

Towards Improved Survivability in Safety-Critical Systems

Jaume Abella¹, Francisco J. Cazorla^{1,2},
Eduardo Quiñones¹

¹Barcelona Supercomputing Center (BSC)
²Spanish National Research Council (IIIA-CSIC)
{jaume.abella, francisco.cazorla,
eduardo.quinones}@bsc.es

Arnaud Grasset, Sami Yehia, Philippe Bonnot
Embedded Systems Lab
Thales Research and Technology
{arnaud.grasset, sami.yehia,
philippe.bonnot}@thalesgroup.com

Dimitris Gizopoulos

Department of Informatics and Telecommunications
University of Athens
dgizop@di.uoa.gr

Riccardo Mariani
Yogitech

riccardo.mariani@yogitech.com

Guillem Bernat
Rapita Systems

bernat@rapitasystems.com

Abstract—Performance demand of Critical Real-Time Embedded (CRTE) systems implementing safety-related system features grows at an exponential rate. Only modern semiconductor technologies can satisfy CRTE systems performance needs efficiently. However, those technologies lead to high failure rates, thus lowering survivability of chips to unacceptable levels for CRTE systems.

This paper presents SESACS architecture (Surviving Errors in SAFety-Critical Systems), a paradigm shift in the design of CRTE systems. SESACS is a new system design methodology consisting of three main components: (i) a multicore hardware/firmware platform capable of detecting and diagnosing hardware faults of any type with minimal impact on the worst-case execution time (WCET), recovering quickly from errors, and properly reconfiguring the system so that the resulting system exhibits a predictable and analyzable degradation in WCET; (ii) a set of analysis methods and tools to prove the timing correctness of the reconfigured system; and (iii) a white-box methodology and tools to prove the functional safety of the system and compliance with industry standards.

This new design paradigm will deliver huge benefits to the embedded systems industry for several decades by enabling the use of more cost-effective multicore hardware platforms built on top of modern semiconductor technologies, thereby enabling higher performance, and reducing weight and power dissipation. This new paradigm will further extend the life of embedded systems, therefore, reducing warranty and early replacement costs.

I. INTRODUCTION

The market for critical embedded systems has experienced unprecedented growth over the last ten years and is expected to continue to grow steadily for the foreseeable future. The demand for increased computational power is widespread among key European embedded industries. The number and complexity of the functions to be run in 978-1-4577-1056-8/11/\$26.00 ©2011 IEEE
2011 IEEE 17th International On-Line Testing Symposium

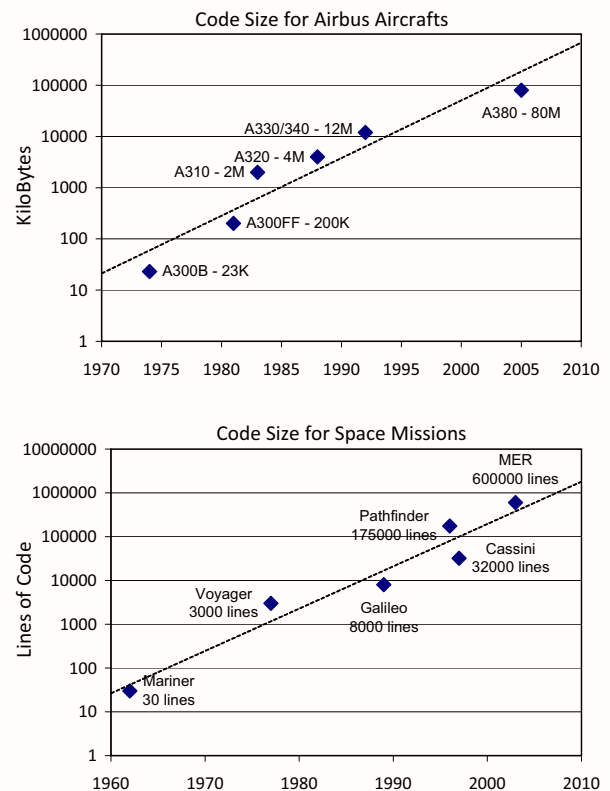


Fig. 1. Code size evolution over time in avionics (top) and space (bottom) applications

the Electronic Control Units (ECUs)¹ have been growing in different market segments including automotive, avionics, railway, medical and space. Figure 1 illustrates this fact for avionics and space in terms of software

