar progressed, increasingly more questions about scalability were raised, mostly asking for more extensive evaluations of the methods & tools in future. Discussions about the need and development of specific benchmarks for scalability were started and agreed to be continued past the seminar by different groups of the seminar. Carefully systematically designed sets of test cases, accompanied by test cases from industry, were considered a good setting for evaluating scalability.

http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=08161

Workshop : CoSy Community Gathering 2008
Amsterdam, The Netherlands, September 2008

The CoSy Community Gathering 2008 has been held at the Amsterdam Public Library. Researchers from RWTH Aachen and TU Berlin attended the event and reported on their CoSy related work.

http://www.opencosy.org/

Full Day tutorial: Peter Marwedel, Embedded Systems in a Nutshell, Spring School on Knowledge Discovery in Ubiquitous Systems,
Porto, Portugal, March 2, 2008

This tutorial provided a brief overview over specification techniques, hardware, scheduling and optimization of embedded systems for a community without any pre-existing knowledge on embedded systems.

http://www.kdubiq.org

Invited Course : Peter Marwedel, Heiko Falk, Embedded Systems with Emphasis On the Exploitation of the Memory Hierarchy
Advanced Institute of Information Technology
Seoul, Korea – August 11-15, 2008

The goal of this course is to provide an overview over key areas in embedded system design which should be taught at Universities. After attending the course, the attendees should be able to compare different approaches to embedded system design education and their advantages and limitations. The attendees will also become familiar with the contents of a course on embedded system design which aims targets second or third year students. The course should enable attendees to design the structure of embedded system education at their universities. In the last third of the course, attendees will be introduced to research topics regarding embedded system optimization. In particular, this last third will address the so-called memory wall problem (the problem resulting from the small performance improvements of memories). This problem is frequently seen as the key problem for further performance enhancements of future systems. This material would be appropriate for an advanced course in embedded system design.

Peter Marwedel and Heiko Falk from TU Dortmund lectured this one-week course for Korean professors (CS and EE) after an invitation by the Korean Advanced Institute of Information Technology.

http://ttt.aiit.or.kr

Tutorial: Peter Marwedel: Memory architecture aware compilation for Embedded Systems (5 lectures of 2 hrs each)
Artist South American Summer School
Florianopolis, Brazil, Aug. 25.-29., 2008

The tutorial focused on compilation techniques exploiting descriptions of the memory architecture.

Lecture : Peter Marwedel, Heiko Falk, Memory-architecture aware compilation
The ARTIST2 Summer School 2008 in Europe
Autrans, France – September 8-12, 2008

The memory system is increasingly turning into a bottleneck in the design of embedded systems. The speed improvements of memory systems are lower than the speed improvements of processors, eventually leading to embedded systems whose performance is limited by the memory. This problem is known as the “memory wall” problem. Furthermore, memory systems may consume the largest share of the system’s energy budget and may be the source of unpredictable timing behavior.

In their lecture during the ARTIST2 Summer School in Europe, Peter Marwedel and Heiko Falk from TU Dortmund present optimization techniques leading to high performance, low energy consumption and tight worst-case execution times by making efficient use of scratchpad memories and processor registers.

http://www.artist-embedded.org/artist/ARTIST2-Summer-School-2008.html

3. Milestones, and Future Evolution Beyond the NoE

3.1 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.

Year2: 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 is using CoSy for research on new compiler optimization and verification technologies.

Year3: New timing-aware optimizations are available at the end of year 3.

This milestone has been achieved, the partners achieved first results with the platform. E.g. Aachen has results on the SIMD support and on retargetability of conditional execution. Dortmund established a WCET-aware compiler and was able to improve the program performance by instruction cache optimizations and procedure cloning. TU Berlin established the Itanium compiler platform and formalized the scheduling phase of a compiler. By that, they were able to show a bug in the popular GCC assembler.

Year4: More timing-aware optimizations are available at the end of year 4.

