IST-214373 ArtistDesign
Network of Excellence
on Design for Embedded Systems

Activity Progress Report for Year 3

Timing Analysis

Cluster:

SW Synthesis, Code Generation and Timing Analysis

Activity Leader:

Prof. Björn Lisper (Mälardalen University)
http://www.idt.mdh.se/~blr/

Policy Objective (abstract)

The activity gathers the most prominent groups in the timing analysis area. They have all previously worked together in the ARTIST2 NoE, and therefore have well established links. The theme of the activity, timing analysis of MPSoC systems, is basically a new field scientifically, and also very timely from an application perspective as MPSoC and Multicore architectures rapidly are becoming mainstream. A research effort in this area will thus establish European dominance in a field that rapidly is becoming very important. ArtistDesign also provides a close to perfect environment for this research due to the relevant competence in other activities and clusters, such as the compiler groups in the local cluster, and the MPSoC cluster.

Versions

number	comment	date
1.0	First version delivered to the reviewers	February 4th 2010

Table of Contents

1. Overview of the Activity

1.1 ArtistDesign participants and their role within the Activity

Prof. Dr. Reinhard Wilhelm – Saarland University (USaar; Germany)
Compiler Design, Static Program Analysis, Timing Analysis
Saarland University has developed much of the timing-analysis technology that is further developed and commercialised by the spin-off company AbsInt.

Dr. Iain Bate – University of York (UK)
Research on Timing Analysis

Prof. Dr. Björn Lisper – Mälardalen University (MDH; Sweden)
Activity Leader, Timing-AnalysisTools
Mälardalen University is working on automatic flow analysis, WCET analysis case studies on industrial code, the maintenance of a WCET-benchmark suite, the definition of interface formats for timing analysis, and the use of WCET tools in education. Mälardalen University is coordinating the integration activity.

Dr. Jan Gustafsson – Mälardalen University (MDH; Sweden)
Timing Analysis Research and Tools, WCET Analysis Case Studies

Dr. Andreas Ermedahl – Mälardalen University (MDH; Sweden)
Timing Analysis Research and Tools, WCET Analysis Case Studies

Prof. Dr. Peter Puschner – TU Vienna (Austria)
Timing-Analysis Tools and Temporally Predictable HW-SW Architectures
Within the Timing-Analysis Activity TU Vienna focuses on measurement-based WCET analysis and on hardware and software architectures that provide time-predictability and composability.

Prof. Dr. Peter Marwedel – TU Dortmund (Germany)
Architecture-Aware Compilation, Low-Power Code Generation, Development of Optimizations for WCET Minimization.

Dr. Claire Maiza/Burguière – Saarland University (USaar; Germany)

Research on Scheduling and Timing Analysis in the presence of interrupts

Dr. Raimund Kirner – TU Vienna (Austria)
Timing-Analysis Tools and Compilation with Support for Timing Analysis, definition of Annotation Language for WCET Analysis, Measurement-Based Timing Analysis.

-- Changes wrt Y2 deliverable --

No changes with respect to Year 2.

1.2 Affiliated participants and their role within the Activity

Dr. Christian Ferdinand – AbsInt GmbH (Germany)
Tool Supplier
AbsInt provides advanced WCET analysis tools for a wide variety of targets. The work within ArtistDesign focuses on the advance of WCET analysis techniques and its integration in other tools.

Gernot Gebhard – AbsInt GmbH (Germany)

Generic specification of processor components for generation and validation of timing analyses.

Dr. Niklas Holsti – Tidorum Ltd. (Finland)

Timing-Analysis Tools

Tidorum supplies the WCET analysis tool Bound-T for several targets. In ArtistDesign Tidorum aims to make the tool applicable to single cores of suitably decoupled multi-core systems, integrated within some multi-core scheduling tool.

Prof. Abhik Roychoudhury – National Univ. of Singapore (NUS; Singapore) – International affiliate

Timing Analysis

The group at National University of Singapore (NUS) has studied methods for micro-architectural modelling in timing analysis, leading to the Chronos timing analyzer. The work on Chronos has led to related works on supporting predictable execution, e.g., by developing scratchpad memory allocation schemes for sequential as well as concurrent embedded software. Recently the NUS group has focused on linking timing analysis with model-driven software development flows, with the goal of model-level performance debugging.

-- Changes wrt Y2 deliverable --

No changes with respect to Year 2.

1.3 Starting Date, and Expected Ending Date

January 1st, 2008 until there is a framework for timing analysis of MPSoC systems.

-- Changes wrt Y2 deliverable --

No changes with respect to Year 2.

1.4 Policy Objective

The activity gathers the most prominent groups in the timing analysis area. They have all previously worked together in the ARTIST2 NoE, and therefore have well established links. The theme of the activity, timing analysis of MPSoC systems, is basically a new field scientifically, and also very timely from an application perspective as MPSoC and multicore architectures rapidly are becoming mainstream. A research effort in this area will thus establish European dominance in a field that rapidly is becoming very important. ArtistDesign also provides a close to perfect environment for this research due to the relevant competence in other activities and clusters, such as the compiler groups in the local cluster, and the MPSoC cluster.

-- Changes wrt Y2 deliverable --

No changes with respect to Year 2.

1.5 Background

All the partners in this activity have participated in the NoE Artist2. They developed a common tool architecture, and exchanged tool components. They have created the WCET Tool Challenge, executed in 2006 and 2008, to evaluate the existing commercial tools and academic prototypes. The Tool Challenge is planned to be executed every second year with improved conditions and more challenging benchmarks.

-- Changes wrt Y2 deliverable --

No changes with respect to Year 2.

1.6 Technical Description: Joint Research

Traditional static timing analysis has three parts: the flow analysis, which finds constraints on the possible program flows, the low-level analysis, which applies hardware timing models to obtain timing estimates for short execution paths, and the calculation which combines the result of the two previous analyses to obtain an estimate of the WCET for the full code.

Timing analysis on code level has so far dealt almost exclusively with sequential programs running in isolation. For MPSoC and multi-core architectures, these assumptions will no longer be valid: tasks might be parallel, and different tasks will run in parallel on different sets of cores. Timing analysis of parallel code, running on parallel hardware, is a new research area, and the aim of this activity is to initiate research in this area. Due to its novel nature, a large part of the research, at least for the first 18 months, will consist of initial investigations, paving the way for future in-depth research.

Some research problems are:

- Flow analysis needs to be extended from single-threaded programs to multi-threaded programs with possible synchronization between threads.
- Current low-level analysis is restricted to synchronous processor models: only [Thesing06] has modelled processor periphery. Hence, hardware modelling must be extended to include asynchronous systems, ultimately including full MPSoC and multicore architectures.
- Novel WCET calculation methods must be developed, which take into account that several interacting threads may have to complete in order for a task to complete.
- Methods to handle common resources must be devised. For single-processor systems, interference between tasks through shared resources like caches can be dealt with on the scheduling level, by bounding the number of preemptions and calculating a maximal timing penalty. For parallel processors, common resources can potentially be accessed at any time by totally unrelated activities. This renders traditional scheduling theory useless to estimate costs from interference with other tasks running in parallel.

