Predictable and fault-tolerant multicore architecture

Keywords: heterogeneous multicore, fault tolerance, predictability, mixed-critical systems, energy consumption
Working place : IRISA/INRIA - team CAIRN, Rennes - France https://team.inria.fr/cairn
PhD Director : Olivier Sentieys (olivier.sentieys@inria.fr)
PhD Co-director : Angeliki Kritikakou (Angeliki.Kritikakou@inria.fr)
Responsable DGA : Youri Helen (DGA MI/SDT/MAN/BSA, Rennes, youri.helen@intradef.gouv.fr)

Domain : Computer science, Electrical and Computer Engineer
Deadline for candidate : 10/05/2016

Introduction

The embedded systems from the safety-critical domain industries face exponential **series growth** in terms of **performance requirements**, while they have to deal with strict hard real-time constraints. The safety-critical domains usually deal with applications that have diverse characteristics creating **mixed-critical systems** [1], where high-critical applications run together with low-critical applications. For the highly critical applications, guarantees must be provided in terms of **safety** and **reliability**, which implies a high fault tolerance and at the same time their execution under hard real-time constraints [2]. The notion of the errors and faults is highly linked to the radiation (from natural or technical sources) over the electronic components and if it is not taken into account, it could lead to catastrophic consequences.

The global performance of the system is improved when resources of the platform can be used by the lower criticality applications. Critical embedded applications are present in a wide range of safety-critical domains, such as the transport (avionics, automobile etc), the defense, the nuclear power, and even in the emerging domains, such as medical ones. With respect to the system safety, as the hard real-time constraints of the high-critical applications must be meet, static analysis tools are needed to provide guarantees on their **timing behavior**. It is important that the bounds provided by the tools are tight to avoid the over-provisioning of the resources and assure the determinism of the system. However, these bounds can become very pessimistic because of the unpredictability of the behavior of both the software and the hardware.

The **multicore** architectures, or manycores, provide a massive computing power and offer the integration of diverse applications in the same platform [3]. Replacing the existing infrastructures with multi-core architectures will allow to significantly reduce the overall system cost, energy and its maintenance, while improving the system functionalities due to increased performance capabilities. The use of multi-core architectures will also increase the reliability of the system, as the number of physical components that can fail, e.g. wires and connectors, is reduced.

However, the multicore systems have increased architecture complexity and a dynamic difficult-to-predict behavior, especially due to the schemes which enhance the average performance. This difficulty is incompatible with critical embedded applications (such as in avionics), where it is obligatory to have a deterministic behavior. This becomes valid especially for the COTS - *Commercial-Off-The-Shelf platforms*. The difficulty of predicting the timing performances in this type f architectures derives from the intra-core architectural features, such as the cashes and the branch predictors, but also from the recently inserted inter-core aspects, such as the communications through the interconnection network on chip (NoC), the shared caches and memory controllers, where the concurrent accesses introduce timing variations. To provide timing bounds, either pessimistic Worst Case Execution Times (WCET) [4], or complete spatial and temporal isolation between applications, is applied. The temporal isolation is achieved if the duration of the application execution in one partition is independent from the applications of all

other partitions. The spatial isolation prevents the partitions from accessing the memory or interfaces that are not in their scope. However, the result could be a sub-optimal use of platform resources, if the pessimistic WCET require severe over-provisioning of platform resources to critical applications, and the isolation prohibits the claiming of such over-provisioned resources for the non-critical applications.

Regarding the system reliability, especially when the environment of the system is quite harsh (natural or technical environment with radiation, thunderstorms, high temperatures, etc) and with the reduction of the transistors size, **transient and permanent faults** occur more and more often during the system operation. Hence, the reliability of the system becomes a very crucial characteristic, especially for the critical applications, such as an implanted cardiac defibrillator or the avionics controller for the stability and the flight direction. Currently, to offer a level of reliability, mono-core radiation-hardened (radhard) processors are used, which, however, offer limited computation capabilities. Concerning many/multi-core systems, triplex architectures are used to improve the reliability increasing hardware resources at the cost of an increase in the used resources, the cost and the energy consumption. Hence, further research is required on new efficient fault tolerance techniques for fault detection and fault recovery in comparison to costly triplex architectures, as well as on the impact of fault tolerance on the predictability of the architectures.

