ARTIST 2

Network of Excellence

IST-004527 ARTIST2:
Embedded Systems Design

Activity Progress Report for Year 1

JPRA-Cluster Integration:

Architecture-aware Compilation

Cluster:

Compilers and Timing Analysis

Activity Leader:

Rainer Leupers (RWTH Aachen)

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

Table of Contents

1. Introduction

1.1 Activity Leaders

Rainer Leupers (RWTH Aachen)
Areas of his team's expertise: co- leader, code optimization, retargetable compilation.

1.2 Policy Objective

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

1.3 Industrial Sectors

Mainly: Embedded software, semiconductor and system houses

Specifically: audio processing, video processing and data streaming applications in the TV, Set Top box, DVD player and recorder, mobile, base stations, printer and disk drive markets.

2. Overview of the Activity

2.1 Artist Participants and roles

Team Leader: Reinhard Wilhelm (Saarland University)
Areas of his team's expertise: co-leader, program analysis tools.

Team Leader: Rainer Leupers (RWTH Aachen)
Areas of his team's expertise: co-leader, code optimization, retargetable compilation.

Team Leader: Christian Bertin (STMicroelectronics)
Areas of his team's expertise: driver applications, resource-aware code generation.

Team Leader: Christian Ferdinand (AbsInt)
Areas of his team's expertise: Program-Analysis Tool.

Peter Marwedel (Dortmund Univ.)
Areas of his team's expertise: low power compilation.

2.2 Affiliated partners and Roles

Team Leader: Hans van Someren (ACE)
Areas of his team's expertise: compiler development platform.

Team Leader: Francky Catthoor (IMEC)
Areas of his team's expertise: high-level code optimization.

Team Leader: Andreas Krall (TU Vienna)
Areas of his team's expertise: code optimization.

2.3 Starting date, and expected ending date

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

2.4 Baseline

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

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

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

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

2.5 Technical Description

On the basis of the JPIA: Compiler Platform, and existing competencies, each partner will integrate new functionalities:

ACE will provide the CoSy compiler system as a common platform. Saarland Univ. will be working on integration of advances program analyzers. Aachen's contribution will be on automatic compiler retargeting. STM will focus on code optimization modules and multiprocessor compilation. Dortmund Univ. will add high-level optimizations for low power. In addition, a tighter cooperation between Dortmund University and AbsInt will be established in order to develop WCET-aware compiler optimizations.Further additions will be made by affiliated partners.

Altogether this will lead to a significant extension of the compiler platform capabilities with benefits for users within and beyond ARTIST.

3. Activity Progress Report

3.1 Work achieved in the first 6 months

[Reviewer comment: The document must be revised to incorporate a discussion of the requirements analysis that was done by the cluster]

Initial requirements analysis: The cluster partners have been discussing current and future evolutions of embedded processor architectures to identify a list of requirements for the architecture aware compilation activitiy. As there are a huge amount of potential, and equally important, areas of research in architecture aware compilation, a core set of activities have been filtered out such that (1) sufficient special expertise is already available with the different partners and this expertise can be leveraged and combined by means of cooperations at the mini-cluster level; and (2) useful results can be likely produced with the limited resources available in the ARTIST2 NoE framework. As a conclusion, the following promising activity areas have been identified:

