2006 WCET Abstracts Collection
6th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis

F. Mueller

North Carolina State University, USA
mueller@cs.ncsu.edu

Abstract. On the 4th of July, 2006, the 6th International Workshop on Worst-Case Execution Time Analysis (WCET'06) was held in Dresden, Germany, co-located with the 18th Euromicro International Conference on Real-Time Systems (ECRTS'06), both with support of Euromicro Technical Committee. The goal of the workshop was to bring together people from academia, tool vendors and users in industry that are interested in all aspects of timing analysis for real-time systems. The workshop provided a relaxed forum to present and discuss new ideas, new research directions, and to review current trends in this area. The workshop was based on short presentations that encouraged discussion by the attendees. Abstracts of the presentations are put together in this paper. Links to extended abstracts or full papers are provided. The first section directs to the preface of the proceedings.

Keywords. WCET'06, workshop proceedings, abstracts collection

2006 WCET Preface – Proceedings of the 6th Intl. Workshop on Worst-Case Execution Time Analysis (WCET'06)

Frank Mueller

On the 4th of July, 2006, the 6th International Workshop onWorst-Case Execution Time Analysis (WCETŠ06) was held in Dresden, Germany, co-located with the 18th Euromicro International Conference on Real-Time Systems (ECRTSŠ06), both with support of Euromicro Technical Committee. The goal of the workshop was to bring together people from academia, tool vendors and users in industry that are interested in all aspects of timing analysis for real-time systems. The workshop provided a relaxed forum to present and discuss new ideas, new research directions, and to review current trends in this area. The workshop was based on short presentations that should encourage discussion by the attendees.

Keywords: WCET'06, workshop proceedings, preface
Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/679

Algorithms for Infeasible Path Calculation

Jan Gustafsson, Andreas Ermedahl, and Björn Lisper

Static Worst-Case Execution Time (WCET) analysis is a technique to derive upper bounds for the execution times of programs. Such bounds are crucial when designing and verifying real-time systems. One key component in static WCET analysis is to derive flow information, such as loop bounds and infeasible paths for the analysed program. Such flow information can be provided as either as annotations by the user, can be automatically calculated by a flow analysis, or by a combination of both. To make the analysis as simple, automatic and safe as possible, this flow information should be calculated automatically with no or very limited user interaction. In this paper we present three novel algorithms to calculate infeasible paths. The algorithms are all designed to be simple and efficient, both in terms of generated flow facts and in analysis running time. The algorithms have been implemented and tested for a set of WCET benchmarks programs.

Keywords: Worst case execution time, real-time, control flow analysis, abstract interpretation, infeasible paths

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/667

Comparing WCET and Resource Demands of Trigonometric Functions Implemented as Iterative Calculations vs. Table-Looku

Raimund Kirner, Markus Groessing, Peter Puschner

Trigonometric functions are often needed in embedded real-time software. To fulfill concrete resource demands, different implementation strategies of trigonometric functions are possible.

In this paper we analyze the resource demands of iterative calculations compared to other implementation strategies, using the trigonometric functions as a case study. By analyzing the worst-case execution time (WCET) of the different calculation techniques of trigonometric functions we got the surprising result that the WCET of iterative calculations is quite competitive to alternative calculation techniques, while their economics on memory demand is far superior. Finally, a discussion of the general applicability of the obtained results is given as a design guide for embedded software.

Keywords: Worst-case execution time, WCET analysis, table lookup, iterative computation, Taylor series, resource demands

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/669

History-based Schemes and Implicit Path Enumeration

Claire Burguière, Christine Rochange

The Implicit Path Enumeration Technique is often used to compute the WCET of control-intensive programs. This method does not consider execution paths as ordered sequences of basic blocks but instead as lists of basic blocks with their respective execution counts. This way of describing an execution path is adequate to compute its execution time, provided that safe individual WCETs for the blocks are known. Recently, a model for branch prediction has been integrated into WCET computation with IPET. This model generates safe estimations of the branch misprediction counts. However, we show in this paper that these counts can be over-estimated because IPET does consider simplified flow information that do not completely reflect the program semantics. We show how additional information on nested loops can be specified so that the model provides tighter WCET estimations.

