Conference Paper

Towards Certifiable Multicore-based Platforms for Avionics

Muhammad Ali Awan

Patrick Meumeu Yomsi

Konstantinos Bletsas

Vincent Nélis

Eduardo Tovar

Pedro Souto

CISTER-TR-150702

2015/04/13

Towards Certifiable Multicore-based Platforms for Avionics

Muhammad Ali Awan, Patrick Meumeu Yomsi, Konstantinos Bletsas, Vincent Nélis, Eduardo Tovar, Pedro Souto

CISTER Research Center

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: muaan@isep.ipp.pt, pamyo@isep.ipp.pt, ksbs@isep.ipp.pt, nelis@isep.ipp.pt, emt@isep.ipp.pt, pfs@fe.up.pt

http://www.cister.isep.ipp.pt

Towards Certifiable Multicore-based Platforms for Avionics

M. Ali Awan*, P. Meumeu Yomsi*, K. Bletsas*, V. Nélis*, E. Tovar*, P. Souto†
*CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Portugal
{muaan, pamyo, ksbs, nelis, emt}@isep.ipp.pt
†University of Porto, FEUP-Faculty of Engineering
pfs@fe.up.pt

Motivation and Challenges

The demand for extra functionality in modern applications is a never ending trend. The hardware vendors are actively improving the design of processors to accommodate these complex applications. The increase in clock speed to enhance the performance of the processor has hit its limits. This is driven by the fact that the performance per watt became costly at high frequencies. Hence, Moore's law is no longer sustained with increasing frequencies but with additional cores [1]. Therefore, in the last decade, the semiconductor industry has experienced a paradigm shift from single processor design to multicore processors (MCP). Cores in MCP share many resources like caches, main memory, I/O devices and interconnects. This sharing, which does not exist in single core processors, makes the temporal behavior of MCPs rather complex and highly unpredictable as these platforms are designed to improve the average-case performance. Consequently, their use in safety-critical applications such as avionics domain is extremely challenging. The certification authorities are very skeptical in the use of MCP platforms in avionics applications.

In May 2014, the North and South American aviation authorities together with their European and Asian counterparts, more specifically the "Certifications Authorities Software Team (CAST)" published a position paper CAST-32 [2] where they express their concerns w.r.t. the use of two-core processors in the implementation of safety-critical avionics systems. Therein, they identify different sources of non-determinism in modern MCPs and suggest guidelines to overcome them.

In order to secure the certifiability of MCPs for avionics, we believe there are three different ways to handle these sources of non-determinism: 1) developing a predictable MCP hardware; 2) providing an extra hardware support to existing MCPs with FPGA(s) (field programmable gate arrays) to circumvent non-deterministic paths and 3) propose software-based mechanisms to mitigate the effect of non-determinism in current MCPs. Although there is a strong industrial drive towards developing hardware-based solutions to the problem, we also firmly believe on the merits of software-based mechanisms being used to mitigate the non-determinism arising from resource sharing in currently available MCPs. Therefore in this paper we focus on some of those software-based mechanisms. In particular, we are exploring timing models that would: (i) accurately incorporate the cache related preemption and migration delays (CRPMD), (ii) mitigate the non-deterministic effect of interconnects, and (iii) consider the interference caused by I/O subsystem on memory, interconnect and caches.

I. CRPMD

Assuming a preemptive and migrative scheduler and MCPs, a premier source of unpredictability stems from sharing caches. This results in an increase in the WCETs of the tasks, unfortunately. We plan to circumvent this issue as follows. First, we strongly believe that by reasoning about the subset of potentially conflicting cache blocks for every pair of preempting/preempted tasks, and bounding the number of preemptions and migrations suffered by each task, we can derive tighter bounds on the WCET for every given scheduling policy. Hence, we opt for such a fully analytical approach for CRPMD estimation [3]–[5]. We will consider two different scheduling approaches: NPSF [6] and C=D [7]. These schedulers are chosen for their ability to achieve high utilization bounds. Upon deriving sound analysis, tweaks to the scheduling algorithms will be needed so that they perform well. One idea to hybridize them is to allow some tasks to migrate only at the job boundary to mitigate the number of migrations. This approach has two advantages: (1) It simplifies the analysis on the one hand and (2) reduces the pessimism on the other. Another idea with potential involves identifying opportunities for deviation from the standard policy at runtime, when it would be safe and provable to reduce CRPMD overheads. Delaying some preemptions/migrations or swapping the task execution order are some ideas in that direction. Finally, on a more practical front, we plan to implement all of this in a real system. Although it is not an architecture for safety critical systems, we plan to target x86Linux for two reasons: to leverage our existing native implementations based on NPSF and C=D and also as a proof of concept. If these techniques work for x86Linux multiprocessors, then we can be confident that the same will apply to predictable platforms.

This work was partially supported by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and when applicable, co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within project UID/CEC/04234/2013 (CISTER Research Centre); also by FCT/MEC and the EU ARTEMIS JU within project(s) ARTEMIS/0001/2013 - JU grant nr.621429 (EMC2) and ARTEMIS/0003/2012 - JU grant nr.333053 (CONCERTO); and also by FCT/MEC and ERDF through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within project FCOMP-01-0124-FEDER-020447 (REGAIN).

