Time-Predictable Embedded Software on Multi-Core Platforms: Analysis and Optimization

Sudipta Chattopadhyay
Linköping University

Abhik Roychoudhury
National University of Singapore

Jakob Rosén
Linköping University

Petru Eles
Linköping University

Zebo Peng
Linköping University

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Electronic Design Automation

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

S. Chattopadhyay, A. Roychoudhury, J. Rosén, P. Eles, Z. Peng. *Time-Predictable Embedded Software on Multi-Core Platforms: Analysis and Optimization*. Foundations and Trends® in Electronic Design Automation, vol. 8, no. 4-3, pp. 199–356, 2014.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-60198-795-2

Foundations and Trends® in Electronic Design Automation
Volume 8, Issue 4-3, 2014

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Electronic Design Automation publishes survey and tutorial articles in the following topics:

- System level design
- Behavioral synthesis
- Logic design
- Verification
- Test
- Physical design
- Circuit level design
- Reconfigurable systems
- Analog design
- Embedded software and parallel programming
- Multicore, GPU, FPGA, and heterogeneous systems
- Distributed, networked embedded systems
- Real-time and cyberphysical systems

Information for Librarians

Foundations and Trends® in Electronic Design Automation, 2014, Volume 8, 4 issues. ISSN paper version 1551-3939. ISSN online version 1551-3947. Also available as a combined paper and online subscription.

Foundations and Trends® in Electronic Design Automation
Vol. 8, No. 4-3 (2014) 199–356

DOI: 10.1561/1000000037

now
the essence of knowledge

Time-Predictable Embedded Software on Multi-Core Platforms: Analysis and Optimization

Sudipta Chattopadhyay
Linköping University
sudipta.chattopadhyay@liu.se

Abhik Roychoudhury
National University of Singapore
abhik@comp.nus.edu.sg

Jakob Rosén
Linköping University
jakob.rosen@gmail.com

Petru Eles
Linköping University
petru.eles@liu.se

Zebo Peng
Linköping University
zebo.peng@liu.se

Contents

Abstract

Multi-core architectures have recently gained popularity due to their high-performance and low-power characteristics. Most of the modern desktop systems are now equipped with multi-core processors. Despite the wide-spread adaptation of multi-core processors in desktop systems, using such processors in embedded systems still poses several challenges. Embedded systems are often constrained by several extra-functional aspects, such as time. Therefore, providing guarantees for time-predictable execution is one of the key requirements for embedded system designers. Multi-core processors adversely affect the time-predictability due to the presence of shared resources, such as shared caches and shared buses. In this contribution, we shall first discuss the challenges imposed by multi-core architectures in designing time-predictable embedded systems. Subsequently, we shall describe, in details, a comprehensive solution to guarantee time-predictable execution on multi-core platforms. Besides, we shall also perform a discussion of different techniques to provide an overview of the state-of-the-art solutions in this topic. Through this work, we aim to provide a solid background on recent trends of research towards achieving time-predictability on multi-cores. Besides, we also highlight the limitations of the *state-of-the-art* and discuss future research opportunities and challenges to accomplish time-predictable execution on multi-core platforms.

S. Chattopadhyay, A. Roychoudhury, J. Rosén, P. Eles, Z. Peng. *Time-Predictable Embedded Software on Multi-Core Platforms:*
Analysis and Optimization. Foundations and Trends® in Electronic Design Automation, vol. 8, no. 4-3, pp. 199–356, 2014.

DOI: 10.1561/1000000037.

1

Introduction

Real-time, embedded systems often need to satisfy several extra-functional constraints, such as timing. In particular, for hard real-time systems, such timing constraints are strictly enforced. Violation of these timing constraints may have serious consequences, potentially costing human lives. Therefore, static timing-analysis of hard real-time systems has emerged to be a critical problem to solve.

In general, a real-time, embedded application is made of several components, usually called *tasks*. Therefore, timing analysis of embedded software is typically performed in two separate phases: *(i)* a low-level analysis which derives the *worst case execution time* (WCET) and *best case execution time* (BCET) of individual tasks, and *(ii)* a system-level schedulability analysis which uses the WCET/BCET derived for each task and computes the overall timing characteristics of the application. In this monograph, we shall primarily focus our discussion on low-level WCET analysis.