Keywords: WCET, IPET (Implicit Path Enumeration Technique), branch prediction

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/670

A Definition and Classification of Timing Anomalie

Jan Reineke, Bjoern Wachter, Stephan Thesing, Reinhard Wilhelm, Ilia Polian, Jochen Eisinger, Bernd Becker

Timing Anomalies are characterized by counterintuitive timing behaviour. A locally faster execution leads to an increase of the execution time of the whole program. The presence of such behaviour makes WCET analysis more diścult: It is not safe to assume local worst-case behaviour wherever the analysis encounters uncer- tainty. Existing definitions of Timing Anomalies are rather imprecise and intuitive in nature. Some do not cover all kinds of known Timing Anomalies. After giving an overview of related work, we give a concise formal definition of Timing Anomalies. We then begin to identify diőerent classes of anomalies. One of these classes, coined Scheduling Timing Anomalies, coincides with previous restricted definitions.

Keywords: Timing analysis, Worst-case execution time, Timing anomalies, Scheduling Anomalies, Abstraction

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/671

PLRU Cache Domino Effects

Christoph Berg

Domino effects have been shown to hinder a tight prediction of worst case execution times (WCET) on real-time hardware. First investigated by Lundqvist and Stenström, domino effects caused by pipeline stalls were shows to exist in the PowerPC by Schneider. This paper extends the list of causes of domino effects by showing that the pseudo LRU (PLRU) cache replacement policy can cause unbounded effects on the WCET. PLRU is used in the PowerPC PPC755, which is widely used in embedded systems, and some x86 models.

Keywords: Embedded systems, predictability, cache memory, PLRU, domino effects, timing anomalies

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/672

Design of a WCET-Aware C Compiler

Heiko Falk, Paul Lokuciejewski, Henrik Theiling

This paper presents techniques to tightly integrate worst-case execution time information into a compiler framework. Currently, a tight integration of WCET information into the compilation process is strongly desired, but only some ad-hoc approaches have been reported currently. Previous publications mainly used self-written WCET estimators with very limited functionality and preciseness during compilation. A very tight integration of a high quality industry-relevant WCET analyzer into a compiler was not yet achieved up to now. This work is the first to present techniques capable of achieving such a tight coupling between a compiler and the WCET analyzer aiT. This is done by automatically translating the assembly-like contents of the compiler's low-level intermediate representation (LLIR) to aiT's exchange format CRL2. Additionally, the results produced by the WCET analyzer are automatically collected and re-imported into the compiler infrastructure. The work described in this paper is smoothly integrated into a C compiler environment for the Infineon TriCore processor. It opens up new possibilities for the design of WCET-aware optimizations in the future.

The concepts for extending the compiler infrastructure are kept very general so that they are not limited to WCET information. Rather, it is possible to use our structures also for multi-objective optimization of e.g. best-case execution time (BCET) or energy dissipation.

Keywords: WCET, compiler, multi-objective, intermediate representation, ICD-C, LLIR, CRL2, aiT

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/673

Loop Nest Splitting for WCET-Optimization and Predictability Improvement

Heiko Falk, Martin Schwarzer

This paper presents the influence of the loop nest splitting source code optimization on the worst-case execution time (WCET). Loop nest splitting minimizes the number of executed if-statements in loop nests of embedded multimedia applications. Especially loops and if-statements of high-level languages are an inherent source of unpredictability and loss of precision for WCET analysis. This is caused by the fact that it is difficult to obtain safe and tight worst-case estimates of an application's flow of control through these high-level constructs. In addition, the corresponding control flow redirections expressed at the assembly level reduce predictability even more due to the complex pipeline and branch prediction behavior of modern embedded processors.