- Support for special instruction set features: Some modern embedded processors show dedicated instruction set architecture (ISA) features that are not yet well supported by today´s compiler technology. However, such features are likely to be a part of most future embedded processors. Among these are SIMD (single instruction, multiple data) instructions, which are extremely important for multimedia applications but which require special code selection techniques. Furthermore, many ISAs support conditional instructions (or predicated execution) that help to minimize code size and prevent performance losses due to pipeline hazards. Again, special compiler engines are required to support this feature.
- Memory-aware compilation: It is well known that a large part of program runtime as well as power/energy consumption is due to memory accesses in an application. The simple traditional model of a flat memory architecture does not hold anymore for modern architectures, which tend to show a hierarchical memory subsystem, including scratch-pad memories as well as different cache levels. Novel code optimization techniques are required to make the compiler more aware of the memory subsystem. Especially, this holds for scratch-pad memories, which are under full compiler control and which can provide a large performance and energy consumption gain. Corresponding code optimizations have to be investigated both at the source code level (for better reusability) and the assembly level (for exploiting machine-specific information). Better incorporation of the memory subsystem in the compiler also leads to better predictability of the application´s worst case execution time (WCET).
- (Re)configurable architectures: Application-specific processor architectures are equipped with more and more flexibility, both in the pre-fabrication and post-fabrication phases. This means that a compiler for such (re)configurable architectures cannot assume a fixed ISA, but need to work for ISAs that more or less vary over the product lifetime. This requires very flexible compiler technology, where new special ISA features can be incorporated by the end-user (i.e. a non.expert) without major changes of the compiler source code.

The main result achieved in the co-operation between IMEC an Univ. Dortmund is a thorough alignment of the research objectives for one particular objective, i.e. the steering of locality-improving loop transformations at the source code level. For this purpose, it is crucial that the impact on the control flow complexity of the applied source code trafo can be evaluated at an early stage, before actually performing all the alternatives and compiling the resulting code on the target platform. The requirement (the WHAT specification) to tackle this problem have been defined.

A cooperation has been established between University of Dortmund and RWTH Aachen University. The objective is to integrate SIMD optimizations developed at University of Dortmund into the LISATek Embedded Processor Designer based tool chain, developed at RWTH Aachen University. The Philips TriMedia architecture has been chosen as a target platform. For this architecture a very basic LISA model was already available. The first result has been a refined TriMedia model to provide sufficient information to generate the complete tool chain (CoSy based C-compiler, assembler, linker, …) with the LISATek Embedded Processor Designer. Further extensions to the TriMedia model have been done for the purpose of integrating SIMD optimizations into the tool chain. The generated tools include a programming interface (Assembler Optimizer API) through which additional post-pass optimizations on assembly level may be easily incorporated into the framework. For this cooperation's purposes this interface has been extended. According to our goal to implement SIMD optimizations, all of the information about architectural properties has to be provided by the API. The main limitations have been identified, and the API has been extended appropriately.

3.2 Work achieved in months 6-12

Based on a the WHAT specification, PhD research work has been initated at Univ. Dortmund to evaluate the alternative ways to come up with such a high-level control flow cost estimator that could base its estimate when only the source code is available (without performing any compilation). At IMEC complementary actions have been started to see how this estimator can be integrated in a loop transformation framework project that has been started up earlier (prior to the start of ARTIST2) and that is now being extended for these high-level estimators.

Since safety-critical embedded real-time systems have to be highly predictable in order to guarantee at design time that certain timing deadlines will always be met, caches are usually not used due to their hardly predictable behavior. The integration of scratchpad memories instead of caches represents an alternative approach which allows the system to benefit from a performance gain comparable to that of caches while at the same time maintaining predictability. During the work done in cooperation between Dortmund University and AbsInt, the impact of scratchpad memories and caches on WCET was analyzed. It was shown that caches can have a negative impact on WCET, while WCET for scratchpad memories scales with the achieved performance gain.

For the Aachen-Dortmund cooperation, once all software tools have been in place and the interfaces have been defined, the integration of the actual optimizations was tackled next. A number of basic analysis technologies have been ported to the API based optimization tool. A bit-precise representation of integers and register values has been integrated, a semantics based control flow analysis has been implemented, a function scope dataflow graph representation is part of the implementation and finally, the bit true dataflow analysis has been integrated into the tool. All of the analysis techniques base on the semantic information provided by the LISA model. Only a very limited set of additional architectural properties needs to be setup in the case of retargeting the tool to a new architecture. As a proof-of-concept, two basic optimizations have already been implemented. Additionally, work has been done on semantics based classification of instructions.