¹ECU refers to any embedded system controlling one or more electrical systems, typically in motor vehicles. For the sake of commodity, we use the term ECU for any type of market, not only automotive. Each ECU consists of one or more MicroController Units (MCUs) and other devices (memories, sensors, etc.).

code size [1]. Similar trends have been observed for the other industries. Recently Gartner Inc. has reported that “*semiconductor content of safety systems (in automotive) will almost double, from \$2.2 billion in 2009 to \$4.3 billion in 2014*” [2].

Real-time embedded systems are used to control safety-related functions and thus inherit the safety requirements of the functions they implement. Any misbehavior may produce catastrophic results. Safety functions are complex and require high computational power. For instance, in the last few years all car manufacturers have started to incorporate systems like airbag modules, electronic parking brakes, tracking and stability control, tire-pressure monitoring and x-by-wire technology [3], [4]. Similarly, modern aircrafts require millions of lines of code just for on-board control functions: guidance, navigation, anti-collision systems, etc [5].

There are two ways to deliver the required performance while guaranteeing both the timing and functional correctness of the system: (i) replicating current ECUs that implement the safety functions or (ii) using advanced hardware architectures (multicores and memory hierarchies) and modern semiconductor technology to fabricate chips. The former approach has a huge cost in terms of hardware, power dissipation, size and weight (linear with the increase in performance required). The latter approach enables performance requirements to be satisfied without increasing the number of ECUs; however FIT rates grow significantly for those semiconductor technologies due to their statistical behavior and the operational lifetime decreases beyond the requirements of Critical Real Time Embedded (CRTE) systems. Those systems may not fulfil safety standards requirements such as those of DO-254 (avionics) [6], ISO 26262 (automotive) [7] and IEC 61508 (metastandard) [8]. In the future, incorrect operations of electronics may not only cost industry billions of Euros annually, but also jeopardize the citizens health and well-being by decreasing safety on the road and in the air.

Classic fault-tolerance techniques focus only on providing functional correctness. However, in CRTE systems it is essential to guarantee both the timing and functional correctness of the system. Moreover, WCET analysis used to schedule tasks in CRTE systems assumes fault-free operation (or within safe FIT bounds). To exacerbate the problem, verification/validation costs of CRTE systems cannot be increased significantly as they are currently expensive processes.

The objective of SESACS architecture (Surviving Errors in SAfety-Critical Systems) is providing novel solutions to achieve **tomorrow’s performance with today’s survivability, reliability, cost and power dissipation** using recent and future semiconductor technologies. As

opposed to existing approaches, which are built on top of sufficiently reliable technologies, this new design paradigm assumes the use of recent and future semiconductor technologies that provide a performance level several orders of magnitude higher by enabling multi-cores and cache hierarchies to be implemented. However, those technologies increase performance at the expense of higher raw failure rates, which are unaffordable in CRTE systems. SESACS architecture aims to provide:

- A hardware/firmware (HW/FW) platform able to provide a virtually fault-free hardware layer to the software layers on top. The new HW/FW architecture relies on introducing modifications at the hardware level for prompt fault detection, diagnosis, recovery and reconfiguration (collectively DDDR).
- A timing analysis tool that takes fault impact into account to provide safe and tight WCET estimations.
- Functional safety methods and tools able to consider both the requirements of the safety standards and the challenges of CRTE multicore systems to make certification easier.

The new design paradigm is a cost-feasible alternative path to survivability that will provide the required performance and safety levels, while avoiding expensive replication of units and without a significant increase of verification/validation costs.

The rest of the paper is organized as follows. Section II describes the rationale behind the new paradigm and its potential impact on performance and survivability. Section III presents the details of the SESACS architecture. Section IV discusses related work and gives some insights to build systems compliant with the new paradigm. Finally, Section V concludes this paper.

