MultiPARTES: Multicore virtualization for Mixed-criticality Systems

Salvador Trujillo
IKERLAN, S.coop.
Arrasate-Mondragon, Spain
strujillo@ikerlan.es

Alfons Crespo
Universidad Politécnica de Valencia
Valencia, Spain
acrespo@disca.upv.es

Alejandro Alonso
Universidad Politécnica de Madrid
Madrid, Spain
aalonso@dit.upm.es

***Abstract*— Modern embedded applications typically integrate a multitude of functionalities with potentially different criticality levels into a single system. Without appropriate preconditions, the integration of mixed-criticality subsystems can lead to a significant and potentially unacceptable increase of engineering and certification costs. A promising solution is to incorporate mechanisms that establish multiple partitions with strict temporal and spatial separation between the individual partitions. In this approach, subsystems with different levels of criticality can be placed in different partitions and can be verified and validated in isolation.**

The MultiPARTES FP7 project aims at supporting mixed-criticality integration for embedded systems based on virtualization techniques for heterogeneous multicore processors. A major outcome of the project is the MultiPARTES XtratuM, an open source hypervisor designed as a generic virtualization layer for heterogeneous multicore. MultiPARTES evaluates the developed technology through selected use cases from the offshore wind power, space, visual surveillance, and automotive domains. The impact of MultiPARTES on the targeted domains will be also discussed.

In a number of ongoing research initiatives (e.g., RECOMP, ARAMIS, MultiPARTES, CERTAINTY) mixed-criticality integration is considered in multicore processors. Key challenges are the combination of software virtualization and hardware segregation and the extension of partitioning mechanisms to jointly address significant non-functional requirements (e.g., time, energy and power budgets, adaptivity, reliability, safety, security, volume, weight, etc.) along with development and certification methodology.

I. INTRODUCTION TO MIXED CRITICALITY SYSTEMS

In many domains such as avionics and industrial control there is an increasing trend for mixed-criticality systems, where multiple functions with different criticality and certification assurance levels are integrated using a shared computing platform. Mixed-criticality is the concept of allowing applications at different levels of criticality to interact and co-exist on the same computational platform. Mixed-criticality systems range from lowest assurance requirements up to the highest criticality levels (e.g., DAL A in RTCA DO-178B or SIL4 in EN ISO/IEC 61508). The foundations for this integration are mechanisms for temporal and spatial partitioning, which establish fault containment and the absence of unintended side-effects between functions. Partitions encapsulate resources temporally (e.g., latency, jitter, duration of availability during a scheduled access, etc) and spatially (e.g., prevent a function from altering the code or private data of another partition).

There is a trend towards the widespread use of Multi-Processor Systems-on-a-Chip (MPSoCs) in embedded systems with estimated deployment of 45% in industrial applications by 2015 [Ernst 2010]. It is expected that up to 95% of these MPSoCs will combine cores of mixed-criticality levels. In contrast, the latest versions of processors in industrial applications employ multiple cores, but typically only one core is used nowadays when highly-critical tasks are involved [Mollison 2010]. There is a trend towards adopting mixed-criticality systems in several domains.

II. HARDWARE SUPPORT FOR PARTITIONING

Today several processors provide an Instruction Set Architecture (ISA) with virtualization support. Virtualization is concerned with supporting a system's partitioning and protecting the execution time and memory space of each application. There are a range of hardware properties and components with a big influence on virtualization for embedded systems. For example, Intel, ARM and Freescale introduced special extensions to support virtualization (e.g., Intel VT-x, ARM Trustzone).

In particular, virtualization of multicore processors gains increasing importance as these processors become widely used. The latest versions of processors in industrial applications employ multiple cores, but typically only one core is used when involving highly-critical tasks. Multicore systems present major problems to their adoption in real time systems because of shared resources. Cache, memory accesses, bus arbitration policy, IO, etc. are shared by all cores and introduce a relevant source of indeterminism. Therefore, multicore processors are harder to analyze than single-core processors.

