WCET Analysis of Multicore Architectures

Andreas Gustavsson

May 17, 2010

Contents

Introduction to WCET Analysis 3

My Research Field 4
  Current Status 4

Conferences & Workshops 5
  IEEE – ECRTS 5
  IEEE – RTSS 5
  IEEE – RTAS 5
  IEEE – RTCSA 6
  ACM – LCTES 6
  UPMARC – MCC 6

Journals 7
  ACM – Transactions on Architecture and Code Optimisation 7
  ACM – Transactions on Computer Systems 7
  ACM – Transactions on Embedded Computing Systems 7
  ACM – Transactions on Programming Languages and Systems 8
  Elsevier – Computer Languages, Systems & Structures 8
  Elsevier – Journal of Parallel and Distributed Computing 9
  Elsevier – Journal of Systems and Software 9
  Elsevier – Parallel Computing 9
  IEEE – Computer 10
  IEEE – Transactions on Computers 10
  IEEE – Transactions on Parallel and Distributed Systems 11
  Springer – Real-Time Systems 12
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Researchers</td>
<td>12</td>
</tr>
<tr>
<td>Literature</td>
<td>13</td>
</tr>
<tr>
<td>Research</td>
<td>15</td>
</tr>
<tr>
<td>Hypothesis</td>
<td>15</td>
</tr>
<tr>
<td>Research Questions</td>
<td>15</td>
</tr>
<tr>
<td>Expected Results</td>
<td>16</td>
</tr>
<tr>
<td>Planned Collaborations and Contributions</td>
<td>16</td>
</tr>
<tr>
<td>Time Plan</td>
<td>16</td>
</tr>
<tr>
<td>References</td>
<td>19</td>
</tr>
</tbody>
</table>

**List of Figures**

1. Single-core and Multicore Comparison 4
Introduction to WCET Analysis

WCET (Worst-Case Execution Time) analysis is a very challenging problem. The execution time of a program can be a very complex function of the input data, the hardware, and the binary code executing on the hardware. The complexity mainly has two sources: the complex performance characteristics of modern processor architectures, and the sometimes very large number of possible paths through the code.

WCET analysis can be performed either statically or dynamically. In dynamic WCET analysis, measurements of the actual execution time of the software running on the target hardware are performed. This method is not guaranteed to execute the program’s worst-case path though (which could be e.g. some error-handling routine), and thus the WCET might be underestimated (there might exist an even worse execution time than the worst one measured).

In static WCET analysis, the actual program code and the properties of the target hardware are analysed. This method tries to find a tight (i.e., close to the actual) WCET, but always overestimates it.

Static WCET analysis is normally split into three subtasks: the low-level analysis, which attempts to find safe timing estimates for executions of code sequences, the flow analysis, which constrains the possible paths through the code, and the calculation, where the most time-consuming path is found, using information derived in the first two phases.

These three analysis phases only work for sequential processors. There is no current low level analysis which considers multiple processor cores – the closest is superscalar processors with instruction-level parallelism, but they emulate sequential processors. Current flow analysis and calculation methods are all based on sequential executions, and are thus not applicable to analysis of parallel systems.

There is currently an ongoing shift in computer architecture; from increasingly complex processors (as depicted in Figure 1a), which execute sequential code, to processors with multiple (often much simpler) cores (as depicted in Figure 1b) and the capability of running several threads completely in parallel. There are basically two strands of the “new” type of processors: commodity processors, with mostly homogeneous sets of cores (“multicore” processors), and Multiple-Processor System-on-a-Chip (MPSoC) architectures which are heterogeneous, custom design, and often made out of standard cores whereof some often are special-purpose. They are already now used in embedded systems with high performance demands.

Today, multicore/MPSoC processors host only a few cores, but it is expected that already in a few year’s time a chip will contain hundreds of cores.
(a) A single-core processor system with a complex processing unit.

(b) Typical multicore processor system – several simpler processing units sharing some level of cache.

Figure 1: A comparison between the “old” (single-core) and the “new” (multicore) style of designing processor systems.

These architectures will be quite unlike current multicore architectures, and the same will hold for the software. Research in WCET analysis must take this into account.

My Research Field

The goal of my research (as it was presented to me) is to find suitable models and methods for WCET analysis of parallel systems, mainly multicore and MPSoC architectures. Since the field is new, emphasis will be on basic models taking parallelism into account, mainly for some form of flow analysis and calculation.

