Using Multicore Architectures in Cyber-Physical Systems

Sibin Mohan, Marco Caccamo, Lui Sha,. University of Illinois at Urbana-Champaign [sibin, mcaccamo, lrs]@illinois.edu
Rodolfo Pellizzoni. University of Waterloo [rodolfo.pellizzoni@uwaterloo.ca]
Greg Arundale. Rockwell Collins ATC [gaarunda@rockwellcollins.com]
Russell Kegley. Lockheed Martin [russell.b.kegley@lmco.com]
Dionisio de Niz. Software Engineering Institute [dionisio@sei.cmu.edu]

Introduction

The demand for higher performance computing platforms has dramatically increased during the last decade due to the continuous feature enhancement process. For instance, in automotive systems new safety features like `night view assist' and `automatic emergency breaking' require the fusion of sensor data, video processing and real-time warnings when an obstacle is detected on the road; in the avionics domain new applications such as the helmet-mounted display systems require intensive video processing capabilities. Commercial-Off-The-Shelf (COTS) components are increasingly used in an effort to raise performance and lower production costs. Fast multicore CPUs and high-performance DMA peripherals are needed for servicing these new demanding applications. However, they are difficult to use due to problems with *timing predictability* and *security*. A multicore architecture is substantially different from a single core implementation in that concurrently executing tasks (on the parallel cores) share critical physical resources such as caches, peripherals, on-chip interconnect networks, etc. This extensive sharing of physical resources on critical paths can jeopardize the timing predictability and at times, security, of safety-critical software applications even when system resources are under the control of a real-time operating system. The stark reality is that without addressing these issues, high assurance product developments will be unable to take full advantage of emerging multicore CPUs.

Security and Real-Time Challenges

In safety critical and mission critical systems such as those employed in the automotive, as well as in the avionics and medical domains, it is important to assign applications with different requirements to different partitions with different criticality levels (*viz.* `catastrophic', `hazardous', `major', `minor' and `no effect'). Partitions should be isolated functionally, temporally and securely, *i.e.*, the execution of one partition should not affect tasks in a different partition. Unfortunately, modern COTS architectures are not built to provide strong isolation guarantees. In our opinion, there are three main challenges to timing predictability and security:

1. ***Shared physical resources***: Multicore architectures employ sharing of several physical resources, including caches, main memory, communication infrastructures, power/clock frequency, etc. Resource sharing improves average case execution when software resource requirements are unknown, which is not the case in time-critical systems. On the other hand, extensive hardware resource sharing makes it extremely challenging to provide isolation guarantees; examples are discussed below.
2. ***Multiple active components***: Embedded multicore processors are moving towards more and more integrated platforms where CPU cores, coprocessors, DMA peripherals, etc. *are implemented as a system on a chip (SoC).* All such *active components* compete for access to physical shared resources and thus interfere with each other. All active components must be controlled to ensure safety thus greatly increasing the complexity of the design and certification process. In particular, to the best of our knowledge no existing worst-case execution time (WCET) analysis can simultaneously handle the sharing of all aforementioned physical resources.
3. ***Integrated applications****:* The current reality in embedded real-time systems is that an ever-increasing number of software applications from multiple sources are integrated to run on a decreasing number of computers. This shift from federated to integrated architectures will only accelerate with the move towards processors with multiple (4, 8, and more) CPU cores. System designers can no longer have a full insight into the source code thus requiring *active mechanisms* to handle the reduced trust levels that result.

Example I: cache sharing problem for security. Shared physical resources increase the chance that important information can be leaked to/snooped upon by lower security level tasks. For example, in a single core chip, when a partition loads the last level cache, it can use it until the end of its assigned time slot and then wipe it before yielding the processor to another partition. This ensures that no information is leaked due to the sharing of the cache by multiple tasks on a single core system. This is not possible when multiple cores share the same cache levels, leading to several issues:

1. ***Denial of service attacks***: The less critical partition can deliberately interfere with the execution of high priority tasks by actively evicting cache lines in the shared case and thus increasing the WCET of the high priority task – this could result in the high priority task missing its deadlines. Furthermore, if the system handles overloads in a brittle way (e.g., restarting the whole computer), purposely overrunning a deadline is a way to impose severe damage, since typically the node will be rendered unusable for minutes.
2. ***Covert channels***: Potential covert channels can be created, for example, by transmitting information by modulating one's cache foot-print (or any other shared resource). By monitoring changes in the hit/miss rate for the high-security task in one partition can send information to lower security task in another partition. This can lead to a serious breach of security for the whole system.