Several processor architectures also support device and Input/Output (I/O) virtualization. An I/O memory management unit (IOMMU) allows guest partitions to directly use peripheral devices through DMA I/O bus and interrupt remapping. Some architectures such as Intel (VT-d), AMD (AMD-Vi) and SPARC V8 (in LEON4 processors) have released their own versions of IOMMU. Also, several hardware extensions for device virtualization improve networking and I/O throughput for virtual machines (e.g.,

`/13 $26.00 © 2013 IEEE
DOI 10.1109/DSD.2013.37

PCI-SIG I/O, network virtualization of Intel VT-c, single-root I/O virtualization SR-IOV). However, the integration of I/O in an architecture with time and space partitioning remaining a research challenge [Shah 2005]. Examples of open issues are the impact of I/O activities on CPU and memory sharing and the safe and seamless communication across different on-chip and off-chip communication networks.

Several virtualization techniques in on-chip and off-chip communication networks are available. Time-triggered networks use time-division multiplexing (TDM) to establish virtual communication links where components cannot interfere with each other in the value and time domain. TTNoC [Paukovits 2008] and AEtheral [Goossens 2010] are examples of time-triggered on-chip networks for time and space partitioning. Examples of time-triggered communication protocols at the cluster level are TTEthenet and FlexRay [Obermaisser 2011]. A major challenge is the seamless virtualization of resources at chip and cluster level. For example, the access to a remote I/O resource located on another chip should be relayed via gateways involving different on-chip and off-chip networks (i.e., vertical integration) and possibly gateways between different types of off-chip networks (i.e., horizontal integration).

III. SOFTWARE SUPPORT FOR PARTITIONING

Software-support for partitioning is realized by hypervisors. Hypervisors are layers of software that exploit the features of the hardware platform to establish independent execution environments. Several virtualization solutions can be observed. These are basically full virtualization, operating system level virtualization and bare metal hypervisors.

- **Full virtualization:** Advantages are that guest operating systems are not modified. Also, both monocore and multicore guest operating systems are supported. Disadvantages are the low performance, the absence of communication mechanisms between partitions, lack of scheduling policies and guarantees for real-time. Examples of products include VMWare Server, Virtual Box and Qemu.
- **Operating system level:** While adequate performance is achieved, no simultaneous execution of multiple operating systems is possible, real-time is not guaranteed and only Linux distributions are supported. Linux VServer, Solaris Zones & Containers, FreeVPS and openVZ are examples of products.
- **Operating system virtualization support:** This virtualization technique offers adequate performance and several guest operating systems can coexist without modification (i.e., full virtualization). An example of this virtualization technique is KVM, which is a solution for full virtualization using the Linux kernel. In general this approach is suitable for soft real-time applications, although there are at present no specific scheduling policies for real-time.
- **Bare metal hypervisor:** A bare metal hypervisor offers good performance and strong isolation of virtual environments. Applications of different levels of criticality can securely coexist and the guest operating systems can be monitored. The small footprint kernel is also easier to validate. A drawback is that guest operating systems have to be modified. Examples of products are Xen, VMWare ESX Server, OKL4 Virtual Logix and INTEGRITY Secure Virtualization.
- **Bare metal Hypervisor for embedded systems:** These real-time oriented hypervisors offer a very low footprint and static allocation of resources. Guest operating systems also have to be modified. Products in this category include XtratuM, PikeOS, RTS Hypervisor, Wind-River VxWorks Integrity-178B LynxSecure, QNX and the Freescale Embedded Hypervisor.

IV. DEVELOPMENT METHODS AND CERTIFICATION

The certification process of safety critical real-time applications, by so-called Certification Authorities (CAs) is based on very conservative assumptions, which typically exceed what is required by the designer's assurance levels. As a consequence, the system designer has to deal with mixed-criticality job sets: high criticality applications (jobs), which have to be certified by CAs, have two different worst case execution times (WCETs): designer's WCET and a more pessimistic (longer) WCET of the CAs.