The analysis techniques for loop nest splitting are based on precise mathematical models combined with genetic algorithms. On the one hand, these techniques achieve a significantly more homogeneous structure of the control flow. On the other hand, the precision of our analyses leads to the generation of very accurate high-level flow facts for loops and if-statements. The application of our implemented algorithms to three real-life multimedia benchmarks leads to average speed-ups by 25.0% - 30.1%, while WCET is reduced between 34.0% and 36.3%.

Keywords: Loop Nest Splitting, Source Code Optimization, WCET, ACET, flow facts, polytope

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/674

Combining Symbolic Execution and Path Enumeration in Worst-Case Execution Time Analysis

Djemai Kebbal, Pascal Sainrat

This paper examines the problem of determining bounds on execution time of real-time programs. Execution time estimation is generally useful in real-time software verification phase, but may be used in other phases of the design and execution of real-time programs (scheduling, automatic parallelizing, etc.). This paper is devoted to the worst-case execution time (WCET) analysis. We present a static WCET analysis approach aimed to automatically extract flow information used in WCET estimate computing. The approach combines symbolic execution and path enumeration. The main idea is to avoid unfolding loops performed by symbolic execution-based approaches while providing tight and safe WCET estimate.

Keywords: Static WCET analysis, flow analysis, symbolic execution, path enumeration, loop analysis

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/675

A Framework for Response Times Calculation Of Multiple Correlated Events

Simon Schliecker, Matthias Ivers, Jan Staschulat, Rolf Ernst

Many approaches to determine the response time of a task have difficulty to model tasks with multiple memory or coprocessor accesses with variable access times during the execution. As the request times highly depend on system setup and state, they can not be trivially bounded. If they are bounded by a constant value, large discrepancies between average and worst case make the focus on single worst cases vulnerable to overestimation.

We present a novel approach to include remote busy time in the execution time analysis of tasks. We determine the time for multiple requests by a task efficiently and and far less conservative than previous approaches. These requests may be disturbed by other events in the system. We show how to integrate such a multiple event busy time analysis to take into account behavior of tasks that voluntarily suspend themselves and require multiple data from remote parts of the system.

Keywords: Response time analysis, multiple memory accesses, multiprocessor, hard real-time, busy time

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/676

Towards Formally Verifiable WCET Analysis for a Functional Programming Language

Kevin Hammond, Roy Dyckhoff, Christian Ferdinand, Reinhold Heckmann, Martin Hofmann, Steffen Jost, Hans-Wolfgang Loidl, Greg Michaelson, Robert Pointon, Norman Scaife, Jocelyn Serot, Andy Wallace

This paper describes ongoing work aimed at the construction of formal cost models and analyses to yield verifiable guarantees of resource usage in the context of real-time embedded systems. Our work is conducted in terms of the domain-specific language Hume, a language that combines functional programming for computations with finitestate automata for specifying reactive systems. We outline an approach in which high-level information derived from source-code analysis can be combined with worst-case execution time information obtained from high quality abstract interpretation of low-level binary code.

Keywords: Worst-case execution time, functional programming, Hume, cost model, asynchronous, finite state machine

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/677

PapaBench: a Free Real-Time Benchmark

F. Nemer, H. Cassé, P. Sainrat, J.P. Bahsoun

This paper presents PapaBench, a free real-time benchmark and compares it with the existing benchmark suites. It is designed to be valuable for experimental works in WCET computation and may be also useful for scheduling analysis. This bench is based on the Paparazzi project that represents a real-time application, developed to be embedded on different Unmanned Aerial Vehicles (UAV). In this paper, we explain the transformation process of Paparazzi applied to obtain the PapaBench. We provide a high level AADL model, which reflects the behaviors of each component of the system and their interactions. As the source project, Paparazzi, PapaBench is delivered under the GNU license and is freely available to all researchers. Unlike other usual benchmarks widely used for WCET computation, this one is based on a real and complete real-time embedded application.

Keywords: Real-Time Benchmark, Complete Application, Worst Case Execution Time (WCET) Computation, Modeling

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2006/678