WCET analysis of an embedded software is typically performed in three stages: *(i)* a flow-analysis using the control flow graph (CFG) of the program (to determine infeasible paths and loop bounds), *(ii)* micro-architectural modeling (to determine the worst case execution time of each basic block in the CFG) and *(iii)* a calculation phase which combines the outcome of

flow-analysis and micro-architectural modeling to derive the worst case execution time (WCET) of the entire program. Micro-architectural modeling systematically considers the timing effects of underlying processor features, such as pipeline, caches, branch prediction and so on. For single-core processors, such a micro-architectural modeling involves the analysis of a single program occupying the processor. However, this criterion no longer holds with multi-core processors. Since their inception, multi-core processors have widely been adopted due to their high-performance and low-power characteristics. Unfortunately, multi-core processors pose some significant challenges in terms of time-predictability. Basically, these challenges arise due to the presence of shared resources, such as shared caches and shared buses [5]. The presence of shared resources makes the WCET analysis significantly more complex than the WCET analysis on single-core processors. In particular, micro-architectural modeling is affected due to the presence of inter-core interferences, such as shared cache conflicts or bus contention. Through this monograph, we primarily aim to highlight the recent advances to address such challenges.

As mentioned in the preceding paragraph, shared resources are the key bottlenecks to build time-predictable embedded software on multi-core platforms. The content of a shared cache is modified by several programs running in parallel on different cores. Therefore, the modeling of inter-core cache conflicts is important to estimate the shared-cache latency accurately. For bus-based systems, shared buses introduce variable access latency to the shared resources (*e.g.* shared caches and main memory). Such a variable access latency highly depends on the *bus contention*, which in turn depends on the amount of memory traffic generated by different cores. In this monograph, we shall first describe an approach to model the timing behavior of shared caches [21]. Such a modeling systematically combines abstract interpretation with *state-of-the-art* program-verification techniques (*e.g.* model checking and symbolic execution). In particular, such an approach leverages both the *scalability* offered by abstract interpretation and the *accuracy* offered by program-verification methods to build a tight modeling of shared caches. We then describe works on analyzing timing behavior for static bus-arbitration policies, such as *time division multiple access* (TDMA). Even with static bus-arbitration policies, an accurate analysis of shared-bus delay is complex. This

is due to the reason that bus delay highly depends on the *context*, such as individual loop iterations and procedure calls. In the worst-case, each loop iteration may experience different bus delay. We describe works [69, 9, 22] in this direction whose requirements range from *full-fledged loop unrolling* to *avoiding loop unrolling altogether*, depending on the analysis accuracy.

Subsequently, we discuss the development of a full-fledged WCET analysis framework by combining the modeling of shared resources [18]. Such a combination is non-trivial due to the possible presence of *timing anomalies* [59]. In the presence of timing anomalies, a local worst-case (*e.g.* a cache miss or maximum bus delay) may not lead to the overall WCET of a program. As a result, it is *unsound* to model the timing behavior of each micro-architectural component and get the overall timing behavior by a simple composition of individual timing models. This framework systematically models the timing interaction of shared resources with the rest of the micro-architectural features (*e.g.* pipeline, branch prediction) and it does not assume a *timing-anomaly-free* architecture. The WCET analysis framework is built on top of `Chronos` [52], a freely-available, open-source WCET analysis tool. We show the evaluation of this analysis framework via several experiments.

Besides modeling individual micro-architectural features in multi-core processors, predictability of embedded software can also benefit from customized compiler optimizations and time-predictable multi-core hardware. In this direction, we discuss an optimization of bus schedules to improve time-predictability. Specifically, we describe the generation of customized bus schedules that may greatly improve the WCET of a program [69]. Finally, we discuss several designs of time-predictable hardware to reduce the pessimism in the WCET analysis on multi-core platforms.

The main purpose of this monograph is to give the readers a thorough background on time-predictability for multi-core platforms. Therefore, we have also performed a discussion of research activities by several research groups in this area. This discussion provides a comprehensive overview of the state-of-the-art solutions in the respective topic. In particular, our discussion reveals that the area is fast evolving and there is an active interest by real-time research groups on the topic discussed in this monograph. Finally, in the concluding section of this monograph, we have highlighted a set of open challenges in achieving high-performance and time-predictable

embedded software on multi-core platforms. We hope that this monograph will provide a foundation of building time-predictable software on multi-core platforms and it will help the research community to address the existing challenges in this area.

References

[1] aiT AbsInt. http://www.absint.com/ait.

[2] The KLEE Symbolic Virtual Machine. http://klee.llvm.org.

[3] Multi-Core Execution of Hard Real-Time Applications Supporting Analysability. http://ginkgo.informatik.uni-augsburg.de/merasa-web/.