While important, there should be less emphasis on the low-level analysis. The main reason for this is that my research group’s main competence is in flow analysis and calculation.

There is a possibility that there will take yet another phase (i.e., in addition to the flow analysis, low-level analysis and calculation phases) to cope with statically analysing multicore processors, though – nobody knows . . .

Current Status

Since I do not yet have a concrete and final method to present, I will simply present the current status of my research.

By using the Uppaal modelling and verification tool box, I try to model the impact of thread (or core) communication and synchronisation via shared
resources, such as caches and buses. Currently my model supports the execution of one (non-periodic) thread per core (i.e., there can be no scheduling of threads) and program-threads which synchronise via spin-locks. Data can be shared (and protected using spin-locks) among the threads in the program.

The model of the processor very much resembles that depicted in Figure 1b. I.e., there are multiple processing units with individual L1 instruction and data caches, and a shared L2 cache.

Using the verification subsystem of the Uppaal model-checker, different properties of the model can be verified and the WCET of the program can be found.

Conferences & Workshops

**IEEE – ECRTS**

The Euromicro Conference on Real-Time Systems (ECRTS) has a workshop specifically focused on WCET analysis. The workshop is funded and organised by the ARTIST community\(^1\) and covers all topics related to timing analysis, e.g. different approaches to WCET computation; low-level timing analysis, modelling and analysis of processor features; strategies to reduce the complexity of WCET analysis; tools for WCET analysis; and WCET analysis for multi-threaded and multicore systems. This is a very interesting workshop with respect to my research field.

**IEEE – RTSS**

The IEEE Real-Time Systems Symposium (RTSS) conference is aimed at presenting innovations in the field of real-time computing with respect to both theory and practise. It covers all aspects of real-time systems regarding design, analysis, implementation, evaluation, and experiences.

**IEEE – RTAS**

The IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) conference is focused on all aspects of real-time and embedded systems. Fields included are among others: real-time communications; embedded system security; energy-aware real-time systems; real-time system modelling and analysis; and WCET analysis.

---

\(^1\)A community of leading researchers within the field of embedded and real-time systems.
This conference also has a specialised track\(^2\) called Hardware/Software Integration and Co-Design. This track includes WCET analysis as one of the covered fields.

**IEEE – RTCSA**

The IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) covers all aspects within its three main tracks: embedded systems, real-time systems and ubiquitous computing. The real-time systems track includes fields such as real-time scheduling; timing (WCET) analysis; and energy aware real-time methods.

**ACM – LCTES**

The ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems (LCTES) is aimed at joining the programming languages and embedded systems engineering communities. This might not be the main conference to focus on with respect to my research field, but it does offer an opportunity to integrate with other research areas.

**UPMARC – MCC**

The Swedish Workshop on Multi-Core Computing (MCC) is a workshop organised by the Uppsala Programming for Multicore Architectures Research Center (UPMARC). The workshop is concentrated at multicore architectures and its purpose is to bring together researchers and practitioners from academia and industry to present and discuss the recent work in the area of multi-core computing. The workshop is organised as an activity within the network “Swedish Multicore Initiative”.

This workshop is organised in Sweden so it is a very suitable workshop to attend in order to keep up to date with the current state of multicore research.

\(^2\)At least regarding the RTAS conference held in 2010.
Journals

ACM – Transactions on Architecture and Code Optimisation

The ACM Transactions on Architecture and Code Optimisation (TACO) focuses on hardware, software, and system research, spanning the fields of computer architecture and code optimisation. Articles in TACO will present new techniques and concepts or report on experiences and experiments with actual systems. TACO welcomes articles that will give insights useful to architects, hardware or software developers, designers, builders and users.

This journal could be of interest to me in terms of making the WCET analysis problem known to computer system architects (both regarding hardware and software).

ACM – Transactions on Computer Systems

The ACM Transactions on Computer Systems (TOCS) publishes the newest findings of the computing research field. Papers published in TOCS are theoretical and conceptual explorations of operating systems, distributed systems and networks. The articles present design principles, case studies and experimental results in specification, processor management, memory and communication management, implementation techniques and protocols.

This journal could be useful for me when trying to find information regarding different types of computer architectures and systems. It might not be the top journal for me to publish papers in, however.