The pessimistic assumptions of the CAs have to be checked and certified before system start. At the same time, design requirements must also be considered and guaranteed. Considering low and high criticality jobs with CAs' pessimistic assumptions in one schedule lead to an underutilization of the system [Baruah 2010, Rushby 2010].

The problem of scheduling such mixed criticality applications has been shown to be NP-hard for event triggered and time triggered systems [Baruah 2001].

Similar considerations apply for networked systems. TTEthernet or Flexray are examples for integration of messages of various criticality, e.g., time triggered and event triggered. As with processing, the key question becomes one of protection and resource utilization. Analysis for the two mentioned networks so far focused on the critical parts only.

In order to achieve an independent certification of partitions the following aspects should be considered:

- Partition should be executed under spatial and temporal isolation.
- Partitions should be analyzed in an isolated way. Methods for this isolated analysis are required. Sufficient independence may be guaranteed.
- The scheduling of the other partitions should not be modified, causing temporal interference.
- The possible execution interference of multicore execution should be modeled following WCET analysis techniques.

These criteria introduce some constraints with respect to the hardware and the scheduling policy of the virtualization layer. In relation to the hardware, resources shared by the processors (cache, memory, bus arbitration, IO, etc.) shall

not impact or have a limited impact on the partition execution. This could be achieved with specific hardware that permits to isolate or allow a partitioning on the resources.

The criteria related to the scheduling policy implemented by the hypervisor should prevent the variability of the partition execution even preserving its processor bandwidth. A cyclic scheduling policy, as proposed in ARINC-653 standard [ARINC-653] should fulfill this need.

The required development process for safety-critical systems is defined in domain-independent (e.g. IEC 61508) or domain-specific (e.g. ISO 26262, DO-178B) certification standards. These standards require appropriate separation mechanisms for mixed-criticality systems, but do not specify concrete mechanisms or methods to achieve this separation. Multicore and mixed-criticality is a hot topic for certification bodies.

In many cases, the V-Model shape is employed since it covers the entire development progress including requirements engineering, specification, implementation and integration as well as validation and verification (e.g., testing). Therefore, it eases certification according to the mentioned safety standards. However, this needs to be adapted to the mixed-criticality realm.

A plethora of methods are being studied to be applied nowadays to engineer mixed criticality systems. More prominent are those related to Model-Driven Engineering (MDE) and Software Product Lines. There is as well a line of work based on Components.

These techniques promote cross-industry reuse with reduced development cost, shorter time-to-market and higher reliability of mixed-criticality systems.

The development process will leverage the reuse of software and hardware components, safety case reuse, multi-vendor tools interoperability and traceability, and prevent any side effects of component interaction.

A key goal of the project is for academia to listen industry, and for industry to explain openly its needs. An updated roadmap to adoption is expected as an output of the discussion.

Currently a number of related European research projects under both FP7 and Artemis programs are working along similar lines, namely, ACROSS, MultiPARTES, ARAMIS, CERTAINTY, VIRTICAL and RECOMP. They featured some works on both the practical and scientific perspectives, on topics such as reference architecture, methodology, tools, certification issues and industrial application, among others.

A key point is how to assure that techniques are certification ready and certification bodies are well aware of the last progress. In this regard, a hot topic is “sufficient” independence among involved subsystems. Other relevant trends as multicore advent, pressure from consumer electronics, and other topics will be covered.

V. MULTIPARTES PROJECT

The main objective of MultiPARTES is to provide execution environments and tools to support the development of mixed criticality applications over partitioned embedded platforms based on a multicore open source virtualization layer thereby shortening the time-to-market. This will be done through the definition, demonstration and validation of a complete engineering framework supporting the design and development of partitioned systems.