II. RATIONALE BEHIND SESACS

The objective of the new design paradigm is to enable the use of recent and future semiconductor technologies in safety-critical systems providing the same survivability and fault-tolerance as mature technologies. This fact is illustrated in Figure 2. The figures of FIT rates over time are based on the NASA’s JPL (Jet Propulsion Laboratory) estimations [9]. After burn-in testing in the fab, FIT rates remain constant until they rise abruptly due to degradation at the end of the chip lifetime. For very mature semiconductor technologies such as 180nm, expected life before wear-out is typically of the order of 100’s to 1000’s of years; however, for more recent technology nodes (i.e. 65nm) it is expected to be below 5 years for the highest safety-integrity levels of standards. This means that chips built using those technologies are not suitable for the demands of industries with long life cycles like automotive, avionics, railways, medical

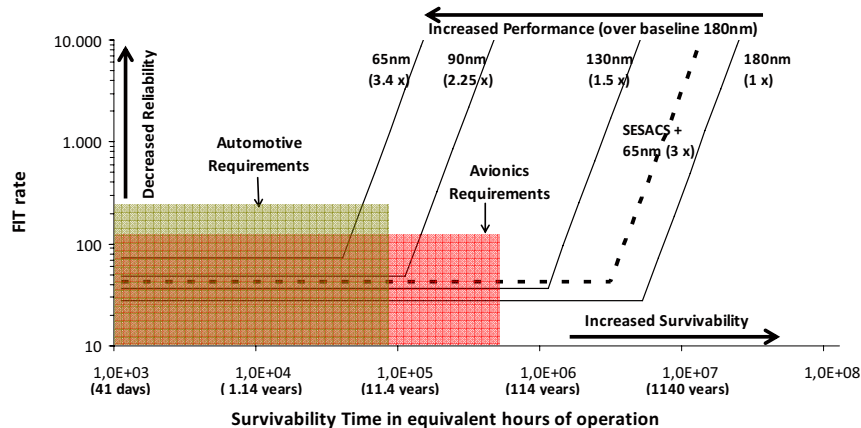


Fig. 2. Trends for FIT rates of semiconductor technologies (65nm, 90nm, 130nm and 180nm) over the chips lifetime after burn-in testing in the fab. The X and Y axes show equivalent hours of operation and FIT rates respectively in logarithmic scale.

and space. Shaded boxes indicate estimated FIT rate requirements following guidelines set by standards for automotive and avionics and assuming expected lifetimes of 10 and 50 years, respectively. Recent semiconductor technology fails to achieve both: low FIT rates (65nm is outside the limits for avionics) and long enough lifetimes (65nm and 90nm exhibit degradation and increased FIT rates long before 10 and 50 years of operation, respectively). MCUs based on the new SESACS design paradigm (dashed line in Figure 2) will cause a small degradation in performance using recent and future technology nodes but achieving high levels of reliability and survivability (i.e. around those of 130nm). Dashed line is an estimate of the FIT levels that SESACS technology will achieve, with only a small performance degradation (less than 10% per technology generation), which will be largely offset by the performance gains obtained with advanced CMOS technologies.

III. SESACS ARCHITECTURE

SESACS, the new design paradigm, consists of three main components (shown in Figure 3): the HW/FW architecture, the timing analysis tool and the functional safety analysis tool. The following subsections elaborate on the design of those components.

A. HW/FW Architecture

The HW/FW architecture consists of a set of coordinated DRR (detection, diagnosis, recovery and reconfiguration) mechanisms [10] built on top of a multicore architecture that meets the increasing integration requirements of CRTE systems. Fault detection techniques identify faults that may corrupt the state of the system in such a way that either functionality or timing of the application being run can be affected. Such detection techniques must be quick enough to prevent the propagation of corrupted data and, even more important, time-bounded so that WCET analysis methods can properly

consider the impact of fault detection on the WCET. HW/FW detection techniques must have high coverage and low HW cost. Thresholds for coverage and cost will depend on the needs of the particular market. Different levels must be defined for each target safety level so that different subsets of solutions can provide the reliability needed at the lowest possible cost. Moreover, intellectual property (IP) intrusion (modifications to HW IP) must be low and detection techniques must be transparent for the SW layers.