ACM – Transactions on Embedded Computing Systems

The design of embedded computing systems, both the software and hardware, increasingly relies on sophisticated algorithms, analytical models, and methodologies. The ACM Transactions on Embedded Computing Systems (TECS) aims to present the leading work relating to the analysis, design, behaviour, and experience with embedded computing systems.

TECS has special issues which cover different fields of research. The papers for these special issues are allowed to be quite large and some interesting examples of special issues from the past years are: the Special Issue on compilers, architecture, and synthesis for embedded systems; the Special Issue on Embedded Systems Week; and the Special Issue on Languages, Tools, and Compilers for Embedded Systems.

\(^{3} 25\) pages for the transaction of 2010.
This journal could be of interest for me to publish papers in; it does not specifically cover computer system analysis however.

**ACM – Transactions on Programming Languages and Systems**

The purpose of the ACM Transactions on Programming Languages and Systems (TOPLAS) is to present research results on all aspects of the design, definition, implementation, and use of programming languages and programming systems. The scope of TOPLAS includes: programming languages and their semantics; programming systems (systems to assist the programming task, such as compilers, run-time systems, and language environments); storage allocation and garbage collection; languages and methods for writing specifications; testing and verification methods; and algorithms specifically related to the implementation of language processors.

My research relates to analysis of the output of compilers (code) and the behaviour of run-time systems. This journal could be great for interacting with researchers in other fields and to increase the overall knowledge about the WCET analysis problem for compiler and system designers.

**Elsevier – Computer Languages, Systems & Structures**

The purpose of this journal is to publish both articles presenting original work and also review articles on programming languages, systems, structures, and theories. The objective is to concentrate on advances in programming language theory, as well as infrastructure developments still under investigation. Included topics are syntax, parsing, compilers, complexities, computability, semantics, automatic programming languages, special purpose languages, programming theories, programming documentation, memory management, micro programming, process control programming, real-time programming, programming for interactive systems, and distributed computer languages. Papers included in this journal only cover the theory of algorithmic and infrastructure systems, and structures regarding these languages and systems. The paper do not cover any particular or specialised uses or applications.

This journal could be a great source of knowledge to me.
Elsevier – Journal of Parallel and Distributed Computing

This international journal is directed to researchers, engineers, educators, managers, programmers, and users of computers who have particular interests in parallel processing and/or distributed computing. The Journal of Parallel and Distributed Computing publishes original research papers and review articles on the theory, design, evaluation, and use of parallel and/or distributed computing systems. The journal also features special issues on these topics; again covering the full range from the design to the use of targeted systems.

Research areas included are the theory of parallel and distributed computing; parallel algorithms and their implementation; innovative computer architectures; shared-memory multiprocessors; peer-to-peer systems; software tools and environments; languages, compilers, and operating systems; fault-tolerant computing; applications and performance analysis; parallel programming; and grid computing.

This journal represents a great source of knowledge with respect to my field of research.

Elsevier – Journal of Systems and Software

The Journal of Systems and Software publishes papers covering all aspects of programming methodology, software engineering, and related hardware-software-systems issues. Topics included are software development environments and tools, techniques for developing, validating, and maintaining software systems, software architecture and design, global software development, service orientation, agile approaches, mobile, multiprocessing, real-time, distributed, concurrent, and telecommunications systems. The journal publishes research papers, state-of-the-art surveys, and reports of practical experience. Occasionally, special issues are devoted to specific topic.

This journal could be of interest to me if trying to influence other fields of research to become more aware of the WCET analysis problem.

Elsevier – Parallel Computing

Parallel Computing is an international journal presenting the practical use of parallel computer systems, including high performance architecture, system software, programming systems and tools, and applications. Within this context the journal covers all aspects of high-end parallel computing.
The journal features original research work, tutorial and review articles as well as novel or illustrative accounts of application experience with (and techniques for) the use of parallel computers. Contributions can cover system software for parallel computer systems including programming languages (new languages as well as compilation techniques), operating systems (including middle-ware), and resource management (scheduling and load-balancing); enabling software including debuggers, performance tools, and system and numeric libraries; general hardware (architecture) concepts, new technologies enabling the realisation of such new concepts, and details of commercially available systems; software engineering and productivity as it relates to parallel computing; application or tool case studies demonstrating novel ways to achieve parallelism; performance measurement results on state-of-the-art systems; and parallel I/O systems both hardware and software.