[4] Time-predictable Multi-core Architecture for Embedded Systems. http://www.t-crest.org/.

[5] Andreas Abel, Florian Benz, Johannes Doerfert, Barbara Dörr, Sebastian Hahn, Florian Haupenthal, Michael Jacobs, Amir H. Moin, Jan Reineke, Bernhard Schommer, and Reinhard Wilhelm. Impact of resource sharing on performance and performance prediction: A survey. In *International Conference on Concurrency Theory*, pages 25–43, 2013.

[6] Sebastian Altmeyer, Robert I Davis, and Claire Maiza. Cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems. In *Real-Time Systems Symposium*, pages 261–271, 2011.

[7] Sebastian Altmeyer, Claire Maiza, and Jan Reineke. Resilience analysis: tightening the CRPD bound for set-associative caches. In *ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems*, pages 153–162, 2010.

[8] Sidharta Andalam, Partha Roop, and Alain Girault. Predictable multithreading of embedded applications using PRET-C. In *International Conference on Formal Methods and Models for Codesign*, pages 159–168, 2010.

[9] Alexandru Andrei, Petru Eles, Zebo Peng, and Jakob Rosén. Predictable implementation of real-time applications on multiprocessor systems-on-chip. In *IEEE International Conference on VLSI Design*, pages 103–110, 2008.

[10] Todd Austin, Eric Larson, and Dan Ernst. Simplescalar: An infrastructure for computer system modeling. *Computer*, 35(2):59–67, 2002.

[11] Philip Axer, Rolf Ernst, Heiko Falk, Alain Girault, Daniel Grund, Nan Guan, Bengt Jonsson, Peter Marwedel, Jan Reineke, Christine Rochange, Maurice Sebastien, Reinhard von Hanxleden, Reinhard Wilhelm, and Wang Yi. Building timing predictable embedded systems. *ACM Transactions on Embedded Computing Systems*, Accepted for publication.

[12] Gogul Balakrishnan and Thomas Reps. Analyzing memory accesses in x86 executables. In *Compiler Construction*, pages 5–23, 2004.

[13] Rajeshwari Banakar, Stefan Steinke, Bo-Sik Lee, M Balakrishnan, and Peter Marwedel. Scratchpad memory: design alternative for cache on-chip memory in embedded systems. In *International Symposium on Hardware/software Codesign*, pages 73–78, 2002.

[14] Christoph Berg. PLRU cache domino effects. *International Workshop on Worst-Case Execution Time (WCET) Analysis*, 2006.

[15] Bach Duy Bui, Rodolfo Pellizzoni, and Marco Caccamo. Real-time scheduling of concurrent transactions in multidomain ring buses. *IEEE Trans. Computers*, 61(9):1311–1324, 2012.

[16] Dai N. Bui, Edward A. Lee, Isaac Liu, Hiren D. Patel, and Jan Reineke. Temporal isolation on multiprocessing architectures. In *Design Automation Conference*, pages 274–279, 2011.

[17] Cristian Cadar, Daniel Dunbar, and Dawson R. Engler. KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs. In *USENIX Symposium on Operating Systems Design and Implementation*, pages 209–224, 2008.

[18] Sudipta Chattopadhyay, Lee Kee Chong, Abhik Roychoudhury, Timon Kelter, Peter Marwedel, and Heiko Falk. A unified WCET analysis framework for multi-core platforms. *ACM Transactions on Embedded Computing Systems*, Accepted for publication (An earlier version appeared in IEEE Real-Time and Embedded Technology and Applications Symposium, 2012).

[19] Sudipta Chattopadhyay and Abhik Roychoudhury. Unified cache modeling for WCET analysis and layout optimizations. In *IEEE Real-time Systems Symposium*, pages 47–56, 2009.

[20] Sudipta Chattopadhyay and Abhik Roychoudhury. Static bus schedule aware scratchpad allocation in multiprocessors. In *ACM SIGPLAN/SIGBED 2011 conference on Languages, compilers, and tools for embedded systems*, pages 11–20, 2011.

[21] Sudipta Chattopadhyay and Abhik Roychoudhury. Scalable and precise refinement of cache timing analysis via path-sensitive verification. *Real-Time Systems*, 49(4):517–562, 2013 (An earlier version appeared in IEEE Real-time Systems Symposium, 2011).