The goal of this PhD research is the exploration and the development of novel techniques (especially for the architecture management, the performance estimation and the code generation) and the relevant multicore architectures to establish safety and reliability in many/multi-core platforms exploring the trade-offs between predictability, safety, determinism, performance and energy consumption of the system.

State of the art

With respect to the predictability of the system, the dynamic hardware core components and the shared components among cores are either assumed to always give a worst case performance (which over-approximates the WCET) or ignored (which leads to reduced performance) [5]. The main challenge to build predictable architectures is to design a hardware that supports timing compositionality with low or without performance degradation [6].

Several approaches exist to improve the predictability of the multicore architectures by modifying the components which create contentions. Most of the work has been applied on the on-chip memories, the memory controllers and the shared bus [7,8,9]. Other approaches are focusing on proposing a full customized predicable architecture. FlexPRET [10] is a processor designed specifically for mixed-criticality systems by allowing each task to make a trade-off between hardware-based isolation and efficient processor utilization. If no critical application is scheduled for a cycle, that cycle is used by some lower criticality tasks in a round-robin fashion. Other architectures, such as MERASA [11], ACCROSS [12] or the European project T-CREST [13] focus on multicore designs, including the memory hierarchy or the NoC, in order to make the system more predictable. Other studies propose software approaches for the architecture management, the task management and the supervision of the parallel application executions [14] [15] [16] [17] . The aforementioned approaches are based on design-time pessimistic estimation of the timing performances and are difficult to scale with the increase of the number of cores.

One possibility to improve the performances is to use a hybrid approach, where at design-time upper timing bounds are provided, but at run-time the pessimism is removed by observing and adapting the system behavior based on the information provided by online monitoring of the system. The static analysis of the code [18] can be used to improve the timing estimations for the worst case in the context f the multicore platforms [19], to provide indications to the controller who will take the decisions at run-time using hardware monitors and automatically manage the software monitors. In [20, 21], we have proposed techniques which suspend and restart the tasks with low criticality based on monitoring the real execution time. The decision to suspend the low critical tasks derives from the use of the remaining WCET when the critical tasks are the only tasks executed on the platform.

With respect to the reliability of the system, it is required to define the type of faults under study, identify the spread of fault effects to the system, to detect the faults and to explore fault recovery

mechanisms. Regarding the fault detection, several approaches have been proposed [22] [23] [24], but several open questions still exist, including the overhead of the controller, the number of the memory locations that are checked for a fault and the percentage of the faults detected for mixed-critical systems, which is affected by the criticality of the application, the propagation of an error, the partition of the memory between applications, the fault latency etc. With respect to the recovery mechanisms, when a fault has been detected, the recovery mechanism that takes place can be either retry or fault masking [25]. In the first case, the re-execution of a part of the application from a already known point (which implies a loss in the performance and the need to find a compromise between the number of saving points and the increase in the code size and the required memory), whereas in the second case the application execution is not interrupted when transient faults occur (but it comes with a cost in terms of area, cost and energy consumption). Therefore, exploration is required to find the compromise between the introduced cost due to the fault masking and the loss in performance due to the re-execution of a part of the application, and the impact to the predictability of the performance with the frequency of fault appearance.

Thesis program

Several challenging directions exist in the development of safe and fault tolerant multi-core systems. In this thesis, we are inspired from the conclusion of the analysis of resource contention presented in [5], where it is stated that COTS multicores are not-well suited for executing memory intensive applications in parallel. Therefore, we focus on **mechanisms to manage** the execution on **heterogeneous multicores** which guarantee **improved safety and reliability** of the system, while improving system performance compared to existing approaches.

More precisely, we will explore the extensions of existing multi-core architectures to provide tolerance of transient and permanent faults, and we will explore the timing behavior of these techniques (performance, predictability) under mixed-critical applications. Run-time control mechanisms will be developed to guarantee a level of safety and reliability of the multi-core platform by exploring the trade-offs between the gain in the quality of service provided by the system, the overhead of the techniques proposed for fault tolerance and predictability, the better utilization of the system resources and the percentage of detected ad corrected faults. The trade-offs could drive the creation of complex, but efficient, run-time adaptation strategies for the proposed run-time controller, which will decide which mechanism for fault detection and/or correction should be used during the applications' execution. To achieve this, we will explore a combination of design-time static analysis techniques and run-time hardware or software monitors to capture the information for the performance and the occurring faults during the execution. The safety of the controller should also be studies.

