Timing Verification of Undocumented Multicore

Today, almost all computers use multicore processors. Unfortunately, satisfying hard real-time requirements of software executing on such computers is challenging because the timing depends on how resources in the memory system are shared and this information is typically not publicly available. Therefore, this project, "Timing Verification of Undocumented Multicore," has addressed this problem.

Multicore processors. Today, almost all computers use multicore processors. These computers have many processor cores such that one program can execute on one processor core and another program can execute on another processor core simultaneously (true parallelism). Typically, processor cores share memory. In today's memory system, there is a large number of resources that are used to make memory accesses faster in general but unfortunately also make execution time more unpredictable and dependent on execution of other programs (because these other programs use shared resources in the memory system). A simplified view of a multicore processor with the memory system is shown in Figure 1.

Embedded real-time cyber-physical systems. These systems are pervasive in society in general as witnessed by the fact that 99% of all processors produced are used in embedded systems. In many of these systems, computing the correct result is not enough; it is also necessary to compute the correct result at the right time.

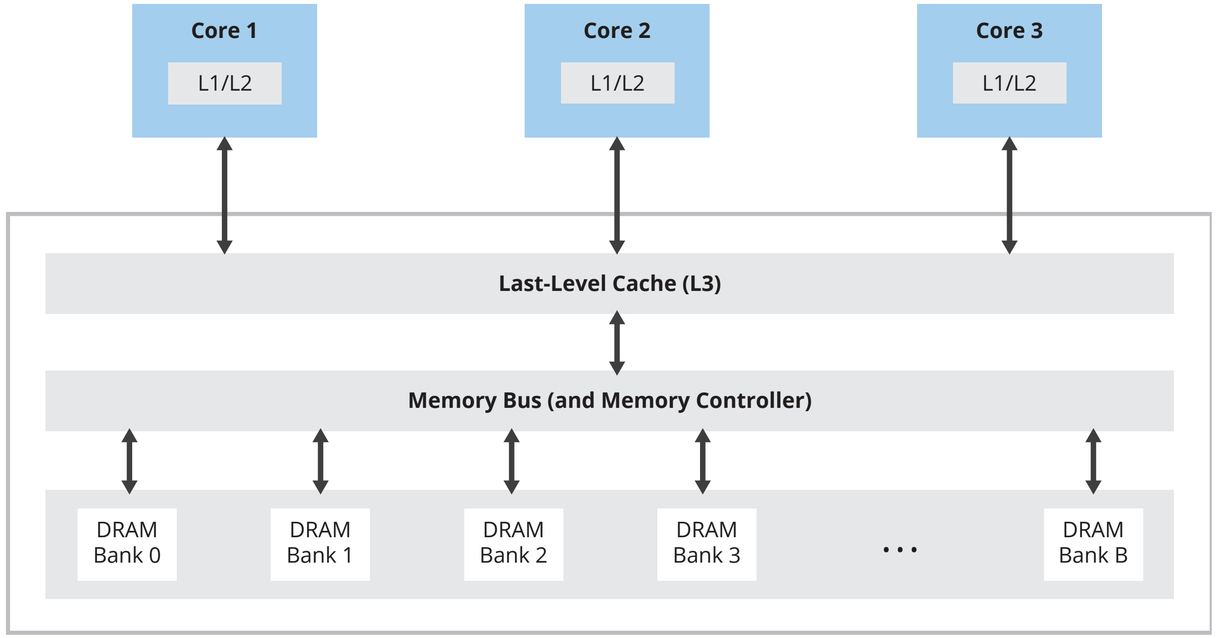

Figure 1: A simplified view of a multicore processor with shared memory.

Figure 2: An example of a DoD system that aims to use multicore processors.
Photo: CPT Peter Smedberg

Department of Defense (DoD). Embedded real-time cyber-physical systems are pervasive in the DoD. An example is shown in Figure 2. Because of the importance of achieving predictable timing, it is common for practitioners to disable all processor cores except one (hence making a multicore processor behave as a single processor system). The importance of timing was recently stressed by AMRDEC's director, stating [1]:

> *"The trick there, when you're processing flight critical information, it has to be a deterministic environment, meaning we know exactly where a piece of data is going to be exactly when we need to—no room for error," Langhout says. "On a multi-core processor there's a lot of sharing going on across the cores, so right now we're not able to do that."*

[1] *"Army still working on multi-core processor for UH-60V,"* May 2017, Available at https://www.flightglobal.com/news/articles/army-still-working-on-multi-core-processor-for-uh-6-436895/

Current solutions. Current state-of-the-art makes solutions available for managing contention for resources in the memory system and for analyzing the impact of this contention on timing. These methods assume that one knows the resources in the memory system; unfortunately, most chip vendors do not make this information available.

Problem addressed. In this project, we have addressed the problem of verifying timing of software executing on a multicore processor assuming that we do not know the resources in the memory system.

Results. We have developed a preliminary method—see B. Andersson et al., "Schedulability Analysis of Tasks with Co-Runner-Dependent Execution Times," ACM Transactions on Embedded Computing Systems, 2018.

Dr. Bjorn Andersson | baandersson@sei.cmu.edu

Distribution Statement A: Approved for Public Release; Distribution is Unlimited

Copyright 2018 Carnegie Mellon University. All Rights Reserved.
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.
The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.
NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.
[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.
Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.
External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission.
Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.
* These restrictions do not apply to U.S. government entities.
Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
DM18-1136