This journal contains knowledge and topics which are quite interesting for my field of research. I will probably not publish any papers in this journal, though.

IEEE – Computer

This journal publishes articles that covers all aspects of computer science, computer engineering, technology, and applications. The articles selected for publication are peer-reviewed and edited to enhance readability for the general Computer reader. Computer is a resource that provides timely information about current research developments, trends, best practises, and changes in the profession. It delivers useful information that is applicable to everyday work environments.

This journal (actually magazine) is yet another possibility to increase the knowledge about the WCET analysis problem among a very wide audience.

IEEE – Transactions on Computers

The IEEE Transactions on Computers is a monthly publication with a wide distribution to researchers, developers, technical managers, and educators in the computer field. It publishes papers, brief contributions, and comments on research in areas of current interest. These areas include computer organisations and architectures; and real-time systems and embedded systems;

In general, my research field is covered by this journal. It might be of interest for me to publish articles in.
IEEE – Transactions on Parallel and Distributed Systems

The Transactions on Parallel and Distributed Systems (TPDS) is published monthly. The goal of TPDS is to publish a range of papers, comments on previously published papers, and survey articles that deal with topics related to multi-processor architecture design, analysis, and implementation; parallel software languages and compilers; scheduling and task partitioning; databases, operating systems, and programming environments for multiple-processor systems; analysis and design of parallel and/or distributed algorithms; modelling and simulation of multiple-processor systems; and conversion of software from sequential-to-parallel forms.

This journal could be of interest for me to publish articles in, regarding the analysis of multicore architectures and parallel software, since it covers both of these fields.


The Computer Journal is one of the longest-established journals serving all branches of the academic computer science community. It is currently published in three sections.

Section A: Computer Science, Methods and Tools publishes high quality research papers, review articles and special issues in all computer science topics other than those covered in Section B. Topics for inclusion in Section A may include areas such as theory, algorithms, software engineering, data structures, and complexity.

Section B: Networks and Computer Systems focuses on the latest ideas and research in computer systems and networks. This section is for researchers involved in the latest developments in topics such as architectures, mobile and wireless computing, network protocols, security, reliability and performance optimisation. In addition to research papers, this section will also publish commissioned reviews and special issues.

Section C: Computational Intelligence builds on artificial intelligence, robotics and machine intelligence to develop smart methods and tools that solve challenging problems in areas such as data mining, image processing, knowledge-based systems and the semantic web.

This journal might not be the primary one for me to publish papers in, but it could be of interest however.
The Real-Time Systems journal publishes papers that concentrate on real-time computing principles and applications. The contents include research papers, invited papers, project reports and case studies, standards and corresponding proposals for general discussion, and a partitioned tutorial on real-time systems as a continuing series.

The range of coverage is broad, including requirements engineering, specification and verification techniques, design methods and tools, programming languages, operating systems, scheduling algorithms, architecture, hardware and interfacing, dependability and safety, distributed and other novel architectures, wired and wireless communications, wireless sensor systems, distributed databases, artificial intelligence techniques, expert systems, and application case studies.

This is one of the primary journals for me to publish papers in. It covers all aspects of real-time systems, which is the field within which my research is valid.

Researchers

The most active research groups on WCET analysis are located in Sweden (Mälardalen and Linköping), Germany (Saarbrücken, Dortmund and Braunschweig), France (Toulouse and Rennes), Austria (Vienna), UK (York), Italy (Bologna), Spain (Cantabria and Valencia), and Switzerland (Zürich). More recently, the topic of code-level timing analysis has found more attention outside of Europe by research groups in the US (North Carolina and Florida), Australia, and Singapore.

Some of the leading researchers within WCET analysis are (other than my fellow researchers here at MDH; Björn Lisper, Jan Gustafsson, Andreas Ermedahl, Stefan Bygde, Marcelo Santos and Linus Källberg) Reinhard Wilhelm, Rolf Ernst, Peter Puschner and Wei Zhang.

Professor Reinhard Wilhelm is the chair for Programming Languages and Compiler Construction at Saarland University. He has several publications on WCET analysis (mostly aiming at single-core processors) and is a member of the ARTIST2 Network of Excellence and the ARTIST DESIGN Network of Excellence.