To implement the propose strategies, we will study existing multicore platforms. Our approach will be based, for instance, on using the processor RISC-V, which is an open source architecture in the form of IP block (simulation, synthesis and compiler) with the possibility to add hardware accelerators, specific instructions and interconnection mechanisms allowing a fast but efficient design of a multicore platform [26]. Another hint is the use of Zynq architecture of Xilinx which offers a dual-core system (processor ARM® Cortex™-A9) where specif hardware accelerators can be easily be synthesized and attached to the multicore architecture. Their evolution to the UltraScale+ platforms includes up to 6 cores o processors [27]. Whatever the final choice of the target architecture, it will allow as to develop in a quick way the required prototypes to evaluate the performances of the proposed methods. With the use of the existing IP we can remove the risk during the actual hardware design of multicores.

On top of the proposition of new management techniques for single effects (temporary or permanent faults), the objective of this work is to provide a simulation model and a synthesizable model for an FPGA implementation. These models will include the proposed techniques for fault detection/correction, on which we will execute representative benchmarks of the area under study to test our approaches in a critical environment where faults are injected. The benchmarks will include critical applications, such as applications from the field of vision, the radar processing, the digital communications or flight control from avionics. At the architecture level, the studies will be

carried out specifically on the interconnection NoC, which will be able to offer a guaranteed quality of service and the mechanisms for fault management and priorities, and the memory hierarchy which has to provide a more predictible behavior.

Bibliography

[1] A. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems : Survey of current and emerging trends," in *Proc. 50th IEEE/ACM Design Automation Conference (DAC)*, pp. 1–10, May 2013.

[2] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. B. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem - overview of methods and survey of tools," *ACM Trans. Embedded Comput. Syst.*, vol. 7, no. 3, 2008.

[3] F. Lemonnier, P. Millet, G. Marchesan Almeida, M. Hubner, J. Becker, S. Pillement, O. Sentieys, M. Koedam, S. Sinha, K. Goossens, C. Piguet, M. Morgan, and R. Lemaire, "Towards future adaptive multiprocessor systems-on-chip : an innovative approach for flexible architectures," in *Proc. IEEE International Conference on Embedded Computer Systems : Architectures, Modeling and Simulation (IC-SAMOS)*, (Samos, Greece), July 2012.

[4] M. Gatti, "Development and certification of avionics platforms on multi-core processors," in *Tutorial Mixed-Criticality Systems : Design and Certification Challenges, ESWeek*, (Montreal, Canada), 2013.

[5] S. Vasudevan, A. Easwaran, and R. Klyne, "Profiling COTS multi-cores to quantify shared resource contention," in *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2015.

[6] S. Hahn, J. Reineke, and R. Wilhelm, "Towards compositionality in execution time analysis – definition and challenges," in *Proc. 6th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS)*, December 2013.

[7] S. Wasly and R. Pellizzoni, "A dynamic scratchpad memory unit for predictable real-time embedded systems," in *25th Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 183–192, July 2013.

[8] H. Kim, D. de Niz, B. Andersson, M. Klein, O. Mutlu, and R. Rajkumar, "Bounding memory interference delay in cots-based multi-core systems," in *IEEE 20th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 145–154, April 2014.

[9] M. Paolieri, E. Quiñones, F. J. Cazorla, G. Bernat, and M. Valero, "Hardware support for wcet analysis of hard real-time multicore systems," in *Proc. 36th Annual International Symposium on Computer Architecture (ISCA)*, pp. 57–68, ACM, 2009.

[10] M. Zimmer, D. Broman, C. Shaver, and E. A. Lee, "Flexpret : A processor platform for mixed-criticality systems," Tech. Rep. UCB/EECS-2013-172, EECS Department, University of California, Berkeley, Oct 2013.

