2008 International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems

Register File Reliability Analysis Through Cycle-Accurate Thermal Emulation

José L. Ayala[1], Pablo G. Del Valle[1], David Atienza[2]
[1]Departamento de Arquitectura de Computadores y Automática
Universidad Complutense de Madrid (Spain)
Email: {jayala,pgarcia}@fdi.ucm.es
[2]Embedded Systems Laboratory
Ecole Polytechnique Fédérale de Lausanne (Switzerland)
Email: david.atienza@epfl.ch

***Abstract*—Continuous transistor scaling due to improvements in CMOS devices and manufacturing technologies is increasing processor power densities and temperatures; thus, creating challenges when trying to maintain manufacturing yield rates and devices which will be reliable throughout their lifetime. New microarchitectures require new reliability-aware design methods that can face these challenges without significantly increasing cost and performance. In this paper we present a complete analysis of reliability for the register file architecture of the Leon 3 processor. The analysis conducted is supported by the use of an accurate HW/SW FPGA-based emulation platform that enables a complete design space exploration of thermal and reliability metrics during the execution of an extended set of benchmarks, in a very limited amount of time. The effect of various compiler optimizations and register assignments on the reliability of the register file is then analyzed. Our results quantify the respective effects of these different factors and enable us to design a reliability-aware register file assignment policy that consistently improves the Mean-Time-To-Failure figure (20% on average) for the various types of applications.**

I. Introduction

The relentless scaling of technology and increase in transistor densities are a primary reason for Multi-Processor System-on-Chips (MPSoCs) to have become possible [1]. However, power requirements have not scaled accordingly, causing power densities to skyrocket and on-chip temperatures to increase at alarming rates [2]. As a result, the International Technology Roadmap for Semiconductors (ITRS) [3] has predicted that traditional design constraints centered around product cost and performance requirements will soon be overtaken by processor wear-out and lifetime reliability issues [4].

While traditionally such issues have been left up to device and process engineers, the ability to model and evaluate reliability effects such as electromigration, stress migration, time-dependent dielectric breakdown, and thermal cycling at the microarchitectural level has been shown to be critical to improving processor lifetime [4]. However, acquiring realistic temperature and reliability estimations at the microarchitectural level has proven to be very complex because cycle-accurate MPSoC simulators [5], [6] need to run for millions of cycles to accurately characterize the systems switching activity and behavior. Because of these reasons, thermal and reliability analysis at a high level of accuracy has typically been extremely time consuming and inefficient in the past.

One of the key structures to model in the reliability analysis of MPSoCs is the register file of processing cores. The multiported SRAMs used to implement it are prone to shorter lifetimes and failures due to two primary reasons. First, the register file consumes a substantial amount of power within modern microprocessors [7], which produces a very high power density due to its relatively small size and makes it an important hotspot within the chip [8]. Second, the large amount of read/write accesses produces a higher amount of switching activity than other components. Since a large number of reliability effects are induced by switching activity and temperature the register file becomes a critical component for improving the lifetime of a microprocessor [4].

In this work we present complete reliability analysis for the register file found in the Leon 3 processing core using an accurate HW/SW FPGA-based microarchitectural emulator. Our emulation provides fast and accurate reliability analysis of the relation between register file organization, workload behavior, compiler optimizations, and steps that can be taken to improve upon the register file's lifetime for four of the main factors of influence in reliability (i.e., EM, SM, TDDB and TC). Moreover, using this reliability analysis of the register file, we propose in this paper a reliability-aware register assignment policy that can be statically applied by the compiler to significantly improve the mean-time-to-failure (MTTF) of the register file for several different types of applications (approximately 20% more expected lifetime on average).

This paper is organized as follows. In Section II, we review related work on reliability and thermal management. In section III we present the reliability models used in this

This work is partially supported by the Spanish Government Research Grant TIN2008-508.

0-7695-3770-7/70 $25.00 © 3770 IEEE
DOI 10.1109/IWIA.2008.7

work and in Section IV we discuss the reliability emulation platform. Then, in Section V we discuss our experimental setup and provide our results. Finally, in Section VI we summarize the paper and provide our conclusions.