Two main results can be identified:

1. The definition of a methodology to enforce the development and production of new applications based on partitioned multicore systems. The methodology will be supported by a tool that allow the definition of activities and its attributes (security level, criticality level, real-time constraints, operating system needs, hardware dependencies, etc.). Based on these attributes, the tool proposes a partitioning scheme to isolate activities in partitions, an allocation of partitions to cores, and a scheduling scheme to execute the partitions. The final result is a set of configuration files that statically define the behavior of the virtualization layer.
2. An execution platform based on XtratuM [Masmano 2012] hypervisor specifically designed for embedded real-time systems. It provides spatial and temporal isolation of partitions and permits to execute partitions without a specific knowledge of its internals. Partitions can contain different execution environment as single thread (bare partitions). multi-thread (guest RTOS or GPOS) or multicore (multicore guest OS).
3. Execution environments. In the project several execution environment (guestOSs) have been ported: MPTAL (MultiPARTES Abstraction Layer) which is a simple runtime which offers services close to the ARINC-653 P4, ORK+ that provides an Ada environment based on Ravenscar profile [Ureña 2007], Partikle that is a POSIX RTOS [Peiro 2007] and a Linux environment.

VI. MULTIPARTES ARCHITECTURE

XtratuM [Masmano 2010, Crespo 2010] is a hypervisor that uses para-virtualisation technique to build a virtualisation layer. Para-virtualisation is a technique that permits to achieve high performance and low complexity. The para-virtualized model offers potential performance benefits when a guest operating system or application is aware that it is running within a virtualized environment, and has been modified to exploit this. One potential downside of this approach is that such modified guests cannot ever be migrated back to run on physical hardware.

Basically, a hypervisor provides multiple isolated virtual machines. Each VM can execute a complete system (OS kernel and the application processes). Communication between VM’s is done commonly by means of a virtual network. From the application layer, a hypervisor system is much more than a distributed system: a set of computers,

where each computer runs its own operating system and applications, and computers are inter-connected with a high speed network. The most noticeable difference is the speed of the virtual computers, which is only a fraction of the native computer.

The main features provided by the MultiPARTES virtualization layer are:

- Spatial isolation: A partition is completely allocated in a unique address space (code, data, stack). This address space is not accessible by other partitions. The hypervisor has to guarantee the spatial isolation of the partitions. The system architect can relax this property by defining specific shared memory areas between partitions.
- Temporal isolation: A partition is executed under a cyclic policy. The execution of a partitions is not impacted by others. However, shared resources could affect to the execution of a partition. Explicit hardware mechanism could reduce it. Also, the appropriated cyclic plan design could avoid this execution dependency in the more critical tasks..
- Predictability: A partition with real-time constraints has to execute its code in a predictable way. It can be influenced by the underlying layers of software (guest-OS and hypervisor) and by the hardware. From the hypervisor point of view, the predictability applies to the provided services, the operations involved in the partition execution and the interruption management of the partitions.
- Security: All the information in a system (partitioned system) has to be protected against access and modification from unauthorized partitions or unplanned actions. Security implies the definition of a set of elements and mechanisms that permit to establish the system security functions. This property is strongly related with the static resource allocation and a fault model to identify and confine the vulnerabilities of the system.
- Static resource allocation: The system architect is the responsible of the system definition and resource allocation. This system definition is detailed in the configuration file of the system specifying all system resources, namely, number of CPUs, memory layout, peripherals, partitions, the execution plan of each CPU, etc. Each partition has to specify the memory regions, communication ports, temporal requirements and other resources that are needed to execute the partition code. Static resource allocation is the basis of predictability and security of the system. The hypervisor has to guarantee that a partition can access to the allocated resources and deny the requests to other not allocated resources.
- Fault isolation and management: A fundamental issue in critical systems is the fault management. Faults, when occur, have to be detected and handled properly in order to isolate them and avoid the propagation. A fault model to deal with the different types of errors is to be designed. The hypervisor has to implement the fault management model and permits to the partitions to manage those errors that involve the partition execution.
- Partition support: The execution environments require to be adapted to work on a virtual platform. The hypervisor has to provide the support to execute partitions and inform about how the system is working.
- Confidentiality: Partitions cannot access to the space of other partitions neither to see how the system is working. From its point of view, they only can see its own partition. This property can be relaxed to some specific partitions in order to see the status of other partitions or control their execution.