In the STM-ACE cooperation, reconfigurable processors are replacing specialized hardware blocks and can provide significant gains for embedded SOC development, especially in terms of time to market. They are more generic and flexible than hardware blocks and allow code reuse. Usually structured around a minimal core, reconfigurable processors may accept new extensions. An extension can include new resources (register banks especially) and/or new instructions dedicated to a class of applications. Dealing with those processors at compiler level implies new capabilities, especially:

* if an extension can be defined like a reconfigurable technology, the compiler must access to and make use of a software description or a 'machine model' of the extension. Though, this model must be read dynamically at compile time, and may have to deal with more numerous and different features

* the compiler and the rest of the toolset must have the ability to be configured to make use of new instructions and resources, with a minimal effort. Ideally, this should be feasible at end-user level. However, this can be done only if one defines what an extension can be. Otherwise, the support through user-defined built-ins can provide a first and light solution.

ST thus has worked on adapting their FlexCC retargetable compiler technology based on CoSy to deal with reconfigurable processors. ST

focus is on:

* the implementation of a mechanism allowing easy definition of user defined built-in functions. Those later can be translated into either a single machine instruction, or into a more complex sequence of code,

* the definition on the best way to access the architecture model in a dynamic way from the compiler,

* reflexion about what an 'extension' to a given core or a reconfiguration feature can be, and what it may imply in terms of compiler support (ABI, data type mapping, code generation, register allocation...).

* the implementation of a binary tool reconfiguration toolkit based on the LISA language.

3.3 *Difficulties Encountered*

No particular difficulties have been encountered.

3.4 *Recommendations*

It is recommended to retain the established mini-cluster structure, yet be open to new partners with complementary expertise.

3.5 *Milestones*

During the past months, the cooperation between Dortmund University and AbsInt was established and has lead to the exchange of several software components between the partners (aiT, CRL intermediate representation, ...). The effect of memory related compiler optimizations developed at Dortmund on WCET was successfully analyzed, the results were published at the WCET workshop 2004 and the DATE conference 2005.

Aachen-Dortmund cooperation:

1. A TriMedia LISA model from which the complete software tool chain including CoSy based C-compiler, assembler, linker, simulator, assembler optimizer API is generated. Established a working path between the assembler optimizer API and the optimization tools from Dortmund.

2. Finished an early version of the assembler optimizer API based SIMD optimization tool which can perform two basic optimizations.

3.6 Main Funding

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

Main sources of funding at IMEC are related to bilateral industry-IMEC programmes. Also a Marie Curie host fellowship programme is active that would allow supporting exchanges of PhD students. A proposal for a Marie Curie Training Network is planned for the Sept'.05 call to further provide support.

3.7 Indicators for Integration

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

3.8 Evolution

Integration of the Program-Analyzer Generator (PAG) will be done during the first 12 months. STM will then integrate existing test cases on top of this during the following 6 months. Loop-splitting algorithms of Dortmund University will be integrated into the IMEC tool flow during the first 12 months. Development of prototypes during ARTIST, further developments and technology transfer to industry also beyond the ARTIST contract period.

A tighter cooperation between Dortmund University and AbsInt will lead to the development of WCET-aware compiler optimizations during the first 12 months.

Aachen-Dortmund cooperation:

The future steps to be taken can be identified as: Improving the model for instruction classification. Based on this classification more sophisticated optimizations can be implemented; like identifying saturated arithmetic in the program code. Finally, a third work topic could be a further refinement of the Assembler Optimizer API to provide additional architectural properties (e.g. register set information).

4. Detailed Technical View

4.1 Brief State of the Art

With the increasing level of customization of embedded processors it becomes more and more obvious that architecture aware compilation is a must to achieve sufficient code quality. Application of only classical code optimization techniques, largely working at the machine-independent intermediate representation level is not good enough. Therefore, members of the ARTIST2 compiler cluster have designed numerous novel code optimization techniques, e.g. Dortmund and Aachen have extensively worked on optimizations for DSP, VLIW and network processors. This work is being continued in the context of ARTIST2.