[22] Sudipta Chattopadhyay, Abhik Roychoudhury, and Tulika Mitra. Modeling shared cache and bus in multi-cores for timing analysis. In *International Workshop on Software and Compilers for Embedded Systems*, pages 6:1–6:10, 2010.

[23] Lee Kee Chong, Clément Ballabriga, Van-Thuan Pham, Sudipta Chattopadhyay, and Abhik Roychoudhury. Towards parallel programming models for predictability. In *IEEE Real-time Systems Symposium*, 2013.

[24] Edmund Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.

[25] Edmund Clarke, Daniel Kroening, and Flavio Lerda. A tool for checking ANSI-C programs. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 168–176. 2004.

[26] Edmund M. Clarke, E Allen Emerson, and A Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263.

[27] EG Coffman Jr and Ronald L. Graham. Optimal scheduling for two-processor systems. *Acta Informatica*, 1(3):200–213, 1972.

[28] Antoine Colin and Isabelle Puaut. Worst case execution time analysis for a processor with branch prediction. *Real-Time Systems*, 18(2-3):249–274, 2000.

[29] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Symposium on Principles of programming languages*, pages 238–252, 1977.

[30] Georgia Giannopoulou, Kai Lampka, Nikolay Stoimenov, and Lothar Thiele. Timed model checking with abstractions: towards worst-case response time analysis in resource-sharing manycore systems. In *International Conference on Embedded Software*, pages 63–72, 2012.

[31] Kees Goossens, John Dielissen, and Andrei Radulescu. Æthereal network on chip: concepts, architectures, and implementations. *Design & Test of Computers, IEEE*, 22(5):414–421, 2005.

[32] Sven Goossens, Jasper Kuijsten, Benny Akesson, and Kees Goossens. A reconfigurable real-time SDRAM controller for mixed time-criticality systems. In *International Conference on Hardware/Software Codesign and System Synthesis*, pages 1–10, 2013.

[33] Daniel Grund and Jan Reineke. Abstract interpretation of FIFO replacement. In *Static Analysis Symposium*, pages 120–136. 2009.

[34] Daniel Grund and Jan Reineke. Precise and efficient FIFO-replacement analysis based on static phase detection. In *Euromicro Conference on Real-Time Systems*, pages 155–164, 2010.

[35] Daniel Grund and Jan Reineke. Toward precise PLRU cache analysis. In *International Workshop on Worst-Case Execution Time Analysis*, pages 23–35, 2010.

[36] Daniel Grund, Jan Reineke, and Gernot Gebhard. Branch target buffers: WCET analysis framework and timing predictability. *Journal of Systems Architecture*, 57(6):625–637, 2011.

[37] Jan Gustafsson, Adam Betts, Andreas Ermedahl, and Björn Lisper. The mälardalen WCET benchmarks: Past, present and future. In *International Workshop on Worst-Case Execution Time Analysis*, pages 136–146, 2010.

[38] Jan Gustafsson, Andreas Ermedahl, Christer Sandberg, and Bjorn Lisper. Automatic derivation of loop bounds and infeasible paths for WCET analysis using abstract execution. In *Real-Time Systems Symposium*, pages 57–66, 2006.

[39] Damien Hardy, Thomas Piquet, and Isabelle Puaut. Using bypass to tighten WCET estimates for multi-core processors with shared instruction caches. In *IEEE Real-time Systems Symposium*, pages 68–77, 2009.

[40] Damien Hardy and Isabelle Puaut. WCET analysis of multi-level non-inclusive set-associative instruction caches. In *IEEE Real-time Systems Symposium*, pages 456–466, 2008.

[41] Damien Hardy and Isabelle Puaut. WCET analysis of instruction cache hierarchies. *Journal of Systems Architecture*, 57(7):677–694, 2011.

[42] Christopher Healy, Mikael Sjödin, Viresh Rustagi, David Whalley, and Robert Van Engelen. Supporting timing analysis by automatic bounding of loop iterations. *Real-Time Systems*, 18(2-3):129–156, 2000.

[43] Christopher A Healy, Robert D Arnold, Frank Mueller, David B Whalley, and Marion G Harmon. Bounding pipeline and instruction cache performance. *Computers, IEEE Transactions on*, 48(1):53–70, 1999.

[44] Benedikt Huber and Martin Schoeberl. Comparison of implicit path enumeration and model checking based WCET analysis. In *International Workshop on Worst-Case Execution Time Analysis*, 2009.

[45] Bach Khoa Huynh, Lei Ju, and Abhik Roychoudhury. Scope-aware data cache analysis for WCET estimation. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 203–212, 2011.