Professor Rolf Ernst is situated at the Technische Universität in Braunschweig. He has numerous publications on WCET analysis, which have also been starting to focus somewhat on multicore processor systems.

Professor Peter Puschner and Assistant Professor Raimund Kirner are
situated at the Vienna University of Technology. Their main research interest is on real-time systems, with a focus on the WCET analysis of real-time programs. Within this focus they are working on the definition of the key problems of WCET analysis, the development of theories and concepts for WCET computation, and the realisation of WCET tools that build on the theoretical results. These researchers have strongly influenced the state of the art within these fields, with numerous conference and journal papers.

Associate professor Wei Zhang at Southern Illinois University, Carbondale, and his former Ph.D. student Jun Yan has conducted research that is very much related to mine. They very recently published a couple of papers on how to statically analyse a multicore processor with a shared L2 instruction cache. This will be further mentioned below. Unfortunately, Jun Yan is no longer a member of the research team and will probably not publish any more papers on the WCET analysis topic.

Ph.D. student Lan Wu on the other hand is a member of that same research team and is still performing research which is even more related to mine. He and Prof. Wei Zhang just recently published a paper, for the work-in-progress session of the IEEE – RTAS 2010 conference, on using model-checking to perform WCET analysis. More on this in the next section.

Another research team has also started considering using model-checking to perform WCET analysis. Ph.D. student Mingson Lv at the Northeastern University in China and Prof. Wang Yi and Ph.D. student Nan Guan at the Uppsala University have also published a paper on this topic for the work-in-progress session of the IEEE – RTAS 2010 conference. More on this in the next section.

**Literature**

WCET related research started with the introduction of timing schemas by Shaw in 1989 [13]. Shaw presents rules to collapse the CFG (Control Flow Graph) of a program until a final single value represents the WCET. Excellent overviews of the WCET research from the years 2000 and 2008 can be found in [10] and [14] respectively. My field of research (i.e., applying WCET analysis for multicore processors) is quite new, so there is no solid basis of previous research to stand on.

One of the most used approaches to solving the WCET problem (for single-core processors) is by computing ILP (Integer Linear Programming) problems. Two ways of computing the ILP problem are by using a graph-based approach [11], and by using IPET (Implicit Path Enumeration) [5]. A comparison of these two approaches is performed in [3]. The graph-based
approach is conducted by using model-checking in Uppaal \cite{4}. It is shown that IPET outperforms the model-checking-based approach, but that model-checking allows for calculating tight WCET bounds and easy integration of complex hardware models. A combined approach is proposed, where model-checking is used to analyse local regions of the code, while IPET is used to solve the global analysis. Another motivation to why model-checking could be useful in WCET analysis can be found in \cite{7}.

This is what my current research is currently focusing on – investigating small regions of code (more specifically spin-lock-synchronisation, but on a multicore system) using model-checking. My research might turn more toward the IPET approach in the future, depending on the actual usability of the model-checking approach.

Some research has been conducted in the field of WCET analysis for multicore and other types of multi-processor systems. In \cite{16}, a static method for analysing a multicore processor with a shared L2 instruction cache is presented. A limitation of this analysis is that the L1 data cache is assumed to be perfect (i.e., all accesses are assumed to be hits – which is generally not the case) and thus does not affect the contents of the L2 cache. Based on this work, the same authors also address the same problem for the case that the shared L2 cache is direct-mapped \cite{17}. This work is very closely related to my research.

In \cite{6} and \cite{15}, model-checking is used to perform WCET analysis. Both papers are closely related to my research, but mainly propose methods to reduce the size of the state space by altering the program model without affecting the true WCET of the program. This is a very important aspect when using model-checking overall. If the model is too large and complex, the state space will “explode”, which means that the number of possible states is very large and analysing the model becomes infeasible. My approach is more focused on analysing the impact on the WCET from allowing synchronising (and communicating) tasks.

Other than this, no great efforts (as far as I know) have been performed within the field of WCET analysis for multicore processors and parallel software. A possibility for future collaboration is to contact Prof. Wei Zhang, since his research is closely related to mine.

