IST-004527 ARTIST2
Network of Excellence
on Embedded Systems Design

Activity Progress Report for Year 3

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 the common activities will build. This will reinforce Europe's leading position in the area of compilers for embedded processors.

Table of Contents

1. Overview of the Activity 4
1.1 ARTIST Participants and Roles 4
1.2 Affiliated Participants and Roles 4
1.3 Starting Date, and Expected Ending Date 4
1.4 Baseline 4
1.5 Problem Tackled in Year 3 5
1.6 Comments From Year 2 Review 6
1.6.1 Reviewers' Comments 6
1.6.2 How These Have Been Addressed 6

2. Summary of Activity Progress 7
2.1 Previous Work in Year 1 7
2.1.1 Cooperation Absint – TU Vienna 7
2.1.2 Cooperation IMEC – University of Dortmund 7
2.2 Previous Work in Year 2 7
2.3 Current Results 9
2.3.1 Technical Achievements 9
2.3.2 Individual Publications Resulting from these Achievements 12
2.3.3 Interaction and Building Excellence between Partners 13
2.3.4 Joint Publications Resulting from these Achievements 14
2.3.5 Keynotes, Workshops, Tutorials 14

3. Future Work and Evolution 17
3.1 Problem to be Tackled over the next 12 months (Sept 2007 – Aug 2008) 17
3.2 Current and Future Milestones 17
3.3 Indicators for Integration 18
3.4 Main Funding 18

4. Internal Reviewers for this Deliverable 19

1. Overview of the Activity

1.1 ARTIST Participants 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 University (Germany)
Leader of Compilers cluster, Software for Systems on Silicon.

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

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 and Roles

Dr. Stylianos Mamagkakis, 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.

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 August 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 a cooperation between the Technical University of Berlin and ACE,verification methods and tools for compilers are investigated.

1.5 Problem Tackled in Year 3

We saw that timing models need to be integrated into compilers. Based on the work on a timing-aware compiler infrastructure performed in Year 2 of Artist2, Year 3 dealt with studies of standard compiler optimzations developed to reduce the program's average-case execution time and their influence on the WCET. As an outcome of the current reporting period, it could be shown that the compiler optimization Procedure Cloning allows improved WCET estimations since it makes the code more predictable. Additionally, WCET-aware memory hierarchy exploitation using locked instruction caches was studied. The work on timing-aware compiler optimizations was published on international conferences and workshops of very high quality.

Continued joint work on an SIMD optimisation framework and Conditional Execution with strong cooperation between Aachen and ACE has involved several members of ACE's core compiler team, an Aachen intern working at ACE for several months and a number of researchers at Aachen. The results were integrated ino CoSy for use by CoSy users in beta form. The SIMD work required considerable work on data dependency analysis and support for interprocedural pointer alignment analysis. CoSy's generic target description had to be extended to faclitate SIMD retargetability. Similarly, building on the research conducted previously by ACE and Aachen, it was possible to define CoSy extensions able to deal with the Conditional Execution capabilities supported by the next generation of hardware.

Berlin has been making extensive use of CoSy within a project examining the correctness of compiler transformations. ACE's role in this has been one of support and advice on the use of CoSy. Berlin engineered an Itanium compiler as a test bed for their research in this area.

IMEC realized multiple fine-grain dynamic memory allocation design options in software modules of the standard compiler library (eg, *libc*), which can be parameterized and combined in many ways according to energy efficiency criteria.

1.6 Comments From Year 2 Review

1.6.1 Reviewers' Comments

The same structure as for D14 would have been nice to see.

1.6.2 How These Have Been Addressed

The structure of D14 as well as of D15 followed the given template. As only structural difference between D15 and D14, D15 did not contain a "Comments From Previous Review" nor a "Internal Reviewers" section. This has been changed in this year's deliverable.

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 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 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

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 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 Current Results

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 priorily done has been shown by adding support for an 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 of the correctness of it. 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 a 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 Dortmunt Uni.) is the the realization of multiple fine-grain dynamic memory allocation design options in software modules of the standard compiler library (eg, *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 to provide 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 to annotate 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

Dortmund University

- 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 progam 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 has been added and integrated to create the Static Analysis Tool Integration Engine, SATIrE. The cooperation with AbsInt allowed to perform 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

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 arre 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

Aachen presented the results of its research at DATE in Nice. A specially organised OpenCoSy stand for academic users of CoSy – www.opencosy.org/announcements. This stand proved very attractive to attendees over the course of the week wth 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/

3. Future Work and Evolution

3.1 Problem to be Tackled over the next 12 months (Sept 2007 – Aug 2008)

Further studies on the influence of standard compiler optimizations on the program's WCET will be continued and the development of novel 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) in order to integrate flow analysis techniques into Dortmund's compiler.

The Dortmund-IMEC collaboration will tackle optimization problems, which address the placement of all data in the distributed/shared data memory hierarchy of a MPSoC platform.

ACE and Aachen will continue their cooperation in the area of architecture aware code optimization engines built around the CoSy platform. This includes: extensions of SIMD optimization and utilization of conditional instructions, and possibly code generation for special-purpose multi.output instructions. To support this cooperation, joint papers are in preparation, and further student exchanges are envisioned. Moreover, ACE and Aachen may engage to promote the new OpenCoSy web portal for simplified exchange of compiler research results.

ACE and TU Berlin will continue their cooperation for further improving the Itanium compiler platform. TU Berlin will develop more optimization techniques to overcome the impact of the memory wall. To this end, more precise alias analyses, which use heuristics, will be considered, and on the other hand novel speculative optimizations, that make use therof. Besides, we will formalize important parts of the semantics of the Itanium assembler code, to be able to bring more verification into the compiler, e.g. by verifying the correctness of the code generation phase.

TU Vienna will further extend the Static Analysis Tool Integration Engine, SATIrE, 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 in developing program analyses with PAG and generating WCET annotations.

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.

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 will be available at the end of year 4.

In **Year 4**, more and more new technologies like code optimization and verification will be integrated into the common platform. This should then 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 include 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.

TU Berlin and ACE also work successfully on integration by developing verification methods and optimizations within the CoSy compiler framework.

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 and of the Technical University of Berlin

4. Internal Reviewers for this Deliverable

- Prof. Dr. Rainer Leupers, RWTH Aachen
- Lars Gesellensetter, Technical University of Berlin