Some ideas how to tackle the research problems are given below:

For flow analysis, there are several possibilities. One is to consider restricted parallel programming models, like Bulk Synchronous Programming, which have been developed in the parallel programming area in order to ease the task of parallel programming. These programming models have simple cost models, which should translate into more predictable timing models. Another possibility may be to use timing analysis to derive a Timed Automaton modelling the parallel code, and use the TA to analyze its synchronization properties. A third possibility is to use information from a parallelizing compiler. Such compilers sometimes use internal representations describing the computation in an abstract way, like an explicit task graph, or a polyhedral index set for sets of loop body executions, which is allocated and scheduled. The compiler then actually has considerable knowledge about where and when different computations are performed, which can be used to help predicting the timing.

For low-level analysis, the necessary hardware modelling should start with a formal specification of the architecture, and should be based on sound methods of abstraction, analysis, and transformation. The attainable accuracy of the models will be critically dependent on the hardware architecture: thus, research is necessary to find suitable MPSoC architectures which are amenable to timing analysis.

The calculation methods will depend on the program execution model. Thus, research to find appropriate such methods will be strongly connected to the flow analysis research.

The common resources issue is a matter of both hardware and system design. As for low-level analysis, research into MPSoC architecture and systems is necessary to reduce the interference between tasks. In particular on-chip networks and memories are crucial components, which have to be designed as to allow predictable timing. A hypothesis is that the ability to dynamically partition the resources, like assigning different parts of the network to different tasks, is helpful in this regard.

In the first 18 months, we foresee the following activities and potential results:

1. Derivation of timing models from MPSoC designs given in a language like Verilog or VHDL.
2. Meetings with researchers in Timed Automata (Modelling & Validation Cluster) to discuss the possible connections between timing analysis on code level and timing analysis on model level. Possible outcome: a report describing one or several combined approaches to the problem of analyzing parallel software with respect to timing properties such as WCET.
3. An investigation whether restricted models for parallel programming can make the problem of WCET analysis easier to solve for programs adhering to these models. Possible outcome: a survey of potentially interesting parallel programming models, with an assessment of their respective amenability to WCET analysis.
4. A joint activity with the MPSoC cluster, where TA expertise is fed back to MPSoC architecture level. Task: to identify features of MPSoC architectures that are critical to the predictability of timing properties, and to suggest possible designs which make the architectures more predictable with respect to these properties. Evident targets are shared resources like on-chip networks and shared memories. Possible outcome: a report describing the problem and some possible solutions, with their respective pros and cons.

-- Changes wrt Y2 deliverable --

No changes with respect to Year 2.

1.7 Work achieved in Year 1 *(Jan-Dec 2008)*

We studied the problem of timing predictability further. On the level of individual cores, work at Saarland University gave for the first time ever a precise and useful definition of predictability of cache replacement policies, competitiveness and sensitivity. This work can pave the ground for the analysis of cache predictability in the presence of multiple cores and of sensitivity to disturbances from outside influences.

Additionally, work on quantifying the influence of preemptive scheduling on cache contents allows to safely bound effects of preemption on WCET analysis. Furthermore, new work aims at guiding developers to select preemption points which minimise the preemption costs.

Work has started on the elaboration of design principles for architectures with predictable and composable timing behaviour. Various kinds of side effects, interferences that inhibit composability and impair predictability of single- and multi-processor systems have been identified. A small set of mechanisms that rely on pre-planned control to protect the time-relevant state from unforeseen changes has been proposed.

Work has been carried out in the area of measurement-based timing analysis. Such methods can be appropriate when timing behaviour is very complex and hard to model, such as for MPSoC systems, and when the real-time requirements are soft. Problems tackled include the adaptive control-flow segmentation to control the effort of test-case generation, and the use of learning techniques to identify timing model features.

Previous work in parametric WCET analysis has been continued. Parametric WCET analysis calculates a formula for the WCET, in some input parameters, rather than a single number. There are strong relations between the techniques used for this analysis and techniques used in parallelizing compilers, especially the polytope method.

-- No changes wrt Y2 deliverable --

This section was already presented in the Y2 deliverable, in section 1.7.

1.8 Work achieved in Year 2 *(Jan-Dec 2009)*

In Year 2, the partners continued to tackle the problem of poor timing predictability of current and emerging multicore and MPSoC architectures. This problem is complex, and the partners have pursued a number of different approaches to cope with it. The approaches can be classified as follows:

- Design principles for timing-predictable parallel architectures
- Static analysis for timing predictability
- Machine learning, model identification, and test case generation

The first approach includes work on timing-composable hardware/software acrhitectures, where constraints on the design of software and hardware yield systems for which a composable timing analysis of parallel tasks is possible. Furthermore, two technical meetings with the MPSoC design cluster have been held where the issue of timing-predictable MPSoC architectures has been investigated.

Static analysis for timing predictability includes work to better understand and predict the effect of hardware features such as caches. This includes the development of a method to analyze shared caches in multicore systems, a method to bound the context-switch penalties in preemptive systems, and improved microarchitectural analyses. The work on parametric

WCET analysis has also continued: recent advances include a method that improves considerably on the complexity of the analysis.

For current multicore/MPSoC systems, a safe static analysis can yield very pessimistic timing estimates. The third approach above is more pragmatic, and yields measurement-based or hybrid methods where models are built from observations. The obtained timing estimates are not safe in general, but the methods are often quite simple to use and tend to give timing estimates closer to the typical case. For applications where some overruns can be accepted, such methods can be appropriate. The partners have worked on methods to identify timing models from observed data, and on methods for test case generation.

Finally, the partners have continued to collaborate on common tool infrastructure and common formats for annotations and analysis results.

-- No changes wrt Y2 deliverable --

This section was already presented in the Y2 deliverable, in sections 1.8 and 3.1.

1.9 Problems Tackled in Year 3 *(Jan-Dec 2010)*

Within the PREDATOR project, we continued the advances in various fields. The work on foundations of predictability has found success in the proposal of an adaptable template for predictability definitions. We also continued the work of timing analysis of cache replacement strategies, tackling FIFO and PLRU replacement strategies.

Timing analysis of tasks under preemptive scheduling was also investigated further. A new approach based on resilience further improved the bounds on cache-related pre-emption delay context-switch costs for preempted tasks.