Some other research addresses the problem of (low) predictability in the multicore processor. This work mostly gives multicore-design guidelines and suggestions on how to use additional hardware to increase the predictability. In an extension to the method found in \cite{16}, memory bits for each instruction is used to determine whether the instruction should be cached or not \cite{2}. E.g., to avoid pollution of the shared cache, “Static Single Usage” \cite{2} instructions (i.e., instructions in the program that are only referenced/executed once)
should not be cached. This generates the possibility to determine a tighter WCET.

In [9], arbiters (hardware circuits) are added to a shared memory multicore processor to synchronise the memory accesses from different cores in order to increase the timing-predictability of the system. The result is a multicore architecture that can be analysed with existing single-core WCET analysis tools.

GAMC [8] is a SDRAM controller which upper bounds the delay a core can suffer from memory-interferences from other cores. This is an important approach since the largest memory access latency will occur when accessing the main memory. The result is a tight WCET approximation which only differs a few percent from the largest measured execution time.

In [1] and [12], TDMA-based memory bus access policies are introduced to make all memory accesses predictable, regarding the WCET.

Research

Hypothesis

“The possibilities of the current methods used to perform WCET analysis can be extended to include analysis of parallel hardware and software.”

Research Questions

These questions regard the WCET analysis of parallel software (including communicating and synchronising threads) executing on (shared- or distributed memory) multi-processor (multicore) systems.

- Is model-checking a feasible method for performing the analysis?
- Is there a feasible method to statically perform the analysis?
- Can the analysis be included in the current static analysis phases, or is it necessary to add an analysis phase to the current (flow, low-level, and calculation) analysis phases?
- What is the achievable precision of the method and how does it scale with respect to analysis of real-world parallel hardware and software (i.e., what is the complexity of the method)?
Expected Results

To start with, a method based on model-checking using Uppaal will be investigated. Based on the real-world applicability of this method, I will either try to integrate this method with the already available static analysis methods or try to develop a method based on static analysis. The model-checking method is expected to not be scalable enough to be feasibly used for real-world problems.

Therefore, I expect to find a static analysis method (or at least contribute with parts of it) which is able to feasibly estimate the WCET of parallel systems. By feasibly, I mean that the resulting WCET estimate should be both tight and safe (i.e., it should be close to, and guaranteed not to underestimate the actual WCET).

The first steps in the development of a static analysis method could be to define a formal semantic, which also describes timing properties, for a suitable parallel programming model. Then a general WCET analysis for this model could be defined and some limited classes of programs, for which simplified analyses are possible, could be studied.

Planned Collaborations and Contributions

A collaboration with (and a visit to) the research team including Professor Wei Zhang might be appropriate. I am quite sure that research on this new field will start to be presented also by other research groups in the near future, however – there may be many opportunities for collaboration.

I hope and expect to publish a number of papers on the research field. I also intend to attend a number of the above mentioned conferences and workshops. This summer I will present a paper at the WCET workshop at the ECRTS conference.

Time Plan

This is a rough plan of my Ph.D. studies:

2009

- Take courses worth approximately 30hp.
- Assist in the “Computer Architecture” course.
- Attend the MCC’09 workshop.
2010
- Licentiate proposal.
- Take courses worth approximately 15hp.
- Assist in developing and teaching the new “Parallel Systems” course (7.5hp).
- Continue to investigate the model-checking-based method for WCET analysis.
- Present my paper at the WCET workshop in the ECRTS’10 conference.
- At least the above mentioned publication.

2011
- Continue to investigate the model-checking-based method for WCET analysis.
- Start developing a static approach to analysing parallel systems (define formal semantics).
- At least one publication.
- Attend at least one conference or workshop.

2012
- Licentiate defence.
- Ph.D. proposal.
- Take courses worth approximately 15hp.
- Continue to develop a static approach to analysing parallel systems (define a general WCET analysis).
- Start developing a tool used in the static analysis, possibly integrating it with SWEET.
- At least one publication.
- Attend at least one conference or workshop.
2013

- Take courses worth approximately 15hp.
- Continue to develop a static approach to analysing parallel systems.
- Continue to develop a tool used in the static analysis, possibly integrating it with SWEET.
- At least one publication.
- Attend at least one conference or workshop.

2014

- Continue to develop a static approach to analysing parallel systems.
- Continue to develop a tool used in the static analysis, possibly integrating it with SWEET.
- At least one publication.
- Attend at least one conference or workshop.
- Ph.D. defence.
References