[11] M. Paolieri, J. Mische, S. Metzlaff, M. Gerdes, E. Quiñones, S. Uhrig, T. Ungerer, and F. J. Cazorla, "A hard real-time capable multi-core smt processor," *ACM Trans. Embed. Comput. Syst.*, vol. 12, pp. 79 :1–79 :26, Apr. 2013.

[12] C. Salloum, M. Elshuber, O. Hoftberger, H. Isakovic, and A. Wasicek, "The across mpsoc - a new generation of multi-core processors designed for safety-critical embedded systems.," in *Euromicro Conference on Digital System Design (DSD)*, pp. 105–113, 2012.

[13] P. Puschner, D. Prokesch, B. Huber, J. Knoop, S. Hepp, and G. Gebhard, "The T-CREST approach of compiler and WCET-analysis integration," in *IEEE 16th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, pp. 1–8, June 2013.

[14] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha, "Memguard : Memory bandwidth reservation system for efficient performance isolation in multi-core platforms," in *Proc. IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 55–64, 2013.

[15] M. Masmano, I. Ripoll, A. Crespo, and J. Metge, "Xtratum : a hypervisor for safety critical embedded systems," in *Eleventh Real-Time Linux Workshop*, (Dresden, Germany), September 2009.

[16] J. Nowotsch, M. Paulitsch, D. Bühler, H. Theiling, S. Wegener, and M. Schmidt, "Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement," Tech. Rep. 2013-10, University of Augsburg, Germany, 2013.
[17] J. Nowotsch and M. Paulitsch, "Quality of service capabilities for hard real-time applications on multi-core processors," in *Proc. 21st International Conference on Real-Time Networks and Systems (RTNS)*, pp. 151–160, ACM, 2013.
[18] K. N. Parashar, D. Menard, and O. Sentieys, "Accelerated Performance Evaluation of Fixed-Point Systems With Un-Smooth Operations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, pp. 599–612, Apr. 2014.
[19] T. Stripf, O. Oey, T. Bruckschloegla, J. Becker, G. Rauwerda, K. Sunesen, G. Goulas, P. Alefragisc, N. Voros, S., S. Derrien, O. Sentieys, N. Kavvadias, G. Dimitroulakos, K. Masselos, D. Kritharidis, N. Mitas, and T. Perschke, "Compiling scilab to high performance embedded multicore systems," *Microprocessors and Microsystems*, vol. 37, pp. 1033–1049, Nov. 2013.
[20] A. Kritikakou, C. Pagetti, O. Baldellon, M. Roy, and C. Rochange, "Run-time control to increase task parallelism in mixed-critical systems," in *Proc. 26th Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 119–128, 2014.
[21] A. Kritikakou, C. Pagetti, C. Rochange, M. Roy, M. Faugère, S. Girbal, and D. G. Pérez, "Distributed Run-time WCET Controller for Concurrent Critical Tasks in Mixed-critical Systems," in *Proc. 22nd International Conference on Real-Time Networks and Systems (RTNS)*, pp. 139–148, ACM, 2014.
[22] C. Angeli and A. Chatzinikolaou, "On-line fault detection techniques for technical systems : A survey," *International Journal of Computer Science & Applications*, vol. 1, no. 1, pp. 12–30, 2004.
[23] S. M. A. H. Jafri, J. Piestrak, Stanislaw, O. Sentieys, and S. Pillement, "Design of the coarse-grained reconfigurable architecture DART with on-line error detection," *Microprocessors and Microsystems*, vol. 38, pp. 124–136, Mar. 2014.
[24] S. M. A. H. Jafri, S. J. Piestrak, O. Sentieys, and S. Pillement, "Design of a fault-tolerant coarse-grained reconfigurable architecture : A case study," in *Proc. of the 11th IEEE International Symposium on Quality Electronic Design (ISQED 2010)*, (San Diego, CA, USA), p. 6 pages, IEEE, Mar. 2010.
[25] M. Pignol, "DMT and DT2 : two fault-tolerant architectures developed by CNES for COTS-based spacecraft supercomputers," in *12th IEEE International On-Line Testing Symposium (IOLTS)*, pp. 1–10, 2006.
[26] The RISC-V Instruction Set Architecture, http://riscv.org, 2016.
[27] Xilinx All Programmable SoC, http://www.xilinx.com/products/silicon-devices/soc.html, 2016.