In multi-core platforms, the hypervisor can provide several virtual CPUs to the partitions. A partition can be configured as mono or multi-core. Different partitions (from the number of cores point of view) can coexist in the system. This allows taking profit from a multicore platform even if the partitions are not multicore. Figure 1 shows an example of this view. It shows a multicore platform virtualised by XtratuM which offers the possibility to build multi-core or mono-core partitions. In the example, two partitions use all the virtualised CPUs due to it uses a multi-core OS. The third partition is mono-core and only uses a virtual CPU.

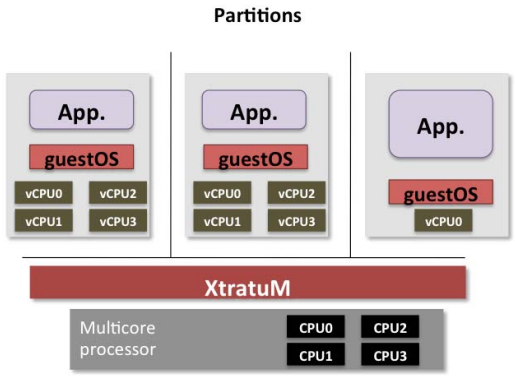

Figure 1 . XtratuM approach

In order to handle the underlying multi-core hardware two software architecture alternatives have been analyzed:

- Asymmetric Multiprocessing (AMP). In this case, one instance of the hypervisor per core executes the partitions allocated to it. As many instances of the hypervisor as core provides the platform run in parallel. Each one of these units executes a different software instance and has exclusive access to its hardware resources. A logical interconnection mechanism is provided in order to communicate/synchronise these logical units. AMP architectures do not permit

parallelism, so no additional exclusion/synchronisation mechanisms are required.

- Symmetric Multi-Processor (SMP) architecture where a single software instance manages all the hardware resources. The software defines execution units (e.g. thread, process) which are dynamically distributed among the hardware processors. SMPs architectures ease the implementation of parallel computing but, on the other hand, more efficient exclusion mechanisms than mutexes must be provided: spin-locks.

In MultiPARTES, the virtualization layer implements a SMP architecture. On the other hand, MultiPARTES deals with heterogeneous multicore systems. The hardware platform consists in two different systems: a dual core x86 based processor and a FPGA with several LEON3 (Sparc V8) synthesized processors. While the x86 subsystem provides high computation capabilities, the LEON3 permits to reach more predictable computation.

XtratuM has been adapted to both platforms and an instance of XtratuM runs in each one. Explicit mechanisms to communicate both instances in different hardware platforms have been implemented. These mechanisms permit to communicate by means of ARINC-653 sampling and queuing ports partitions allocated in different processors.

VII. MULTIPARTES METHODOLOGY

The development of mixed-criticality virtualized multi-core systems poses new challenges that are being subject of active research work. There is an additional complexity: it is now required to identify a set of partitions, and allocate applications to partitions. In this job, a number of issues have to be considered, such as the criticality level of the application, security and dependability requirements, operating system used by the application, time requirements granularity, specific hardware needs, etc.

The MultiPARTES toolset relies on Model Driven Engineering (MDE), which is a suitable approach in this setting, as it facilitates to bridge the gap between design issues and partitioning concerns. MDE is changing the way systems are developed nowadays, reducing development time significantly. In general, modeling approaches have shown their benefits when applied to embedded systems. These benefits have been achieved by fostering reuse with an intensive use of abstractions, or automating the generation of boiler-plate code.

Applications are described using UML, as the standard format. They are enriched with annotations related with extra-functional requirements, such as timing, criticality, security, etc. Additional information can be included, such as requirements on the execution platform (hardware and operating system), and requirements on needed computational resources (CPU time, memory, or bandwidth). For this purpose, it is used UML-MARTE. This information will be used for assessing the fulfillment of some of the non-functional requirements, and for generating a suitable system partitioning.