Example II: memory bottleneck problem for timing isolation. In a typical COTS-based system, main memory is implemented using one or more banks of single-port dynamic RAM. All active components compete for access to the single-port memory, and the implemented hardware arbitration scheme can significantly affect the delay experienced by each component. In particular, when a task suffers a cache miss, contention for access to main memory can significantly delay cache line fetches and greatly increase the WCET of the task. We performed a set of experiments to understand the severity of this issue. We engineered a task so that it continuously suffers cache misses and measured its WCET in isolation. We then added to the experiment a second copy of the task running on a separate core (using a separate cache) and a PCI-E peripheral using DMA to saturate main memory with write requests and measured the WCET increase of the task. The increase in WCET was as much as 196% for the task, i.e., the *WCET almost tripled*. Simulation results for a system with 4 cores show that a task can suffer a worst case execution time increase of almost 120% when interfering tasks running on other cores spend 18% of their time performing memory accesses. Even worse, simulations with 8 cores show that a task can suffer a worst case execution time increase of more than 300% when interfering tasks spend only 10% of their time performing memory accesses. These problems are highlighted when we hear from our industry colleagues, *e.g.*: "we increasingly hear from developers that they think the current crops of multicore chips with shared cache are so unusable that we should just pass on them as an industry" [1]. As the number of cores grows the inter-core network inside multi-core chips becomes another point of contention. It has been shown that traffic among some cores can block the traffic in and out of other unrelated cores in some inter-core networks (1).

Outlines of Solution Approach

The severity of the described security and real-time challenges demand prompt and concrete actions by the community. The complexity of the design and certification process must be greatly reduced to allow widespread adoption of multicore platforms in cyber-physical systems with safety critical requirements. A paradigm shift in the approach to the problems is required along several directions:

1. ***Hardware-aware OS solutions***: Application portability across different architectures is clearly required in the automotive market. At the same time, the software platform must control all active components in the system and predictably allocate physical resources. Therefore, control mechanisms must be implemented at the OS/hypervisor level. An increased attention to research in hardware-aware OS mechanisms is required.
2. ***Hardware isolation mechanisms****:* We believe that significant improvements in system isolation could be obtained through small architectural modifications that would require very limited device re-engineering and would not significantly affect average performance. SoC integration presents both challenges and

[1] A direct quote from a senior avionics engineer

opportunities. On one hand, it allows manufacturers to implement custom interconnections that can be even harder to analyze than existing systems. On the other hand, it opens up the possibility to insert ``hooks" in hardware resource arbiters to let the designer control system resource allocation. New initiatives in the computer architecture research community to explore OS-driven resource allocation and scheduling mechanisms (1) (2) (3) indicate an increased sensitivity to this matter that should be foster to create multi-core hardware that can support both throughput-oriented and real-time workloads.

3. ***Integrated analyses and tool support****:* Both analyses and control schemes have been proposed for specific types of physically-shared resources. However, there is a lack of integrated early-analysis and evaluation frameworks and tools. In a market dominated by frequent product updates, the designer must be able to evaluate the effect of architectural choices well in advance of any concrete system implementation/integration. Solutions in this arena must take into account the challenges of a design chain distributed across the OEM and its multiple suppliers as well as of the multiple analytical domains included in a CPS (e.g. mechanical, thermal, control, scheduling, etc). A component-based approach that uses libraries of predefined hardware platform components is especially desirable.

One specific research technique we would like to put forward is to create the technology for *Single Core Equivalent (SCE)* software partitions. The key idea is that the execution of a SCE partition on a multicore-based system is *certifiably equivalent* to the execution of the same partition on a single core system. Once we achieve this goal, large numbers of existing certified software applications developed for single core systems can be reused with standard single core (re)certification processes -- a monumental saving in engineering and certification effort. In particular, we believe the community should create challenging and representative model applications to guide research in the following directions: *(1)* isolation technologies for cross-partition timing interference; (*2)* observability management to counter covert channels; *(3)* integrated tool-assisted WCET analysis; *(4)* certification support for safety and security; *(5)* compatibility layers for porting certified single core software; *(6)* application-aware resource partitioning, isolation, coordination and performance optimization and finally *(7)* a characterization of all the sources of platform dependencies that could cause problems in the use of shared resources.