[46] Ilog, Inc. Solver CPLEX, 2003. `http://www.ilog.fr/products/cplex/`.

[47] Lei Ju, Bach Khoa Huynh, Abhik Roychoudhury, and Samarjit Chakraborty. Performance debugging of Esterel specifications. In *International conference on Hardware/Software codesign and system synthesis*, pages 173–178, 2008.

[48] Timon Kelter, Heiko Falk, Peter Marwedel, Sudipta Chattopadhyay, and Abhik Roychoudhury. Bus-aware multicore WCET analysis through TDMA offset bounds. In *Euromicro Conference on Real-Time Systems*, pages 3–12, 2011.

[49] Hyoseung Kim, Dionisio de Niz, Björn Andersson, Mark Klein, Onur Mutlu, and Ragunathan Raj Rajkumar. Bounding memory interference delay in COTS-based multi-core systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2014.

[50] James C. King. Symbolic execution and program testing. *Commun. ACM*, 19(7):385–394, 1976.

[51] Marc Langenbach, Stephan Thesing, and Reinhold Heckmann. Pipeline modeling for timing analysis. In *Static Analysis Symposium*, pages 294–309, 2002.

[52] Xianfeng Li, Yun Liang, Tulika Mitra, and Abhik Roychoudhury. Chronos: A timing analyzer for embedded software. *Science of Computer Programming*, 2007. `http://www.comp.nus.edu.sg/~rpembed/chronos`.

[53] Xianfeng Li, Tulika Mitra, and Abhik Roychoudhury. Modeling control speculation for timing analysis. *Real-Time Systems*, 29(1):27–58, 2005.

[54] Xianfeng Li, Abhik Roychoudhury, and Tulika Mitra. Modeling out-of-order processors for WCET analysis. *Real-Time Systems*, 34(3):195–227, 2006.

[55] Yau-Tsun Steven Li, Sharad Malik, and Andrew Wolfe. Cache modeling for real-time software: beyond direct mapped instruction caches. In *IEEE Real-time Systems Symposium*, pages 254–263, 1996.

[56] Yun Liang, Huping Ding, Tulika Mitra, Abhik Roychoudhury, Yan Li, and Vivy Suhendra. Timing analysis of concurrent programs running on shared cache multi-cores. *Real-Time Systems*, 48(6):638–680, 2012.

[57] Björn Lisper. Towards parallel programming models for predictability. In *International Workshop on Worst-Case Execution Time Analysis*, pages 48–58, 2012.

[58] Paul Lokuciejewski, Daniel Cordes, Heiko Falk, and Peter Marwedel. A fast and precise static loop analysis based on abstract interpretation, program slicing and polytope models. In *International Symposium on Code Generation and Optimization*, pages 136–146, 2009.

[59] Thomas Lundqvist and Per Stenström. Timing anomalies in dynamically scheduled microprocessors. In *IEEE Real-time Systems Symposium*, pages 12–21, 1999.

[60] Mingsong Lv, Wang Yi, Nan Guan, and Ge Yu. Combining abstract interpretation with model checking for timing analysis of multicore software. In *IEEE Real-time Systems Symposium*, pages 339–349, 2010.

[61] Arian Maghazeh, Unmesh D Bordoloi, Petru Eles, and Zebo Peng. General purpose computing on low-power embedded GPUs: Has it come of age? In *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, 2013.

[62] Renato Mancuso, Roman Dudko, Emiliano Betti, Marco Cesati, Marco Caccamo, and Rodolfo Pellizzoni. Real-time cache management framework for multi-core architectures. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 45–54, 2013.

[63] Marco Paolieri, Eduardo Quiñones, Francisco J. Cazorla, Guillem Bernat, and Mateo Valero. Hardware support for WCET analysis of hard real-time multicore systems. In *International Symposium on Computer Architecture*, pages 57–68, 2009.

[64] Sudeep Pasricha, Nikil Dutt, and Mohamed Ben-Romdhane. Fast exploration of bus-based on-chip communication architectures. In *IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 242–247, 2004.

[65] Rodolfo Pellizzoni, Emiliano Betti, Stanley Bak, Gang Yao, John Criswell, Marco Caccamo, and Russell Kegley. A predictable execution model for COTS-based embedded systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 269–279, 2011.

[66] Rodolfo Pellizzoni and Marco Caccamo. Impact of peripheral-processor interference on WCET analysis of real-time embedded systems. *IEEE Trans. Computers*, 59(3):400–415, 2010.