The MultiPARTES project is developing tool support for partitions identification and applications allocation. The overall approach is shown in figure 2. The initial inputs are:

- Application models: They describe application specific information, such as extra-functional requirements, resource usage needs, etc. In particular, the criticality level of an application should be included in this model.
- Platform models: They include information on the characteristics of the hardware, and the operating systems that can be run on top of XtratuM, for this specific platform.

A system to be developed is initially composed by a set of applications that has to be executed on top of a given platform. The first step is the specification of the partitioning constraints model, which includes information for partitioning that relates the platform and application models. For example, a set of applications that has to be executed on the same partition, an application having specific hardware requirements, etc.

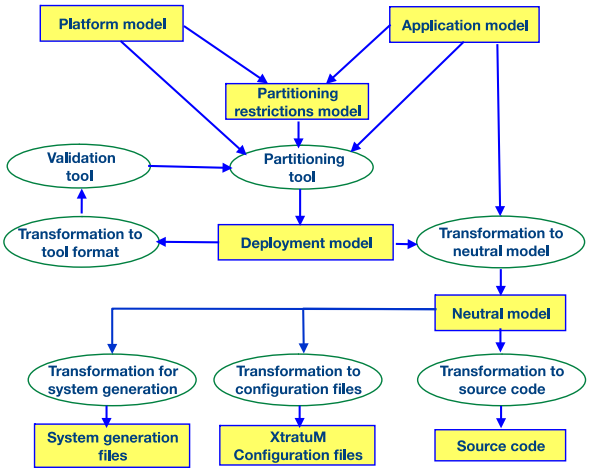

Figure 2 . MultiPARTES toolset

The partitioning tool takes the information from these models as input, for generating a deployment model, which defines a system partitioning: a set of partitions, assignment of hardware resources to each partition, communication ports, partitions security configuration, and the allocation of applications to partitions. The deployment model must meet all system functional and extra-functional requirements. Some of them are correct by construction. For example, the

partitioning tool ensures that two applications with different criticality level will never be in the same partition. However, in some cases analysis tools are required. This is the case with time requirements, as it is needed to ensure that they are met. For this purpose, a response time analysis tool will be used. A model transformation will be in charge of generating the tool input format from the described models. The results of the analysis are fed back to the partitioning tool. If time requirements are met, the generated deployment model is valid. Otherwise, current system partitioning must be refined.

When the deployment model is valid, a neutral model is generated. It includes only the information required for the generation of the final toolset outcomes. It relies mainly on the deployment model and on the applications model. However, in some cases it may also require to use information from the platform model. The aim of the neutral model is to isolate the partition generation activities from the generation of the final files. As it is simpler than the other models, it will be easier to develop model transformations for different programming languages.

The toolset generates three types of final outcomes:

- Source code: the toolset is able to generate Real-Time Java code, or Ada 05 package skeletons.
- XtratuM configuration files: This hypervisor requires a number of files including information about partitions, allocated applications, hardware devices used, and assigned computational resources. These files are automatically generated, taking as main input the information from the deployment model, extracted in the neutral model.
- System generation files: The toolset also generates a set of make files intended to ease the generation of the final system. These files compile and links together the source code, operating systems, and XtratuM, to form the executable that will downloaded to the embedded platform for system execution.

The models and toolset being currently developed will speed-up the integration of mixed-criticality systems based on partitioning and deployed on multicore hardware.

VIII. CONCLUSIONS

This paper presented the approach followed in the MultiPARTES FP7 project to support mixed-criticality integration for embedded systems based on virtualization techniques for heterogeneous multicore processors.

The major outcomes of the project are the methodology to deal with activities with different levels of criticality and the MultiPARTES virtualization layer based on XtratuM, an open source hypervisor designed as a generic virtualization layer for heterogeneous multicore and the methodology.