Simultaneously, new code optimization goals have appeared. For instance, in spite of powerful optimizing code transformations at the IR or assembly level, the resulting code can be only as efficient as the source code passed to the compiler. For a given application algorithm, an infinite number of C code implementations exist, possibly each resulting in different code quality after compilation. For instance, downloadable reference C implementations of new algorithms are mostly optimized for readability rather than performance, and high-level design tools that generate C as an output format usually do not pay much attention to code quality. Even if they are optimised for performance, this happens typically for the processor of the host station (like a PC or workstation) and not with sufficient eye for more platform-independent higher-level code optimisations. In addition, the performance is not the only important criterion. Especially for embedded platforms or memory footprint and energy consumption (see further) are important targets. This motivates the need for code optimizations at the source level, e.g. C-to-C transformations that complement the optimizations performed by the compiler, while retaining the program semantics. Moreover, such C-to-C transformations are inherently retargetable, since the entire compiler is used as a backend in this case. Currently proposed techniques exploit the implementation space at the source level to significantly optimize code quality for certain applications, while tools like PowerEscape focus on efficient exploitation of the memory hierarchy in order to minimize power consumption of C programs.

Furthermore, low power and/or low energy consumption have become primary design goals for embedded systems. As such systems are more and more dominated by software executed by programmable embedded processors, it is obvious that also compilers may play an important role, since they control the code efficiency. At first glance, it appears that program energy minimization is identical to performance optimization, assuming that power consumption is approximately constant over the execution time. However, this is only a rule-of-thumb, and the use of fine-grained instruction-level energy models shows that there can be a trade-off between the two optimization goals, which can be explored with special code generation techniques. The effect is somewhat limited, though, when neglecting the memory subsystem, which is a major source of energy consumption in SoCs. Once the memory hierarchy is included, major trade-offs between performance and energy consumption have been observed in systematic research studies. The IMEC and Univ. of Dortmund partners have contributed to that in a major way in the past. More optimization potential is offered by exploitation of small on-chip (scratchpad) memories , which can be treated as entirely compiler-controlled, energy efficient caches. Dedicated compiler techniques are required to ensure an optimum use of scratchpads for program code and/or data segments.

4.2 Industrial Needs and Experience

With the increasing acceptance of application specific instruction set processors (ASIPs) as efficient and flexible implementation vehicles in embedded system-on-chip SoC design, more and more EDA platforms (e.g. LISATek, Axys, Target, Tensilica) are available for ASIP architecture exploration and design. These platforms comprise retargetable software development tools, including C compiler, instruction set simulator, debugger, and (dis)assembler, enabling the designer to quickly explore ASIP architectural alternatives for a given range of embedded applications. A key component of many of these platforms is the retargetable C compiler, which can, automatically or semi-automatically, be adapted to generate code for different target architectures. While retargetable compilers have found significant use in ASIP design in the past years, they are still hampered by their limited code quality as compared to hand-written compilers or assembly code. This is no surprise, since higher compiler flexibility comes at the expense of a lower amount of target-specific code optimizations. Therefore, it is not uncommon to manually enhance a generated compiler with target-specific optimizations, once the ASIP architecture exploration phase has converged and an initial working compiler is available.

A promising approach to further reduce ASIP compiler design effort is to identify target processor classes which, due to their architectural features, demand for specific code optimization techniques, and to implement these specific techniques such that retargetability for the given processor class is achieved. An example is the retargetable software pipelining support recently introduced for the CoSy compiler platform. While being less useful for scalar architectures, software pipelining is a necessity for the class of VLIW processors, and for this class it can be designed in a retargetable (or configurable) fashion.

The ARTIST2 compiler cluster responds to these trends via different architecture aware code optimization activities. For instance, Dortmund and Aachen have started a cooperation that targets the class of embedded processors with SIMD instruction sets. Dortmund and IMEC have a long-standing co-opeation in the area of memory-aware code transformations. That co-operation is continued under the ARTIST2 umbrella.