Once faults have been detected, they are diagnosed and characterized following the same philosophy: quick and time-bounded fault diagnosis, high coverage, low HW cost, low IP intrusion and transparently for the SW layers. Diagnosis techniques identify whether the fault is transient or permanent, which components have been affected and how severe the fault is for the architectural state of the system. Based on such information, recovery is performed to reach a valid state of the system so that execution can be resumed following the same philosophy as for detection and diagnosis (with low HW cost and in a short interval). Finally, if the fault was permanent, HW is reconfigured (potentially degraded) so that it can continue operating without experiencing any further error manifestation due to the detected fault. Once reconfigured, HW must offer a degraded but time-predictable behavior and gradual WCET degradation with respect to further faults. Two main issues must be considered for effective reconfiguration: (i) making the reconfiguration process time-bounded and (ii) providing predictable performance for the reconfigured system.

The design of HW/FW architectures compliant with the new design paradigm requires identifying which particular components must be protected and to what extent (granularity) based on the safety standards requirements for the markets being addressed.

HW/FW time-predictable DRR solutions must be provided for both on-chip components (e.g., processor

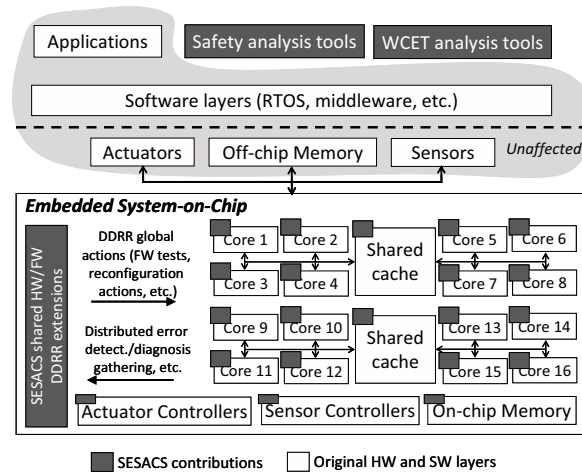


Fig. 3. SESACS platform schematic

cores, shared cache memories, on-chip interconnects, etc.) as well as off-chip ones (e.g., main memory, sensors, etc.). While on-chip DDRR techniques can be suited as needed (although trying to keep IP modifications to a minimum), off-chip components cannot be modified. Thus, DDRR techniques for off-chip components will make use of fault-tolerance features of those components and will be implemented inside the MCU to reduce the HW cost of the platform.

Different types of DDRR techniques can be used in SESACS [10]. The main challenge is choosing the set of solutions that provides the coverage required by the highest safety levels in the standards at low cost (power, area and delay), with low and bounded impact in WCET, and with low IP intrusion. This requires trading off between centralized and distributed components as depicted in Figure 3.

B. Timing Analysis Tool

The precise quantification of performance degradation over time under different numbers and types of faults is a key issue. For example, the probability of a fault hitting the cache is much higher than that of a fault hitting an ALU, however their timing impact is vastly different. One faulty cache line would result in those memory locations that hit that cache line experiencing additional delays (if the reconfiguration results in having to fetch it always from main memory, or re-routed to a spare cache line), however the reconfiguration of a faulty ALU may result in only one ALU being available (assuming that there were initially two), in this case, almost all instructions in the program can have an additional delay as they all compete for the same ALU for data operations and address calculations.

The timing analysis tool must quantify the sensitivity of the system to faults as a function of probability of the fault occurring, and the impact on the WCET of the reconfiguration required. The resulting method must

enable the computation of a distribution of execution times under different fault models, number of faults and set of DDRR techniques. Indeed, this results in moving away from considering the WCET as a single number, as it has been the common practice, but rather as a function of time (meaning lifetime), or as a function of the number of faults or fault-rates for any given set of HW/FW DDRR techniques. Given that those methods can be computationally very expensive, an appropriate trade-off between the precision of the results and the computational complexity must be achieved.