The partitioning approach and the methodology introduced pave the way to certification. Indeed, this is a major enabler for the adoption of the MultiPARTES approach when engineering mixed-criticality systems.

REFERENCES

[1] [ARINC-653] ARINC: Avionics Application Software Standard Interface ARINC Specification 653-1 (October 2003)

[2] [Baruah 2011] S. Baruah and G. Fohler. Certification-cognizant time-triggered scheduling of mixed-criticality systems. Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Vienna, Austria, 2011.

[3] [Baruah 2010] S. Baruah, H. Li, L. Stougie. Towards the Design of Certifiable Mixed-criticality Systems. IEEE Real-Time and Embedded Technology and Applications Symposium RTAS 2010, Stockholm, Sweden, April 12-15, 2010.

[4] [Crespo 2010] A. Crespo, I. Ripoll, M. Masmano, S. Peiró: Partitioned Embedded Architecture Based on Hypervisor: The XtratuM Approach. Eighth European Dependable Computing Conference, pp: 67-72, Valencia, Spain, 28-30 April 2010.

[5] [Ernst 2010] R. Ernst, Certification of trusted MPSoC platforms, in MPSoC 2010.

[6] [Kopetz 2005] H. Kopetz; A. Ademaj; P. Grillinger; K. Steinhammer The Time-Triggered Ethernet (TTE) Design *8th IEEE International Symposium on Object-oriented Real-time distributed Computing* (Seattle, Washington), May 2005

[7] M. Masmano, I. Ripoll, A. Crespo, S. Peiro. XtratuM for LEON3: an OpenSource Hypervisor for High-Integrity Systems. Embedded Real Time Software and Systems (ERTS2 2010). May 2010. Toulouse (France). 2010

[8] M. Masmano, M. Patte, V. Lefftz, M. Zulianello, A. Crespo, J. Coronel. System impact of distributed multicore systems. In DASIA 2012. DAta Systems In Aerospace. May. Dubrovnik 2012.

[9] [Mollison 2010] M. Mollison and et al. Mixed-criticality real-time scheduling for multicore systems, Proc. of IEEE Int. Conf. on Embedded Software and Systems 2010.

[10] [Peiro 2007] S. Peiro, M. Masmano, I. Ripoll, and A. Crespo. PaRTiKle OS, a replacement of the core of RTLinux. 9th Real-Time Linux Workshop. November 2-4, 2007. Linz, Austria

[11] [Shah 2005] R. Shah, Y.-H. Lee, and D. Kim, "Sharing I/O in strongly partitioned real-time systems," in Embedded Software and Systems, ser. Lecture Notes in Computer Science, Z. Wu, C. Chen, M. Guo, and J. Bu, Eds. Springer Berlin / Heidelberg, 2005, vol. 3605, pp. 502--507.

[12] [Paukovits 2008] C. Paukovits, H. Kopetz. Concepts of Switching in the Time-Triggered Network-on-Chip. Proc. of the 14th IEEE Int. Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2008).

[13] [Goossens 2010]K. Goossens et al., The Aethereal network on chip after ten years: Goals, evolution, lessons, and future. Proc. of Design Automation Conference (DAC), 2010.

[14] [Obermaisser 2011] R. Obermaisser (Ed.): "Time-Triggered Communication"; CRC Press, Taylour & Francis Group. 568 pages, ISBN 9781439846612. 2011.

[15] [Rushby 2011J. Rushby. New Challenges In Certification For Aircraft Software. Proceedings of the Ninth ACM International Conference On Embedded Software (EMSOFT), pp. 211–218, Taipei, Taiwan, October 2011.

[16] [Ureña 2007] S. Ureña, S. Pulido, J.A., Redondo, J., Zamorano, J.: Implementing the new Ada 2005 real-time features on a bare board kernel. Ada Letters XXVII(2), 61–66 (2007); Proceedings of the 13th International Real-Time Ada Workshop (IRTAW 2007).