Experiments have shown the cache-aware memory allocator to work successfully in practice, and our algorithm for transforming dynamic into static allocation has been extended as well. Both approaches allow for timing analysis of tasks with dynamic memory allocation.

Concerning the goal of reconciling timing analysis with compilation, the WCET-aware compiler WCC developed at TU Dortmund has matured in the past 12 months. WCC is currently evaluated at Bosch and is now able to generate and optimize industrial code representing an engine control system. First results indicate that WCC produces code with a WCET of a factor of 2 below that of the GCC compiler. In ArtistDesign year 3, WCC has been extended towards code generation and optimization for multi-process systems. This included porting a real-time operating system to the TriCore architecture mainly supported by WCC. In ArtistDesign year 3, the collaboration between TU Dortmund, Saarbrücken, AbsInt, ETHZ and Pisa has been strengthened significantly. Furthermore, cooperation between TU Dortmund and NUS Singapore led to a joint effort for bus-aware timing analysis for multi-core systems.

TU Dortmund's work on WCET-aware compilation has been awarded several times during ArtistDesign year 3. Timon Kelters master's thesis on WCET-aware superblock formation and optimization has been awarded three times. It received the Hans-Uhde award and was awarded as best thesis in the field of real-time systems by the German Society of Informatics (GI) and as nationally best computer science thesis by Germany's congregation of computer science faculties. Furthermore, Dortmund's former group member Paul Lokuciejewski received TU Dortmund's dissertation award.

The work on parametric WCET analysis has continued. A main problem to be tackled is the complexity of the analysis, and how to find the best tradeoffs between efficiency and precision. During Year 3 the partners developed new, more efficient methods for the symbolic calculation

phase, as well as for symbolic path (program flow) analysis. The use of parametric WCET estimates to obtain a more precise schedulability analysis has also been studied.

The problem of analyzing explicitly parallel programs with respect to execution time has been the focus of an initial study. Such programs cannot be analyzed using methods and models for purely sequential tasks. Real-time scheduling theory for parallel systems is not applicable either, since parallel programs may synchronize in a much more general manner than assumed by this theory.

Several partners have worked on the issue of timing predictability. Besides the work on timing analysis of cache replacement strategies mentioned above, we have studied how to use code transformations to avoid the unpredictability caused by timing anomalies. We have also introduced the concept of timing barrier. The work on coding policies for timing predictability has been forwarded by extending the allowed coding policy to partially input data-dependent programs. The results indicate that the more relaxed policy still allows a fully automated loop bounds analysis. Finally the problem of mapping source-level program flow annotations to binary level, in the presence of code optimizations, has been attacked by developing an approach where the annotations are transformed in parallel with the code.

The problem of early WCET analysis has been attacked. Early WCET approximations are useful to direct the system design work in order to avoid costly system redesigns at late stage due to timing and resource bottlenecks. A tool chain combining a fast and approximate WCET analysis with a source-level program flow analysis has been implemented.

The work on hybrid- and measurement-based approaches has been continued. An important part is test-case generation, where advances have been made regarding suitable coverage criteria as well as for search-based approaches to find representative trace samples. We have also applied learning techniques to approximate cache modeling from test data.

-- The above is new material, not present in the Y2 deliverable --

2. Summary of Activity Progress in Year 3 *(Jan-Dec 2010)*

2.1 Technical Achievements

Timing Analysis and Timing Predictability (USaar, TU Vienna)
We have continued the work on cache replacement analysis: First, we devised a pair of precise and efficient must- and may-analyses for FIFO replacement, based on the novel principle of static phase detection. These analyses are more precise than the first ones for FIFO, which were reported on last year. Second, we addressed PLRU replacement, for which the precision of analyses still lags behind. We devised a more precise, but still expensive, must-analysis that can predict more cache hits than state-of-the art analyses.

We also continued the work on predictability. Some architectural features make timing analysis very hard, if not infeasible: we have shown how smart configurations of existing complex architectures can alleviate this problem. Furthermore, we resumed efforts towards a formal definition of predictability, arriving at definitions distinguishing between several kinds of predictability with respect to several sources of uncertainty.

We further elaborated on the two techniques developed in 2009 to handle the additional uncertainty of cache effects arising from dynamic memory allocation. Tests with our predictable memory allocator confirm that in practice cache-awareness does not further increase worst-case execution times or fragmentation of real-time dynamic memory (de)allocation. Our algorithm for transforming dynamic allocation into static allocation with comparable memory consumption was extended to cope with programs for which only parametrical bounds on the number of allocated objects are statically available. The work on derivation on infeasibility constraints from Simulink models and mode-specific WCETs also continued. We derived procedures to couple the analyses to improve the precision of timing bounds even further.

We continued the discussion on the meaning and possible formalizations of the term time-predictability. The different uses of the term in literature have been reviewed and possible valid interpretations of the term have been analyzed – predictability as a statement about the phenomenon execution time vs. predictability as a property of the process of execution-time modeling. It was concluded that a definition of time predictability should cover both aspects, stability and analyzability of a timing model. A formalization of this model was presented. Further, the concept of a timing barrier, a mechanism of an architecture that limits the propagation of timing effects, was introduced. Providing timing barriers should help to build real-time computer systems with less complex timing dependencies, whose timing is thus easier to understand.

Parametric WCET Analysis (MDH, USaar)

We have proved termination, and upper bounds on recursion depth, of an efficient algorithm for symbolic WCET calculation, which was developed earlier. Previous parametric calculation methods have scaled very poorly, and the new algorithm improves significantly on this. An implementation in the WCET analysis tool SWEET of a full parametric WCET analysis method using this algorithm is well underway, and a first working version is expected early 2011.

Another performance bottleneck of timing analyses, numeric and especially parametric, is the so-called path analysis, which determines the path in the analyzed task with the longest execution time bound. We present a new approach to path analysis which exploits the often rather regular structure of software for hard real-time and safety-critical systems. We strongly improve upon former techniques in terms of precision and runtime in the parametric case.

Even in the numeric case, the approach competes with state-of-the-art techniques and may be an alternative to commercial tools employed for path analysis.

WCET Analysis in the Presence of Context Switches (USaar, AbsInt, SSSA)

In preemptive real-time systems, scheduling analyses need — in addition to the worst-case execution time — the context-switch costs. The delay caused by cache misses due to context switches, which constitutes the major part of the context-switch costs, is referred to as *cache-related preemption delay* (CRPD). We developed a new approach to compute tight bounds on the CRPD for LRU set-associative caches, based on analyses of both the preempted and the preempting task. As the basis of our approach we introduce the notion of *resilience* of a memory block of a preempted task: the maximal number of memory accesses a preempting task could perform without causing an additional miss to this block. By computing lower bounds on the resilience of blocks and an upper bound on the number of accesses by a preempting task, one can guarantee that some blocks may not contribute to the CRPD. This resilience-based CRPD analysis considerably outperforms previous approaches. Ongoing work at Absint aims at integrating the resilience analysis in the timing analysis tool aiT.