This milestone has been achieved, the partners could obtain further results based on last year's achievements. TU Dortmund developed several WCET-aware optimizations based on the infrastructure set up during earlier years of Artist2. TU Berlin has considered two approaches to mitigate the implications of the memory wall. In one case, notable performance improvement could be shown. In the other case, they could show that predictors for memory dependency can be automatically built via statistical machine learning techniques. Besides, TU Berlin successfully investigated how relevant aspects of two crucial phases in the compiler, namely scheduling and code generation, could be verified formally to prove them correct. TU Vienna investigated the transformation of source-level WCET annotations in the presence of loop optimizations. This work involved the entire tool chain built in Y1-Y4 and served as use case of the established tool connections between AbsInt and TU Vienna. As a result it was possible to compute the WCET for non-optimized and optimized versions of the Mälerdalen Benchmarks.

3.2 Indicators for Integration

As we expected, this activity led to a high-quality compiler platform prototype whose capabilities include many existing and newly developed techniques which previously have

been largely separated due to heterogeneous compiler platforms in use by the different partners. As a consequence, high-level transformations are now 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. TU Berlin and ACE also worked successfully on integration by developing verification methods and optimizations within the CoSy compiler framework. TU Vienna and AbsInt further integrated SATIrE and PAG. The work in Y4 continued on daily basis and motivated new PAG features and extensions of SATIrE for integration of other timing tools. Based on this platform a new tool, TuBound, was created for source-level based WCET analysis.

3.3 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.

PREDATOR, STREP proposal within 7th FP.

Absint receives funding from different BMBF and EU projects.

Resources of the University of Dortmund and of the Technical University of Berlin.

3.4 Future Evolution Beyond the Artist2 NoE

In the future, TU Dortmund will keep on studying WCET-aware compiler optimizations. The techniques developed so far can be extended towards dynamic scratchpad memory allocation which means that an optimized program will be allowed to modify its scratchpad contents dynamically at runtime. Supporting multi-process applications and MPSoCs will be an important aspect of Dortmund's future work. Furthermore, multi-objective optimization techniques trading-off e.g. hard real-time constraints versus energy dissipation will be interesting to investigate.

The results achieved within the Artist2 NoE at the TU Berlin constitute a promising fundament for further research and cooperation. For compiler optimization, we could show that the impact of the memory wall can be mitigated by our techniques, and in further work, we will extend these results. For compiler verification, we could successfully formalize the scheduling phase, which yielded correctness criteria a correct scheduler has to fulfill, which could be used to identify a bug in the scheduler of the GNU Itanium assembler. Besides, we have promising first results for the verification of the code generation phase. We are convinced that our further research continues to benefit from cooperation with Artist partners, especially with ACE (CoSy infrastructure) and the University of Edinburgh (Machine Learning techniques).

Several ARTIST2 activity partners also participate in the FP7 project ALL-TIMES (Integrating European Timing Analysis Technology), which will run until Feb. 2009. A major theme in ALL-TIMES is the definition and implementation of open interfaces between timing analysis tools, followed by an integration where different tool combinations are investigated. An explicit goal of ALL-TIMES is the further development of analyses on the C/C++ source level and their integration with analyses on the binary level. For code level timing analysis, the work is very much in line with the ARTIST2 platform integration work and paves the way for its future

exploitation. ALL-TIMES also considers the integration with system level tools, such as Symta/S from Symtavision. This provides another line of development from the platform integration work performed within ARTIST2.

The connection between AbsInt and TU Vienna, which has been established through ARTIST2, has proven fruitful for both sides. Both partners have extended their activities in source-level analyses of C/C++ codes for embedded systems. Building on this successful cooperation, both partners continue to further develop timing technology beyond ARTIST2 in the ALL-TIMES project.

IMEC will continue the research and development of source-to-source pre-compiler tools that optimize the parallelization and the data memory hierarchy assignment in the context of FP7 STREP IST projects MNEMEE (IST-216224) and MOSART (IST-215244).

4. Internal Reviewers for this Deliverable

Dirk Tetzlaff, TU Berlin

Peter Marwedel, TU Dortmund