Conclusions

With the increasing demands being placed on automotive and avionics systems the move towards multicore architectures is inevitable. Unfortunately, the current state-of-the-art COTS components available to designers of such systems have serious problems with timing predictability and security. Hence, we believe that the community should come together to *(a)* characterize the true nature of these issues and *(b)* suggest practical solutions that can be adapted by the COTS component manufacturers with minor changes. One such idea (that of *Single Core Equivalence*) is presented in this paper. We believe that it can alleviate many of the problems with predictability and security. But this is just a starting point and we believe that the larger research and industrial communities must work together to find lasting solutions to these critical issues.

Bibliography

1. *Preemptive Virtual Clock: A Flexible, Efficient and Cost-Effective QoS Scheme for Networks on Chip.* **B. Grot, S.W. KEckler, O. Mutlu.** New York : s.n., 2009. 42nd International Sysmposium on Microarchitecture (MICRO). pp. 268-279.

2. *Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors.* **Moscriboda, O. Mutlu and T.** Chicago : s.n., 2007. 40th International Symposium on Microarchitecture (MICRO). pp. 146-158.

3. *Application-AWare Prioritization MEchanisms for On-Chip Networks.* **R. Das, O. Mutlu, T. Moscibroda and C.R. Das.** New York : s.n., 2009. International Symposium on Microarchitecture (MICRO). pp. 280-291.

4. **Administration, Federal Aviation.** *Software Approval Guidelines. FAA Order 8110.49.* s.l. : U.S. Department of Transportation (Federal Aviation Administration)., 2003.

Author Biographies

Lui Sha graduated with Ph.D. from CMU in 1985. He is Donald B Gillies Chair professor of Computer Science in UIUC. In 1998, he was elected as an IEEE Fellow "for technical leadership and research contributions, which enabled the transformation of real-time computing practice". He was elected as ACM Fellow in 2005 "for contributions to real time systems." He is active in dependable real time computing systems research, and served on the National Academy of Science's study committee on certifiably dependable software.

Sibin Mohan completed his Ph.D. from NC State in 2008 during which he analyzed the worst-case behavior of real-time embedded systems that use contemporary processors. He is currently a Research Scientist at UIUC where he works on analyzing system integration issues for complex safety-critical systems, integration of security for cyber-physical systems with real-time constraints and analyzing the bottlenecks for using multicore processors in time-critical systems such as avionics, automobiles, etc.

Marco Caccamo received a Ph.D. in Computer Engineering from Scuola Superiore Sant'Anna, Italy in 2002. He is Associate Professor at Department of Computer Science, University of Illinois at Urbana-Champaign. His research interests include real-time operating systems, real-time scheduling and resource management, wireless real-time networks, and quality of service control in next generation digital infrastructures. He has published over 60 papers and received the US National Science Foundation CAREER Award in 2003.

Rodolfo Pellizzoni obtained his PhD in Computer Science from the University of Illinois at Urbana-Champaign (UIUC) in 2010. While at UIUC he received the W.J. Poppelbaum Memorial Award for outstanding creativity in computer architecture design. In September 2010 he joined the Department of Electrical and Computer Engineering at the University of Waterloo as a new Assistant Professor. Rodolfo's main research interests are in hardware/software co-design and novel architectures for timing predictability and safety certification.

Greg Arundale currently works as a Principal Systems Engineer at Rockwell Collins Advanced Technology Center. He received a B.S. degree in Computer Engineering from UIUC He has over 25 years of experience in Military and Commercial Avionics systems engineering and development, including Surveillance and Reconnaissance systems architectures, Fault Tolerant Computing Architectures, development of the safe Avionics networks such as AFDX, hardware systems lead for the Large Format Display System (LFDS), was a principle contributor to the Common Reusable Element (CoRE) System and the Maintenance Access Terminal (MAT) programs. His early career included 11 years at the Boeing. His current interests include Fault Tolerant Computing, Systems Safety, Hardware and Software Systems Engineering, and Multicore Processing Architectures. Most recently he has led studies of Avionics architectures for next generation systems.

Russell Kegley, a Fellow with Lockheed Martin Aeronautics, has worked in distributed embedded systems since 1978. At Lockheed Martin, he leads contracted and university research activities in schedulability analysis, middleware architectures, multicore processors, as well as contributing to product developments such as the F-22 and F-35 fighter programs.

Dionisio de Niz is a senior member of the technical staff of the Software Engineering Institute at Carnegie Mellon University. He has been leading the research in multi-core real-time scheduling at the SEI. He is also involved in the evolution of the SAE AADL standard. He holds a Computer Engineering Ph.D. from Carnegie Mellon University. His research interest includes real-time multi-core scheduling, mixed-criticality real-time systems, and model-driven development of cyber-physical systems.