Abstraction of VHDL Processor Models (USaar)

We have continued our work on deriving processor timing models out of formal VHDL specifications semi-automatically. Derivation is a two-step process: A model-preprocessing step removes code elements that have no influence on the timing behavior. A second step introduces processor state abstractions that approximate parts of the system.

In order to automate this derivation process, we have developed several abstract interpretation based static analyzes and transformation tools. The analyzes consist of a slicing tool for interactive exploration and understanding of the VHDL model, a reset analysis to identify unused parts of the model due to a given processor configuration and an assumption-based model refiner for identifying so-called timing-dead code based on input assumptions. The transformation tools are a timing-dead code eliminator that prunes the model by parts marked as timing dead, a domain abstractor for abstracting from a concrete domain to an abstract one and a process replacer with which whole processes can be replaced by custom simulation routines.

Exploitation of timing analysis during compilation (TU Dortmund, AbsInt)

The work on Dortmund's Worst-Case Execution Time-aware C Compiler WCC has been continued in ArtistDesign Year 3.

We exploited the concept of superblocks (regions in a program code that consist of multiple basic blocks) for the optimization of real-time systems. Superblock formation is now based on a novel trace selection algorithm, which is driven by WCET data. Moreover, superblocks are translated for the first time from assembly to source code level. An adaption of the traditional optimizations common subexpression and dead code elimination to the proposed WCET-aware superblocks allows an effective WCET reduction [LKM10]. This activity was awarded three times, e.g. as best computer science thesis in Germany.

TU Dortmund developed techniques for WCET-aware basic block reordering in order to avoid unconditional branches and to support prediction of conditional branches. A genetic approach determines an optimized order of basic blocks of a function by applying evolutionary algorithms considering the WCET of the program to optimize as fitness value. Additionally, an integer-linear programming-based approach determines the optimal order of basic blocks and also takes the branch prediction into account.

A generic framework was developed by TU Dormtund integrating machine learning into a compiler to enable an automatic search for the best learning algorithm. To find good settings for the learner parameters within the large search space, optimizations based on evolutionary algorithms are applied. In contrast to the majority of other approaches aiming at a reduction of the average-case execution time (ACET), the goal of this activity is the minimization of the worst-case execution time (WCET). Based on the proposed machine-learning framework, the well-known optimization loop-invariant code motion has been made WCET-aware [LSMM10].

http://ls12-www.cs.tu-dortmund.de/research/activities/wcc

WCET-aware memory allocation techniques for single-task systems (TU Dortmund)
TU Dortmund proposed a new WCET-driven cache-aware memory content selection algorithm, which allocates functions whose WCET highly benefits from a cached execution to cached memory areas. Vice versa, rarely used functions which do not benefit from a cached execution are allocated to non-cached memory areas [PLM10].

Register allocation is considered the most important compiler optimization at all. TU Dortmund developed a novel WCET-aware register allocator based on integer-linear programming (ILP). The proposed ILP formulation relies on an already known approaches for ILP-based register allocation minimizing spill code. Significant effort has been spent in order to model the WCET and its heavily dynamic nature w.r.t. the worst-case execution path (WCEP) inside the ILP. Furthermore, care has been taken in order to model the impact of individual spill instructions generated by the register allocator on the processor's pipeline, as these additional instructions might contribute to the WCETs of their belonging basic blocks and thus might impact a program's WCEP and WCET.

http://ls12-www.cs.tu-dortmund.de/research/activities/wcc

Computation of Pareto-optimal optimization sequences (TU Dortmund, ETHZ)
Due to the complex interactions between a compiler's optimizations, the choice for a promising sequence of code transformations is not trivial. Compiler developers address this problem by proposing standard optimization levels, e.g., O3 or Os. However, previous studies have shown that these standard levels often miss optimization potential or might even result in performance degradation. In this activity, TU Dortmund and ETH Zurich propose the first adaptive WCET-aware compiler framework for an automatic search of compiler optimization sequences which yield highly optimized code. Besides the objective functions ACET and code size, we consider the WCET. To find suitable trade-offs between these objectives, stochastic evolutionary multi-objective algorithms identifying Pareto optimal solutions are exploited [LPFM+10a, LPFM+10b].

http://ls12-www.cs.tu-dortmund.de/research/activities/wcc

Cache analysis framework and ERIKA-port to TriCore architecture for multi-task systems (TU Dortmund, SSSA, Evidence SRL)
Since the WCET-analyzer integrated into the WCC is aimed at determining the WCET of isolated tasks, its application in scenarios with multiple tasks being scheduled on a single processor is limited. This is primarily due to the mutually invisible effects on the hardware states which largely affect each task's WCET. To perform safe estimations of the respective WCET of each task, particularly detailed knowledge of the cache states throughout the schedule is mandatory. Therefore, a framework for cache analysis for one or multiple tasks has been developed which is portable, modular, fast, precise and features state-of-the-art analyses for cache-related effects on the tasks' WCET. In the future, this framework will be the basis for scheduling- and operating-system-aware WCET optimizations integrated into WCC.

To support the ongoing efforts at TU Dortmund to feature WCET-aware multi-task optimizations, the ERIKA operating system provided by SSSA/Evidence has been ported to the TriCore platform which plays an important role in today's automotive industry. To meet the industry's demand for highly time-predictable software, the WCC compiler not only supports automatic WCET-aware optimizations for single, isolated tasks but is also increasingly adopted to support these kinds of optimizations in the context of multiple tasks scheduled on a single processor. The TriCore-specific additions to ERIKA are officially packaged with an entirely open development platform provided by Evidence SRL.

http://www.evidence.eu.com/

Bus-aware timing analysis for multicore processors (TU Dortmund, NUS)

For single-core systems, industrial-strength WCET analyzers are already available, but up to now, only first proposals have been made to analyze the WCET in multicore systems, where the different cores may interfere during the access to shared resources. An important example for this are shared buses which connect the cores to a shared main memory. The time to gain access to the shared bus may vary significantly, depending on the used bus arbitration protocol and the access timings. In this activity, TU Dortmund and NUS propose a new technique for analyzing the duration of accesses to shared buses. A prototype tool is implemented which uses the new analysis and is tested on a total of 26 realworld benchmarks. Results demonstrate that this novel analysis achieves the same precision as the best existing approach while drastically outperforming it in matters of analysis time [KFMC+10].
http://www.comp.nus.edu.sg/~rpembed/chronos

Using Learning to Support the Development of Embedded Systems (York, MDH)
The University of York have been engaged in an EPSRC funded project "Using Machine Learning to Support the Development of Embedded Systems", ref: EP/F00334X, since 2007. This supports a post-doctoral researcher and a PhD student. The main thrust of the post-doctoral research is to use Bayesian Networks to learn a cache model for the execution of the software on the given platform. This model can then be directly used in a probabilistic form of optimisation, e.g. an IPET formulation dealing with probabilities, or the worst-case cache state determined using a technique such as Monte Carlo Optimisation.