II. Related work

Power density improvements and thermal management techniques have become two key factors for extending the reliability of multi-core and MPSoCs [4]. Morever, it has been shown that large temperature variations causing increased leakage current will consequently cause reliability to suffer in future technologies [2], [9]. Therefore, the need for improved power and thermal management techniques still exist, however, now expanding those techniques to incorporate reliability metrics is crucial as we head into the future.

In the past there has been attempts to improve upon energy and performance at the microarchitectural level and above. In [10], the authors try to dynamically minimize energy and improve system performance by exploiting architectural and application-level adaptability. Also, fault-tolerant microarchitectures have been proposed to improve reliability and hard failures [11]. In [12], the authors propose the use of redundancy at the architecture level to improve system reliability and processor lifetime. Additionally, several works analyze how to handle soft errors [13]. Finally, the work presented in [14] introduces dynamic fault-tolerance management as a method to enhance system reliability, taking into account energy efficiency, computation performance, and battery lifetime. However, all these previous methods imply performance or area overheads to enhance MPSoC and microarchitectural reliability against soft- and hard-errors, and can only be included according to the particular reliability requirements of each final system. Thus, frameworks that use effective modeling to help analyze the links between power, thermal behavior, and reliability in MPSoCs and multi-core systems are still required.

Several groups have addressed the problem of thermal and reliability modelling at different levels of abstraction. [15] presents a thermal model for embedded architectures, while [2] presents a thermal/power model for superscalar architectures. It predicts temperature variations in processor components and shows effects in leakage power and performance, but no link to reliability is given. Then, [16] outlines a simulation model and its validation on embedded cores, but no reliability figures are given. On the reliability side, RAMP [17] models chip wide MTTF as a function of the failure rates of individual structures on chip due to different failure mechanisms. Thus, RAMP can be combined with architecture-level MPSoC and microprocessor simulators that give power and temperature estimates needed by its reliability models [5], [6]. However, these complex cycle-accurate simulators are very limited in performance (circa 100-200 Khz) due to signal management overhead. Thus, they are not suitable to analyze, in detail, the reliability issues in MPSoC architectures running real-life applications. In contrast to previous work, we use in this paper a complete micro-architectural emulation framework that combines thermal and reliability models to target a detailed reliability analysis of the register file while executing different types of applications.

III. Reliability Models

The analysis of the temperature on the reliability of CMOS systems is investigated through the use of several mathematical models that include this dependency. The effects that will be included in our experimental work have been selected by their strong impact on the MTTF [4]: electromigration (EM), time-dependent-dielectric-breakdown (TDDB), stress migration (SM), and temperature cycling (TC).

em appears due to the momenta exchange between the electrons and the aluminum ions found in long metal lines. The induced mechanical stress may eventually cause fractures and shorts. The model generally accepted to describe the MTTF due to this effect takes the form:

$$MTTF = A_0 \cdot (J - J_{crit})^{-N} \cdot exp(E_a/kT)$$

where A_0 is the scale factor, J_{crit} is the critical current density and N is assumed to be 2 in metal-layered systems [4]. Our reliability model considers that the em is a non-reversible process and the actual value depends on the instantaneous temperature.

TDDB is an important failure mechanism that models how the dielectric fails when a conductive path forms in the dielectric, shorting the anode and cathode. This effect is modeled as:

$$MTTF = A_0 \cdot exp(-\gamma E_{ox}) \cdot exp(E_a/kT)$$

where γ is a field acceleration parameter, which is temperature dependent. In this case, our reliability model considers that this effect is a recovery process non-dependent on the instantaneous temperature, but a wider simulation window in the order of few seconds.

SM describes the movement of metal atoms under the influence of mechanical-stress gradients. The resistance rise associated with the void formation may cause electrical failures. The thermomechanical stress model can be written:

$$MTTF = A_0 \cdot (T_0 - T)^{-n} \cdot exp(E_a/kT)$$

where n ranges between 2-3, depending the manufacturing sizes [4].