[67] Jan Reineke and Daniel Grund. Relative competitive analysis of cache replacement policies. In *ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems*, pages 51–60, 2008.

[68] Jan Reineke, Isaac Liu, Hiren D. Patel, Sungjun Kim, and Edward A. Lee. PRET DRAM controller: bank privatization for predictability and temporal isolation. In *International Conference on Hardware/Software Codesign and System Synthesis*, pages 99–108, 2011.

[69] Jakob Rosen, Alexandru Andrei, Petru Eles, and Zebo Peng. Bus access optimization for predictable implementation of real-time applications on multiprocessor systems-on-chip. In *IEEE Real-time Systems Symposium*, pages 49–60, 2007.

[70] Jakob Rosén, C Neikter, Petru Eles, Zebo Peng, Paolo Burgio, and Luca Benini. Bus access design for combined worst and average case execution time optimization of predictable real-time applications on multiprocessor systems-on-chip. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 291–301, 2011.

[71] Erno Salminen, Vesa Lahtinen, Kimmo Kuusilinna, and Timo Hamalainen. Overview of bus-based system-on-chip interconnections. In *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, volume 2, pages II–372, 2002.

[72] Martin Schoeberl, Florian Brandner, Jens Sparsø, and Evangelia Kasapaki. A statically scheduled time-division-multiplexed network-on-chip for real-time systems. In *International Symposium on Networks on Chip*, pages 152–160, 2012.

[73] Andreas Schranzhofer, Jian-Jia Chen, and Lothar Thiele. Timing analysis for TDMA arbitration in resource sharing systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 215–224, 2010.

[74] Rathijit Sen and YN Srikant. WCET estimation for executables in the presence of data caches. In *Proceedings of the 7th ACM & IEEE international conference on Embedded software*, pages 203–212, 2007.

[75] SPIN. SPIN Model Checker, 1991. `http://spinroot.com/spin/whatispin.html`.

[76] Friedhelm Stappert, Andreas Ermedahl, and Jakob Engblom. Efficient longest executable path search for programs with complex flows and pipeline effects. In *International conference on Compilers, architecture, and synthesis for embedded systems*, pages 132–140, 2001.

[77] Vivy Suhendra, Tulika Mitra, Abhik Roychoudhury, and Ting Chen. WCET centric data allocation to scratchpad memory. In *Real-Time Systems Symposium*, pages 10–pp. IEEE, 2005.

[78] Vivy Suhendra, Tulika Mitra, Abhik Roychoudhury, and Ting Chen. Efficient detection and exploitation of infeasible paths for software timing analysis. In *Design Automation Conference*, pages 358–363, 2006.

[79] Vivy Suhendra, Abhik Roychoudhury, and Tulika Mitra. Scratchpad allocation for concurrent embedded software. *ACM Transactions on Programming Languages and Systems*, 32(4):13, 2010.

[80] Henrik Theiling, Christian Ferdinand, and Reinhard Wilhelm. Fast and precise WCET prediction by separated cache and path analyses. *Real-Time Systems*, 18(2/3):157–179, 2000.

[81] Bryan C. Ward, Jonathan L. Herman, Christopher J. Kenna, and James H. Anderson. Making shared caches more predictable on multicore platforms. In *Euromicro Conference on Real-Time Systems*, pages 157–167, 2013.

[82] Reinhard Wilhelm. Why AI + ILP is good for WCET, but MC is not, nor ILP alone. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 309–322, 2004.

[83] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, et al. The worst-case execution-time problemâĂŤoverview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems*, 7(3):36, 2008.

[84] Reinhard Wilhelm, Daniel Grund, Jan Reineke, Marc Schlickling, Markus Pister, and Christian Ferdinand. Memory hierarchies, pipelines, and buses for future architectures in time-critical embedded systems. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(7):966–978, 2009.

[85] Jun Yan and Wei Zhang. WCET analysis for multi-core processors with shared L2 instruction caches. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 80–89, 2008.

[86] Heechul Yun, Renato Mancuso, Zheng-Pei Wu, and Rodolfo Pellizzoni. PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2014.

[87] Heechul Yun, Gang Yao, Rodolfo Pellizzoni, Marco Caccamo, and Lui Sha. Memguard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 55–64, 2013.

[88] Mohamed Zahran, Kursad Albayraktaroglu, and Manoj Franklin. Non-inclusion property in multi-level caches revisited. *International Journal of Computers and Their Applications*, 14(2):99, 2007.