In addition we have been investigating how a multi-criteria optimisation approach to the generation of test data can help find the WCET using search and looking at how to use search-based approaches to choose representative samples of large input traces. This last item is particularly important as many streams have very large input spaces, i.e. the range of data that can be streamed into them makes traditional approaches to dynamic analysis at best give low coverage of the possible options if no guidance is provided. In this work search-based techniques are used to determine which parts of a larger input stream set is representative of the larger set. In effect we can then separate the WCET analysis problem into dealing with the impact of the input data from dealing with the way in which the software executes.

In recent work, we have cooperated with Mälardalen University to consider how parametric WCETs can be derived using the TIMES model checker: a joint paper has appeared in 2010. This work has also spawned other research looking at how systems cope with change and how certain changes within the execution times of software have a wider impact on the response times of software.

http://www-users.cs.york.ac.uk/~bartlett/rts/

WCET analysis for explicitly parallel programs (MDH)
A study has been made how to use model checking of timed automata to do WCET analysis for tasks that are explicitly parallel, and consist of different threads running in parallel. A multicore architecture was modeled using UPPAAL's modeling language, together with a parallel program running on the architecture. The study showed that it indeed is possible to perform WCET analysis for such programs in this manner, and that the modelling was easy and convenient using the modelling capabilities of UPPAAL, but it also revealed that the approach has scalability problems.

Early WCET analysis (MDH, TU Vienna, AbsInt)
WCET analysis is typically applied late in the development process, when verifying the timing properties. There is a need to estimate timings earlier in the development process, to support the time budgeting, and the dimensioning of the hardware. We have investigated an approach based on fast and approximate microarchitectural analysis, combined with source-level program flow analysis, for this purpose. A version of aiT (AbsInt) has been created that implements a fast microarchitectural analysis. The tools SWEET (MDH) and SATiRE (TU Vienna) have been given backends to generate program flow constraints that aiT can read. The result is a tool chain that can perform quick design space exploration with a high level of automation.

Avoiding Timing Anomalies by Code Transformations (TU Vienna)
Computer architectures that show timing anomalies impede the construction of simple, composable timing models for WCET analysis. As part of our work we started to explore software-based strategies for avoiding timing anomalies on state-of-the-art processor architectures. In an effort to eliminate the timing anomalies originating from the processor's out-of-order instruction pipeline, we conceived different methods of inserting instructions in the program code that render the dynamic instruction scheduler of the CPU inoperative. Strategies of inserting NOP or XOR instructions that limit the usable fetch width or the usable prefetch window size or introduce new artificial dependencies between CPU registers were implemented and compared wrt. their effects on code-size increase and run-time overhead. Experiments showed that timing anomalies can be eliminated by software measures; those are however processor specific and come at a high cost.

Code Optimization and Timing Analysis (TU Vienna)
WCET analysis needs information about possible execution paths or control flows, which may be provided manually by the user or calculated automatically by program analysis. For flexibility and simplicity reasons it is desirable to specify the flow information at the same level at which the program is developed, i.e., at the source level. Also, automatic control-flow analysis is best done at a higher-level representation, where most semantic information about the code is available. In contrast, the machine-specific analysis and the final steps of WCET computation have to be performed at machine-code level. Mapping and transforming the flow information from the source-level down to the machine code, where flow information is used in the WCET analysis, is challenging, even more so if the compiler generates highly optimized code. We therefore developed a method for transforming flow information from source code to machine code. To obtain a mapping that is safe and accurate, flow information is transformed in parallel to code transformations performed by an optimizing compiler. This mapping is not only useful for transforming manual code annotations but also if platform-independent flow information is automatically calculated at the source level. It is shown that the proposed method has no limits with respect to the flow-fact transformations that can be modeled.

Coding Policies for Automated Path Analysis (TU Vienna, MDH)
In order to make code WCET-analyzable, programmers are often required to provide additional information on the program flow, analyzing subtle, context dependent loop bounds manually. To avoid this tedious and error-prone step, we propose to restrict written and generated code to the class of programs with input-data independent loop counters. The proposed policy builds on the ideas of single-path code, but is less restrictive in that it requires partial input-data independence. The effect of using this coding policy is that it makes it possible to find precise loop bounds for these programs, by using an efficient variant of abstract execution. The systematic construction of tasks following the policy is facilitated by embedding knowledge on inputdata dependence in function interfaces and types. We have tested this proposal for a coding strategy with a number of algorithms and benchmarks. These initial experiments showed that this restriction is indeed a good candidate for removing the need for manual annotations. We have started to discuss the implications of the coding policies on WCET analysis strategies and tools, e.g., SWEET. For example, exploring the impact of the coding policies on the scalability of flow analysis or parametric analysis could be fruitful research issues.

Test-Case Generation for WCET Analysis (TU Vienna)
Within this activity, we explore the efficient coverage of program models for measurement-based WCET analysis. We continued the investigation of segmentation strategies and coverage metrics for measurement-based timing analysis. An evaluation of coverage metrics from functional testing showed that those metrics do not suffice for WCET analysis, as they do not sufficiently reflect the effects of execution paths (the execution history) on possible cache or pipeline states and thus on context specific execution times of basic blocks. To be able to model dependencies between blocks and yet keep the number of test cases manageable, it was decided to explore a test-data generation that achieves full pair coverage (i.e., generate test suites that include at least one test vector for each feasible *pair of nodes* of the control flow graph). We conducted first experiments in which pair coverage yielded better results than the classical test-coverage metrics.

-- The above is new material, not present in the Y2 deliverable --

2.2 Individual Publications Resulting from these Achievements

USAAR

M. Schlickling and M. Pister. *Semi-Automatic Derivation of Timing Models for WCET Analysis*. Proceedings of the ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems (LCTES), 2010.

S. Altmeyer, C. Maiza, and J. Reineke. *Resilience Analysis: Tightening the CRPD Bound for Set-Associative Caches*. Proceedings of the ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems (LCTES), 2010.

D. Grund, and J. Reineke. *Precise and Efficient FIFO-Replacement Analysis Based on Static Phase Detection*. Proceedings of the 22nd Euromicro Conference on Real-Time Systems (ECRTS), 2010.

D. Grund, and J. Reineke. *Toward Precise PLRU Cache Analyis.* Proceedings of 10th International Workshop on Worst-Case Execution Time (WCET) Analysis, 2010.