II. Interconnects

In MCP system-on-chip, the interconnect is a key component. Its timing behavior impacts virtually any operation at the architectural level. Some efforts have been devoted to derive a timing model for this component in the avionics domain. However, most of the available techniques are based on assumptions which are not met in modern MCPs. Our goal is to develop a model of the CoreNet as used in Freescale's P4080 for example, and more recently, in its P5020 multicores. A risk that can prevent us from reaching this objective may be the non-disclosure of essential information by Freescale [8]. However, we were told by sources in Airbus [9] that Freescale has recently been more forthcoming in providing implementation details. Hence, we expect to have access to the necessary information. As a fall-back plan, we will analyze a wormhole Network-on-Chip (NoC) using virtual channels. NoC is the de facto technology that is meant to become mainstream in future multi/manycore processor chips, because it has been proven through extensive experiments over the last years that any bus topology performs poorly when the number of cores exceeds 8 [10]. Thus any bus-based technology is not a scalable solution. Independently of the interconnect used, we plan to validate the model developed experimentally. To this end, we will use embedded systems benchmarks that are known to be representative of the kinds of applications in avionics [11]. In addition, we will try to obtain from Airbus and Embraer [12] a characterization of novel avionics applications. The novelty of the contribution in this section will be to shift and put the focus on the practicality of the solution proposed. This will inherently make our work different from most works in the state of the art as we will give more importance to the simplicity, practicality, and safety of the solution rather than its efficiency.

III. I/O subsystem

Most safety-critical applications in avionics require the I/O devices to interact with the environment on one front and perform some functions on the other. Among aforementioned resources, I/O devices are also shared among cores on MCPs. The generated traffic from different shared resources (cache, memory, I/O, core) may interfere with each other. For example, CoreNet Coherency Fabric in Freescale P4080 platform connects cores with last-level cache, memory and I/O devices. One of the major issues in such MCP platforms is to develop timing models to tackle the effect of traffic generated from all shared resources. We intend to explore different aspects of the I/O subsystem interference on various shared resources. More precisely, we will address the following two issues.

a) I/O and traffic generated from cores: Most of the peripherals are based on buffered data. In this case, the requests received over a period of time are buffered before initiating the direct memory access (DMA) transfer from the peripheral to the main memory. In this context, a request may not suffer the same delay as the previous one. On the other hand, every core requests to fetch data from the main memory on a cache miss. The size of the requests is usually few bytes (e.g., cache line size, typically 64bytes), so hardware prefetchers sometimes may combine requests for multiple cache lines. However, the size of the request is still small (perhaps two or three cache lines) when compared to a DMA transfer. On top of this, speed of the core is too high when compared to the main memory. Hence, the pipeline is likely to stall when a core retrieves a cache line from main memory. So, the delay suffered by the requests made by a core is cumulative (each request delays the next one), while it is not necessarily the case for peripherals (if there is enough buffer). This should fundamentally change the way the analysis is performed. By considering the difference between buffered and non-buffered traffic, we can achieve better bound in the timing analysis.

b) I/O devices and caches: I/O devices also affect the cache traffic in several ways. For instance, some of the MCP platforms (such as Freescale P4080) allow the DMA units in the peripherals to write directly to cache instead of transferring data to main memory. On one side, this feature (called cache stashing) allows cores to directly read the data from the cache instead of generating any memory requests and loading the contents in the cache. On the other side, this mechanism complicates the cache analysis as designer needs information about the instructions of such transfers. There is a need to develop the timing analysis to cope with this issue by providing a mechanism to differentiate between the DMA transfer bounded to memory or cache. There is a possibility in some MCP platforms to configure some memory as either cache or scratchpad. I think that using scratchpad for allocating I/O data (which has to be managed through a driver, so we have some direct control over buffers and addresses) could be a good way to avoid some of the aforementioned problems (interference, coherency, etc.).

References

[1] D. Geer, "Chip makers turn to multicore processors," Computer, vol. 38, no. 5, pp. 11–13, 2005.

[2] "Certification authorities software team (cast), position paper (cast-32) multicore processors," Certification authorities in North and South America, Europe, and Asia, May 2014.

[3] W. Lunniss, S. Altmeyer, G. Lipari, and R. I. Davis, "Accounting for cache related pre-emption delays in hierarchical scheduling," in 22st RTNS, ser. RTNS '14. ACM, 2014, pp. 183–192.

[4] W. Lunniss, R. I. Davis, C. Maiza, and S. Altmeyer, "Integrating cache related pre-emption delay analysis into edf scheduling," in 19th RTAS, ser. RTAS '13. IEEE Computer Society, 2013, pp. 75–84.

[5] S. Altmeyer, R. Davis, and C. Maiza, "Cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems," in 32nd RTSS, Nov 2011, pp. 261–271.

[6] K. Bletsas and B. Andersson, "Preemption-light multiprocessor scheduling of sporadic tasks with high utilisation bound," in 30th RTSS, Dec 2009, pp. 447–456.

[7] A. Burns, R. Davis, P. Wang, and F. Zhang, "Partitioned edf scheduling for multiprocessors using a c=d task splitting scheme," J. Real–Time Syst., vol. 48, no. 1, pp. 3–33, 2012.

[8] F. Semiconductor. [Online]. Available: http://www.freescale.com

[9] A. c. Airbus. [Online]. Available: http://http://www.airbus.com/

[10] B. Nikolic and S. M. Petters, "Real-time application mapping for many-cores using a limited migrative model," J. Real–Time Syst., 1 2015.

[11] J. Nowotsch, M. Paulitsch, D. Buhler, H. Theiling, S. Wegener, and M. Schmidt, "Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement," in 26th ECRTS, July 2014, pp. 109–118.

[12] E.-E. B. de Aeronautica. [Online]. Available: http://www.embraer.com