4.3 Ongoing Work in the Partner Institutions

Aachen University coordinates the architecture aware compilation activities (also in the context of the ARTIST2 compiler platform activity) and performs R&D work on retargetable compilation and code optimization. In particular, Aachen has a tight cooperation with Dortmund on the topic of code optimization for SIMD instruction sets.

Absint cooperates with Dortmund University in the area of worst-case execution time aware compilation. AbsInt provides program representation formats and related libraries as well as the aiT WCET tools.

TU Vienna works on investigating the impact of sophisticated program analyses for embedded systems compilers. To ensure that the analyses can be compared on different platforms, a high-level specification of the analyses is necessary. The Program Analysis Generator from AbsInt has been chosen as tool for specifying the analyses. A substantial effort is being made in integrating PAG in different infrastructures and in automatically generating the required glue code from grammar based descriptions. The impact of program analysis results has been investigated on the instruction scheduler of the ATAIR OCE/xDSPcore compiler for embedded systems. TU Vienna also maintains a close collaboration with the Lawrence Livermore National Laboratory, CA, USA. This permits investigating the impact of PAG specified analyses on platform-dependent C/C++ source-to-source transformations as well.

IMEC has a long-standing research tradition in the source code optimisation area. Already since the late 80's data memory hierarchy oriented loop and control-flow transformations have been studied at the C code level to optimise memory footprint, energy consumption and memory related performance. Later also other transformations that change the data-flow itself have been added, and also more dynamic data types have been addressed. That work is continued in the scope of ARTIST2 with particular focus on obtaining systematic Pareto trade-offs between these different aspects for static arrays. In order to achieve this, high-level cost estimators have to be investigated and developed to estimate footprint (high-level size estimation), data reuse (to identify on-chip memory access bottlenecks) and especially also control flow complexity (to estimate additional cycles spent on more complex condition and loop constructs). In this area, a tight collaboration with Dortmund also takes place.

4.4 Interaction, Building Excellence Between Partners

Interaction between the compiler cluster partners takes place via regular global meetings every few months as well as numerous bilateral "min-cluster" meetings. Furthermore, there is an extensive exchange of software components. For instance, ACE is making the CoSy platform available free of charge to interested parties, and specific tools and interfaces are being exchanged for common R&D work. As a result, ARTIST2 has significantly contributed to the partners´ awareness of each others achievements and technologies. With the progress of interfaces being defined and implemented and the increasing shift of focus towards measurable results it is strongly expected that the ARTIST2 network will be able to boost the European excellence in compiler for embedded systems.

4.5 Spreading Excellence

As the cooperation within the compiler cluster, based on common platforms, gives a lot of opportunities for exploring new avenues in compilers for embedded processors, it is expected that a number of joint publications will soon result, some of which are already in progress. Furthermore, there are common teaching activities. Members of the University of Dortmund, RWTH Aachen, and IMEC taught at ALARI in Lugano, Switzerland. Special arrangements were made with CoWare and ACE to provide group licenses for design software used during hands-on sessions. Students at ALARI are going for a Master's degree in embedded system design. The program is organized in cooperation with industrial sponsors. The members also taught at EPFL, Lausanne. EPFL runs a continuing education program aiming at advanced PhD students and industrial participants. In both cases, research knowledge was transferred. P. Marwedel, head of the group at Dortmund, is also chair of the steering committee of the SCOPES workshop. SCOPES focusses on compilers for embedded systems. In 2005, the SCOPES workshop is being held in Dallas. In addition, an ARTIST workshop was held during DATE (Design, Automation and Test in Europe) 2005 in Munich. Furthermore, dissemination also includes the publication of the text book "embedded system design" by P. Marwedel. This book is being adopted by a growing number of Universities around the world and a cheaper paperback version will be published in 2005. The group at Dortmund is also transfering research results via the local technology transfer centre ICD. ICD works on a contract basis for industrial customers. IMEC is organizing courses that train the participants in the most mature aspects of the Data Transfer and Storage Exploration techniques that are developed in research projects like ARTIST2. These courses are open to both industry and academia.