R. Wilhelm, Ph. Lucas, O. Parshin, L. Tan, and B. Wachter. *Improving the Precision of WCET Analysis by Input Constraints and Model-derived Flow Constraints.* In *Advances in Real-Time Systems,* LNCS, 2010. To appear.

E. Althaus, S. Altmeyer, and R. Naujoks. *Precise and Efficient Parametric Path Analysis.* Proceedings of the ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems (LCTES), 2011. To appear.

TU Dortmund

[LSMM10] P. Lokuciejewski, M. Stolpe, K. Morik and P. Marwedel. *Automatic Selection of Machine Learning Models for WCET-aware Compiler Heuristic Generation*. In Proceedings of SMART '10: 4th Workshop on Statistical and Machine Learning Approaches to Architectures and Compilation, January 2010, pp. 3-17.

[PLM10] S. Plazar, P. Lokuciejewski and P. Marwedel. *WCET-driven Cache-aware Memory Content Selection*. In Proceedings of ISORC '10: 13th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing, May 2010, pp. 107-114.

[LKM10] P. Lokuciejewski, T. Kelter and P. Marwedel. *Superblock-Based Source Code Optimizations for WCET Reduction*. In Proceedings of ICESS '10: 7th International Conference on Embedded Software and Systems, June 2010, pp. 1918-1925.

[FaLo10] H. Falk and P. Lokuciejewski. *A compiler framework for the reduction of worst-case execution times*. In The International Journal of Time-Critical Computing Systems (Real-Time Systems), 46(2):251-300, Springer, October 2010.

[LoMa10] P. Lokuciejewski and P. Marwedel. *Worst-Case Execution Time Aware Compilation Techniques for Real-Time Systems*. Springer, November 2010.

NUS

Bach Khoa Huynh, Lei Ju and Abhik Roychoudhury. Scope-aware Data Cache Analysis for WCET Estimation. *Proc. 17th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS)*, 2011.

Lei Ju, Bach Khoa Huynh, Abhik Roychoudhury and Samarjit Chakraborty. Timing Analysis of Esterel Programs on General-purpose Multiprocessors. *Proc. ACM Design Automation Conference (DAC)*, 2010.

Sudipta Chattopadhyay, Abhik Roychoudhury and Tulika Mitra. Modeling Shared Cache and Bus in Multi-cores for Timing Analysis. *Proc. 13th International Workshop on Software and Compilers for Embedded Systems (SCOPES)*, 2010.

York

M. Bartlett, I. Bate, and J. Cussens. Instruction Cache Prediction Using Bayesian Networks. In *Proc. 19th European Conference on Artificial Intelligence (ECAI 2010)*, pp. 1099--1100, Lisbon, Portugal, Aug. 2010.

M. Bartlett, I. Bate, and J. Cussens. Learning Bayesian Networks for Improved Instruction Cache Analysis. In *Proc. 9th International Conference on Machine Learning and Applications (ICMLA'10)*, 2010.

MDH

Stefan Bygde, Andreas Ermedahl, and Björn Lisper. An Efficient Algorithm for Parametric WCET Calculation. *Journal of Systems Architecture*, in press, July, 2010.

Andreas Gustavsson, Andreas Ermedahl, Björn Lisper, and Paul Pettersson. Towards WCET Analysis of Multicore Architectures using UPPAAL. *Proc. 10th International Workshop on Worst-Case Execution Time Analysis*, pp. 103-113, Bruxelles, July, 2010.

TU Vienna

Raimund Kirner and Peter Puschner. *Time-Predictable Computing*. In Proc. 8th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, p. 23-34, 2010.

Benedikt Huber and Peter Puschner. *A Code Policy Guaranteeing Fully Automated Path Analysis*. In Proc. 10th Euromicro Workshop on WCET Analysis, p. 80-90, 2010.

Raimund Kirner, Peter Puschner, and Adrian Prantl. *Transforming Flow Information during Code Optimization for Timing Analysis*. Journal of Real-Time Systems, 45(1-2), p. 72-105, 2010.

Albrecht Kadlec, Raimund Kirner, and Peter Puschner. *Avoiding Timing Anomalies Using Code Transformations*. In Proc. 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, p. 123-132, 2010.

-- The above are new references, not present in the Y2 deliverable --

2.3 Interaction and Building Excellence between Partners

Interaction between Dortmund, Pisa, ETHZ and NUS

Strong interaction between Dortmund, Pisa, ETHZ and NUS in the context of the WCET-aware compiler WCC has been achieved during ArtistDesign year 3. Dortmund and NUS developed an improved bus-aware timing analysis for multi-core systems. An intensive cooperation with ArtistDesign's operating systems cluster has been set up, in particular with the cluster leader and core member SSSA-Pisa and with the affiliated member Evidence. Furthermore, Zurich's tool box of evolutionary algorithms has been integrated successfully into the WCC compiler. All these integration activities led to an official port of the ERIKA operating system which is publicly available on Evicende's websites, and to joint publications.

Interaction between MDH, AbsInt, and TU Vienna

AbsInt, Mälardalen, and Vienna have continued their cooperation on methods for early stage timing analysis. The tools SWEET, and SATiRE, can now export source code flow analysis results to the commercial tool aiT. See further Section 2.1.

Interaction between MDH, and York

MDH and York have cooperated on the use of parametric WCET estimates for calculating more precise response time bounds.

Interaction between TU Vienna, and MDH

TU Vienna and MDH investigate the connections and cooperation between the flow-fact

analysis and its realization in the SWEET tool (focus at MDH) and the work on writing time-predictable software (focus at TU Vienna).

WCET Challenge

The WCET Challenge is a recurring event where timing analysis tools are jointly evaluated using some predefined benchmark codes. There are multiple evaluation criteria like the precision of the analysis, the level of automation, and the usability of the tool. Thus, the Challenge is not primarily a contest, but rather an event for showcasing tools and promoting the area of timing analysis. This provides visibility of the tools as well as valuable information for potential users.

The third WCET Challenge is now being arranged with support from ArtistDesign. This year eight tools have entered, including both commercial and academic tools from both ArtistDesign partners and tool providers outside the NoE. There are two classes: one "industrial strength" class where the tools are evaluated on real industrial automotive code, and one "open" class where simpler benchmark codes are used. As of the date of writing, the evaluation is planned to take place in March 2011 and the results will be presented in the summer of 2011.

http://www.mrtc.mdh.se/projects/WCC/

-- Changes wrt Y2 deliverable --

The interaction between Dortmund, Pisa, ETHZ and NUS, between MDH and York, and the WCET Challenge, are novel to the Y3 deliverable.

2.4 Joint Publications Resulting from these Achievements

C. Cullmann, C. Ferdinand, G. Gebhard, D. Grund, C. Maiza, J. Reineke, B. Triquet, and R. Wilhelm. *Predictability Considerations in the Design of Multi-Core Embedded Systems*. *Proceedings of Embedded Real Time Software and* Systems (ERTSS), 2010.