In addition, the results of WCET analysis can only be used in the context of certification if they can be trusted. This requires proof that the methods used are sound and that the tool that implements those methods has been correctly constructed. The methods implemented in the timing analysis tool have to be rigorously proven to support the tool qualification process.

C. Safety Analysis Tool

This tool is responsible for guaranteeing that the system provides the functional safety levels required by standards such as ISO 26262 [7] (for automotive hardware), DO-254 [6] (for avionics hardware) and IEC 61508 [8] metastandard. Different safety levels must be considered since the highest safety level is not always required. In particular, safety standards define several safety levels depending on the potential impact of faults, being ASIL D for ISO 26262, DAL-A for DO-254, and SIL 3 / SIL 4 for IEC 61508 the most stringent levels.

The tool must follow a methodology so that based on the description of the HW/FW designs, the fault simulation and injection methods and the simulation/emulation results, it produces the measures required. Those measures must describe the type of faults observed, the coverage of the HW/FW DDRR techniques within the timing bounds dictated by the timing analysis and the impact of faults in line with safety standards definitions.

Note that impact of faults in this context refers to the impact of those faults escaping HW/FW DRRR mechanisms and/or timing bounds. The safety analysis tool must quantify the probability of faults to happen (either functional or timing faults) and the severity of those faults to determine the safety level for which the system is compliant.

IV. STATE-OF-THE-ART AND NEW DIRECTIONS

A. *State-of-the-art on HW/FW DRRR*

Existing DRRR techniques focus on the functional correctness of systems [10]; however, most of them are unsuitable for future CRTE systems because they are unable to be used in environments where safe and tight WCET estimations are needed within stringent cost constraints.

For instance, techniques based on full redundancy [11], [12], partial redundancy [13], [14] and coding [12], [15] fail to provide survivability (the system is not usable after the first permanent fault) and may have unaffordable costs for many safety market segments. Approaches based on redundant code execution do not work for permanent faults and cause very large WCET degradation [16]. An alternative for permanent faults is using periodic on-line self-test [17]–[19] and/or performing on-line error diagnosis [20], [21]. However, those techniques do not provide timing guarantees and it is unclear how they can diagnose degraded systems. Existing recovery techniques based on checkpointing [22] also fail to provide timing guarantees. Reconfiguration techniques based on HW replacement have excessive cost for most of the components as it is the case for HW redundancy techniques [23]. On the other hand, approaches based on disabling faulty HW fail to provide timing guarantees for the reconfigured system [24]–[26].

Some recent works provide some degree of time predictability in the presence of cache faults [25], [27], but not WCET guarantees. To the best of our knowledge, only the Reliable Victim Cache and its extension [28], [29] provide WCET guarantees at low cost for cache reconfiguration in the presence of faults, thus facilitating WCET analysis.

B. *State-of-the-art on Timing Analysis*

There are various approaches to perform timing analysis and in particular WCET analysis of CRTE systems [30]. Industry generally follows a measurement-based approach in which the system is observed (usually at the task level) under test conditions, measurements are taken and a posteriori analysis is performed. An alternative approach is to use static timing analysis; this relies on a very precise timing model of the hardware and the analysis of the high level structure of binary

code. An intermediate hybrid approach represented by the RapiTime tool [31] uses direct measurements from the programs running on real-targets instead of a model of the processor, and static analysis of the high-level structure of the code.

To the best of our knowledge there are no works that study the WCET of programs under fault conditions, in particular the case where machine instructions can have different timings due to faulty operation of the hardware. Current WCET analysis techniques assume fault-free operation of the system and are completely unaware of the timing impact of transparent error detection, diagnosis, recovery and reconfiguration of the hardware. It is currently an open problem what is the WCET of a program under fault conditions, mainly because where and when these faults occur is unknown. The problem is in general computationally intractable with existing tools even for a small number of faults as it would require calculating the WCET of all combination of possible faults, and only on HW architectures where the timing impact of such faults could be quantified. Note that it is a widely known issue that is reported in the literature that small changes in a program can have significant impact on the WCET [32].