[LPFM+10a] P. Lokuciejewski, S. Plazar, H. Falk, P. Marwedel and L. Thiele. *Multi-Objective Exploration of Compiler Optimizations for Real-Time Systems*. In Proceedings of ISORC '10: 13th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing, May 2010, pp. 115-122.

[LPFM+10b] P. Lokuciejewski, S. Plazar, H. Falk, P. Marwedel and L. Thiele. *Approximating Pareto Optimal Compiler Optimization Sequences - a Trade-Off between WCET, ACET and Code Size*. Submitted to: Software - Practice and Experience, 2010 (Current Status: Accepted after minor revision).

[KFMC+10] T. Kelter, H. Falk, P. Marwedel, S. Chattopadhyay and A. Roychoudhury. *Bus-Aware Multicore WCET Analysis through TDMA Offset Bounds*. Submitted to: The 23rd Euromicro Conference on Real-Time Systems (ECRTS), 2011.

Y. Lu, T. Nolte, I. Bate, and C. Norström. Timing Analyzing for Systems with Execution Dependencies between Tasks. *Proc. 25th ACM Symposium on Applied Computing (SAC2010)*, pp. 357-358, 2010.

Björn Lisper, Andreas Ermedahl, Dietmar Schreiner, Jens Knoop, and Peter Gliwa. Practical Experiences of Applying Source-Level WCET Flow Analysis on Industrial Code. In Tiziana Margaria and Bernhard Steffen, eds. *Proc. 4th$ International Symposium on Leveraging Applications of Formal Methods (ISOLA'10)*, pp. 449-463, Heraclion, Crete, October, 2010.

-- The above are new references, not present in the Y2 deliverable --

2.5 Keynotes, Workshops, Tutorials

Invited Talk: *Peter Marwedel, Heiko Falk:* **Reconciling compilers and timing analysis**
Industrial Workshop and Exhibition at the CPSWEEK 2010
Stockholm, Sweden – April 12, 2010

Most embedded/cyber-physical systems have to respect timing constraints. Ensuring meeting such constraints is currently typically based on a trial-and-error procedure involving many time-consuming software generation attempts. In this talk, we will demonstrate how the integration of timing analysis into a compiler for an automotive processor can provide a systematic path toward optimized worst-case execution times and can cut down costs.

http://www.mrtc.mdh.se/CPSweek/industrialWS/

Workshop : 10th Int'l Workshop on Worst-Case Execution Time Analysis (WCET'10)
Brussels, Belgium – July 6th, 2010
On July 6th, 2010, thirty-five people met in Brussels, Belgium, to hold the 10h International Workshop on Worst-Case Execution Time Analysis (WCET'10, http://www.artist-embedded.org/artist/WCET-2010.html). The workshop was organised as a satellite event of the 22nd Euromicro Conference on Real-Time Systems (ECRTS'10, http://ecrts.eit.uni-kl.de/ecrts10). ArtistDesign supported the workshop by paying the travel costs of the invited speaker, Dr. Jean Souyris (Airbus), as well as the costs of printing and distributing the proceedings and putting them online). ArtistDesign participants presented several of the workshop papers.

Workshop : 8th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS 2010)
Waidhofen/Ybbs, Austria, October 13-15, 2010

The 8th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems was held in Waidhofen an der Ybbs, Austria. Thirty-four researchers and practitioners met to present and discuss novel contributions to the state-of-the-art and the state-of-practice in the field of embedded and ubiquitous computing systems. The workshop program included a number of presentations by representatives from ArtistDesign partners.
http://pan.vmars.tuwien.ac.at/seus2010/index.html

Tutorial: *Heiko Falk, Peter Marwedel:* **Reconciling Compilers and Timing Analysis for Safety-Critical Real-Time Systems – the WCET-aware C Compiler WCC**
International Symposium on Code Generation and Optimization (CGO 2011)
Chamonix, France, April 03, 2011

Timing constraints must be respected for safety-critical real-time applications. Traditionally, compilers are unable to use precise estimates of execution times for optimization, and timing properties of code are derived after compilation. A number of design iterations are required if timing constraints are not met. We propose to reconcile compilers and timing analysis and to create a worst-case execution time (WCET) aware compiler in this way. Such WCET-aware compilers can exploit precise WCET information during compilation. This way, they are able to improve the code quality. Also, we may be able to avoid some of the design iterations.

In this tutorial, we present the integration of a compiler and a WCET analyzer, yielding our WCET-aware compiler WCC. We are then considering compiler optimizations for their potential to reduce the WCET, assuming that the WCET is now used as the cost function. Considered

optimizations include loop unrolling, register allocation, scratchpad memory allocation, memory content selection and cache partitioning for multi-task systems. For large sets of benchmarks, average WCET reductions of up to 40% were achieved. The results indicate that this new area of research has the potential of achieving worthwhile execution time reductions for safety-critical real-time code.
http://www.cgo.org/cgo2011/tutorials.php

-- The above is new material, not present in the Y2 deliverable --

3. Milestones, and Future Evolution

3.1 Problems to be Tackled in Year 4 *(Jan 2011 – Dec 2011)*

Even though the PREDATOR project lasts only until January 2011, the work on foundations of predictability and design of predictable systems will continue. In the next 12 months, we strive to clarify further the notions of predictability and to experimentally validate or falsify traditional beliefs about the performance enhancements effects of hard-to-predict hardware features. The work on timing analysis of preemptive systems will also continue, including studying the influence of the task model on the precision of scheduling analysis for preemptive systems

In the future, predictable code generation using the WCET-aware C Compiler WCC will be extended towards multi-task and multi-core systems. For multi-task systems, it is crucial that the compiler is aware of accesses to resources shared by the different tasks. Furthermore, detailed knowledge about the underlying operating system must be available inside the compiler. For multi-core systems, the compiler has to be aware of accesses to resources shared among the individual cores, like e.g. shared busses or memories. It is envisioned that WCET-aware compilation reduces accesses to resources shared within a single core and shared between different cores in order to improve a system's worst-case timing. Additionally, the compiler will be able to tailor the underlying operating system, in particular its scheduling policy and its scheduling parameters.

Current methods for deriving program flow constraints to be used in WCET analysis have the limitation that numbers are considered unbound. In reality, they have representations of bounded size. This can affect the semantics, especially for small, embedded processors with small wordlength. A sound program flow analysis must take this into account. Our previously developed parametric WCET analysis relies on relational abstract domains. We will tackle the problem how to make such domains bit-precise, i.e., that the operations on these domains become proper abstractions of finite size arithmetics.