C. *State-of-the-art on Safety Analysis*

One of the key problems in using state-of-the-art functional safety analysis methodologies and tools for CRTE systems is that they do not have specific means to verify the behavior of hardware architectures in the presence of faults leading to timing failures. The current approach consists of performing a massive random fault injection and afterwards a very time-consuming analysis to identify the faults that may have caused a timing failure [33]. Some preliminary attempts have been done recently to provide new methodologies considering timing correctness [34].

D. *New Directions for the SESACS Paradigm*

Techniques providing time-predictable and low-latency fault DRRR are needed for on-chip and off-chip structures. Some existing designs are particularly suitable to be adapted to the needs of the SESACS paradigm. The most promising paths to follow for fault detection and diagnosis are HW partial redundancy [13], [14] and FW periodic on-line self-tests [17]–[19] due to their low IP modifications, although they do not provide timing guarantees. Recovery approaches will strongly depend on the latency between fault occurrence and its detection/diagnosis so no particular directions can be given at this point. Regarding reconfiguration, some effective and low cost solutions have been already proposed for particular components [28], [29], but most of the components lack still of effective solutions.

Timing analysis under fault conditions has not been reported yet, although RapiTime [31] is a good starting point due to its flexibility and given that it is currently being extended towards probabilistic timing analysis, which will be of much use under faults. Finally, new safety analysis methodologies are needed to consider the impact of faults in timing. Initial attempts show promising results [34], but they require some significant effort to address complex ECUs.

V. CONCLUSIONS

Increasing performance needs of Critical Real-Time Embedded (CRTE) systems require the use of recent and future semiconductor technologies. However, those technologies offer survivability levels largely below CRTE systems needs. Therefore, new paradigms are required for CRTE system design on such a new environment.

SESACS is a new design paradigm addressing those issues introduced by inherently-faulty modern semiconductor technologies. The new design paradigm combines a HW/FW fault-tolerant architecture and timing and safety analysis tools to guarantee functional and timing correctness satisfying the most stringent constraints of functional safety standards such as DO-254 and ISO 26262. The new design paradigm shows how components must be designed to achieve **tomorrow's performance with today's survivability, reliability, cost and power dissipation.**

ACKNOWLEDGMENTS

Jaume Abella, Francisco J. Cazorla and Eduardo Quiñones have been partially supported by the Spanish Ministry of Education and Science under grant TIN2007-60625 and HiPEAC. Jaume and Eduardo have also been supported by the Generalitat of Catalunya under grant Beatriu Pinós 2009 BP-B 00260 and the Spanish Ministry of Science and Innovation under the grant Juan de la Cierva JCI-2009-05455.