As regards WCET analysis of explicitly parallel programs, we will investigate alternatives to the model checking approach in order to overcome the inherent complexity problems. We believe that a viable approach is to identify segments of the code that can be analyzed by conventional WCET analysis methods, assuming sequential execution, and analyze the timing behaviour pertaining to synchronization by model checking or other means.

We will continue to investigate the connections and cooperation between the flow-fact analysis and its realization in the SWEET tool and the work on writing time-predictable software. For example, SWEET could be used to automatically derive flow facts for certain classes of programs of interest, e.g., single-path programs or programs with input-data independent loop counters. In this context it is planned to explore the relationship between these classes of programs and the different merging strategies that SWEET needs to use to analyze these codes.

-- Changes wrt Y2 deliverable --

The objectives have evolved to include predictable code generation for multicore/MPSoC systems.

3.2 Current and Future Milestones

- **PROMPT design principles**

  1. The PROMPT design principles for predictable architectures were developed, based on the principles of de-sharing.
  2. The design principles are based on some intuitive assumptions derived from experience with developing timing analyzers for several architectures. These intuitive assumptions were formally defined and proved for the case of cache architectures. For other architectural components we still lack the underlying theory.
  3. The design principles will be applied to the design of the PREDATOR multi-core architecture, resulting in the first version of this design.

     100% achieved. The design principles have been developed.

- **Timing Side Effects**
  A clean design and analysis process of embedded systems timing is strongly simplified if the temporal planning and the timing analysis can be performed in a hierarchical way, e.g., in a separation of WCET and schedulability analysis, of single-core analysis from the analysis of the whole multi-core system. Unfortunately many features found in contemporary hardware and software cause phenomena that inhibit a hierarchical decomposition of the planning and analysis process. It is our goal to investigate these phenomena and name the unwanted side effects they have on timing. The results will be used to work out alternative hardware and software structures that provide better support to a hierarchical design and analysis of worst-case timing.
  100% achieved. We have identified various types of side effects, ranging from data-dependent instruction execution times and data paths, over processor-state dependencies between different invocations of a single task, to the side effects that the execution of a task has on the timing of other tasks, where other tasks may reside on the same processor (in a single or multi-processor system) or a different processor of a multi-processor system. The findings of this investigation have been published and presented at the Workshop on Worst-Case Execution-Time Analysis, WCET2008.

- **Composable Software/Hardware Architectures**
  Temporal composability is essential for making the factor time an integral part of a meaningful hierarchical development process for embedded real-time applications. Our goal is to work out software structures, hardware mechanisms, and appropriate strategies for the use and control of these hardware mechanisms that support temporal composability.
  75% achieved. In year 1, we started towards this goal, and some ideas on how to obtain composability have been published. In year 2, the topic has been further explored. We investigated the prerequisites for a hierarchical software construction process that provides temporal composability in more detail. A simple example of an application was implemented to show that the use of adequate hardware and software models can provide the framework that avoids temporal interferences between tasks and supports temporal composability.
  In year 3 we continued to investigate into the phenomenon of time predictability and composability and worked out alternatives to implement time-predictable memory hierarchies that would support temporal composability.

  Year 4: Concretization of ideas for memory hierarchies. First implementation of a hierarchical memory architecture that supports the temporal predictability and composability.

- **Initial Timing Model for Parallel Programs**
  We plan to define a formal timing model for explicitly parallel programs. This model will serve as a "reference point", and the correctness of forthcoming timing analyses for such programs will be possible to verify with respect to this model.
  25% achieved. This activity did not get started in Year 2 as planned, due to a delay in the recruitment of a PhD student to work on the problem. In Year 3, progress was made towards a timing model using Timed Automata.
  Year 4: A formal semantics for parallel programs that includes timing.

- **Measurement-based WCET analysis**
  Generating the right test data to get a good coverage of relevant test cases is a core issue in measurement-based WCET analysis for complex computer architectures. We will explore methods for generating test cases that provide a good coverage by investigating methods for partitioning and analyzing control-flow graphs.
  50% achieved. In Year 2 we investigated the segmentation of control-flow graphs to reduce the effort for test generation and measurements for the WCET analysis. In Year 3, the further investigation into segmentation techniques made us aware that there is a lack of adequate coverage metrics for the generation of test data for the execution-time measurements – we found that the traditional metrics used for functional testing are not sufficient, as they do not allow us to model timing effects between different blocks within the code under consideration. We identified pair-coverage as a viable criterion for generating input data for the time measurements.
  Year 4: An evaluation of the use of pair coverage as a criterion for generating input data for the measurement-based analysis of code segments.

3.3 Main Funding

The ArtistDesign NoE funds integration and building excellence with the partners, and with the European research landscape as a whole. Beyond this "glue" for integration and excellence, during Year 3 this activity has benefited from direct funding from:

- **AVACS**
  The AVACS project *(Automatic Verification and Analysis of Complex Systems)* is funded by the DFG and supports Saarland University.
  http://avacs.org
- **PREDATOR**
  The PREDATOR project *(Design for Predictability and Efficiency)* is a research project within the European Commission's 7th Framework Programme on Research, Technological Development and Demonstration. Bologna, Pisa, ETHZ, Saarland University and Dortmund take part in PREDATOR, along with AbsInt, Airbus and Bosch. The goal of PREDATOR is to reconcile performance and predictability, in particular for future multi-core architectures.
  http://predator-project.eu
- **COSTA**
  The COSTA project (*Compiler-Support for Timing Analysis*) is funded by the Austrian Science Fund (FWF) and supports TU Vienna. The project started in July 2006.
  http://costa.tuwien.ac.at

- **FORTAS**
  The FORTAS project (Formales Zeitanalyseframework für Echtzeitsysteme) is a cooperative project that brings together the orthogonal expertise of the TU Vienna real-time systems group and the group on formal methods and systems engineering from TU Darmstadt. Within FORTAS TU Vienna is supported by the Austrian Science Fund (FWF). FORTAS started in January 2007.

  http://www.fortastic.net

- Mälardalen has funding from the Foundation for Strategic Research (**PROGRESS** Strategic Research Centre, duration 2006-2010), the FP7 STREP **ALL-TIMES** (Integrating European Timing Analysis Technology) 2007-2010, and the Swedish Research Council "Worst-Case Execution Time Analysis of Parallel Systems", 2009-2011.
  http://www.mrtc.mdh.se/progress/
  http://www.all-times.org

- The University of York has funding from the EPSRC project **Using Machine Learning to Support the Development of Embedded Systems**, ref: EP/F00334X, since 2007.

-- Changes wrt Y2 deliverable --

No change from the Y2 deliverable.

4. Internal Reviewers for this Deliverable

- **Jan Gustafsson** (Mälardalen University, Sweden)
- **Christian Lengauer** (Univ. Passau, Germany)