REFERENCES

- [1] G. Edelin, "Embedded systems at THALES: the artemis challenges for an industrial group," in *presentation at the ARTIST Summer School in Europe*, 2009.
- [2] P. Clarke, "Automotive chip content growing fast, says Gartner," <http://www.eetimes.com/electronics-news/4207377/Automotive-chip-content-growing-fast>, 2010.
- [3] E. Bretz, "By-wire cars turn the corner," *IEEE Spectrum*, vol. 38, 2001.
- [4] G. Leen and D. Heffernan, "Expanding automotive electronic systems," *IEEE Computer*, vol. 35, 2002.
- [5] European Organisation for the Safety of Air Navigation (EUROCONTROL), *Study Report on Avionics Systems for the Time Frame 2007, 2011 and 2020*.
- [6] RTCA and EUROCAE, *DO-254 / ED-80, Design Assurance Guidance for Airborne Electronic Hardware*, 2000.
- [7] International Organization for Standardization, *ISO/FDIS 26262*, 2011.
- [8] International Electrotechnical Commission, *IEC 61508 Edition 2.0*, 2009.
- [9] S. Guertin and M. White, "CMOS reliability challenges the future of commercial digital electronics and NASA," in *NEPP Electronic Technology Workshop*, 2010.
- [10] D. Gizopoulos, M. Psarakis, S. Adve, P. Ramachandran, S. Hari, D. Sorin, A. Meixner, A. Biswas, and X. Vera, "Architectures for online error detection and recovery in multicore processors," in *DATE*, 2011.
- [11] A. Avizienis and J. Kelly, "Fault tolerance by design diversity: Concepts and experiments," *IEEE Computer*, vol. 17, 1984.
- [12] I. Koren and C. Krishna, *Fault-Tolerant Systems*. Morgan Kaufmann, 2007.
- [13] R. Mariani, P. Fuhrmann, and B. Vittorelli, "Cost-effective approach to error detection for an embedded automotive platform," in *SAE World Congress*, 2006.
- [14] R. Mariani, M. Baumeister, and P. Fuhrmann, "A single channel, fail-safe microcontroller to simplify SIL3 safety architectures in automotive applications," in *ESV-VDI*, 2007.
- [15] S. Lin and D. Costello, *Error Control Coding (2nd Edition)*. Prentice Hall, 2004.
- [16] S. Reinhardt and S. Mukherjee, "Transient fault detection via simultaneous multithreading," in *ISCA*, 2000.
- [17] H. Al-Asaad, B. Murray, and J. Hayes, "Online BIST for embedded systems," *IEEE Design and Test*, vol. 15, 1998.
- [18] M. Psarakis, D. Gizopoulos, E. Sanchez, and M. Sonza-Reorda, "Microprocessors software-based self-testing," *IEEE Design and Test of Computers*, vol. 27, 2010.
- [19] D. Gizopoulos, M. Psarakis, M. Hatzimihail, M. Maniatakos, A. Paschalis, A. Raghunathan, and S. Ravi, "Systematic software-based self-test for pipelined processors," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 16, 2008.
- [20] F. Bower, D. Sorin, and S. Ozev, "A mechanism for online diagnosis of hard faults in microprocessors," in *MICRO*, 2005.
- [21] J. Carretero, X. Vera, J. Abella, T. Ramirez, M. Monchiero, and A. Gonzalez, "Hardware/software-based diagnosis of load-store queues using expandable activity logs," in *HPCA*, 2011.
- [22] D. Sorin, M. Martin, M. Hill, and D. Wood, "Safetynet: Improving the availability of shared memory multiprocessors with global checkpoint/recovery," in *ISCA*, 2002.
- [23] F. Bower, P. Shealy, S. Ozev, and D. Sorin, "Tolerating hard faults in microprocessor array structures," in *DSN*, 2004.
- [24] P. Shivakumar, S. Keckler, C. Moore, and D. Burger, "Exploiting microarchitectural redundancy for defect tolerance," in *ICCD*, 2003.
- [25] J. Abella et al., "Low vccmin fault-tolerant cache with highly predictable performance," in *MICRO*, 2009.
- [26] C. McNairy and J. Mayfield, "Montecito error protection and mitigation," in *Workshop on High Performance Computing Reliability Issues (HPCRI)*, 2005.
- [27] N. Ladas, Y. Sazeides, and V. Desmet, "Performance-effective operation below Vcc-min," in *JSPASS*, 2010.
- [28] J. Abella, E. Quinones, F. Cazorla, Y. Sazeides, and M. Valero, "RVC: A mechanism for time-analyzable real-time processors with faulty caches," in *HiPEAC*, 2011.
- [29] J. Abella, E. Quinones, F. Cazorla, Y. Sazeides, and M. Valero, "RVC-Based time-predictable faulty caches for safety-critical systems," in *IOLTS*, 2011.
- [30] Wilhelm R. et al., "The worst-case execution-time problem: overview of methods and survey of tools," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, 2008.
- [31] Rapita Systems, Ltd., *RapiTime: Worst-case execution time analysis. User Guide*, 2007. [Online]. Available: <http://www.rapitasystems.com>
- [32] E. Mezzetti, N. Holsti, A. Colin, G. Bernat, and T. Vardanega, "Attacking the sources of unpredictability in the instruction cache behavior," in *RTNS*, 2008.
- [33] A. Benso, A. Bosio, S. D. Carlo, and R. Mariani, "A functional verification based fault injection environment," in *DFT*, 2007.
- [34] M. Paolieri and R. Mariani, "Towards functional-safe timing-dependable real-time architectures," in *IOLTS*, 2011.