Finally, TC produces a permanent damage that accumulates each time the device undergoes a normal power-up and power-down cycle. This effect is modeled as:

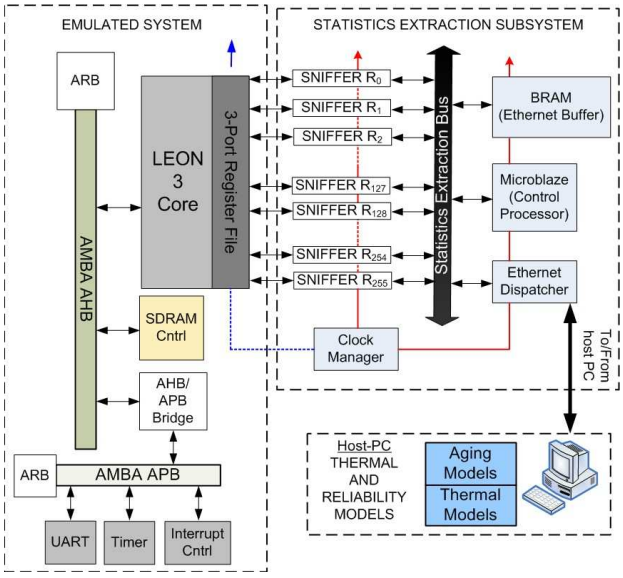

Figure 1. Overview of reliability emulation framework of the Leon 3 register file

$$MTTF = log\left(\frac{1}{T - T_{ambient}}\right)^{q}$$

where q is 2.35 for the considered technology [4].

IV. Reliability Emulation Framework

To enable a meaningful evaluation of register files found in modern day processors, the hardware emulation platform must have the ability to investigate a variety of real-world processors. As a result, we have implemented an emulation system built around the IEEE-1754 Leon 3 Sparc v8 Processor core [18]. The Leon 3 core is a fully customizable microprocessor containing multiple features common to those found commercially. The main features include separate instruction and data caches, a hardware multiplier and divider, a memory management unit (MMU), separate (or combined) instruction and data translation lookaside buffers (TLBs), and has the potential to be extended to a multi-core configuration. The Leon 3 is designed primarily for embedded systems applications and allows for a large range of customizations. In particular, components such as the register file, caches, and TLBs are customizable in terms of size and replacement policy in our Leon 3-based framework. Thus, the designer has the flexibility to configure the system architecture that needs to be tested from the viewpoint of reliability.

Our Leon 3 emulation framework (see Figure 1) is made up of three primary components, namely, the emulated system, the statistics gathering engine, and the host PC. In this case, the emulated system contains the Leon 3 system that is under investigation. The statistics gathering engine includes components that can monitor evens that occur in the Leon 3 register file and the host PC is running applications that directly interact with the statistics gathering engine to calculate the thermal behavior and reliability effects.

The emulated Leon 3 core architecture is shown on the left side of Figure 1. This architecture contains a 3-port register file of 256 registers (with 8 register windows), has a SDRAM memory controller, 16Kb 4-way set associative instruction and data caches, and separate instruction and data TLB's, each containing 32 entries. Furthermore, the Leon 3 system includes 64KB of on-chip ROM and RAM (not shown), 512MB DDR Memory, AMBA buses (both the AHB and APB), a serial I/O controller, timers, and interrupt controllers. Finally, the communication interface to load applications is provided through a serial UART (RS232) port.

A. Register file modeling

The register file found within the Leon 3 system is composed of two read ports and one write port with each port having separate address and data buses. The register file is actually composed of 8 *global* registers and a configurable number of register windows. The structure of the register windows is specified by the Sparc v8 standards and contains 8 *local* registers, 8 *in* registers, and 8 *out* registers.

To provide communication between the register windows the *in* and *out* registers are shared between the previous and next register windows respectively, with the *local* registers being exclusive to the currently selected register window. The specific layout of the register file considered in this work is depicted in Figure 2.

As this figure shows, the layout of the register file is divided in 32 rows and 8 columns, configuring a device with 256 registers. As was previously said, this number can be configured on user demand.

B. Statistics gathering engine

The statistics gathering engine, shown on the right side of Figure 1, was modeled based on the framework described in [19]. In this work the statistics engine was extended with the necessary components used to control and monitor the emulated Leon 3 system. The first main component was the hardware sniffers used to snoop signals within the Leon 3, with each sniffer capable of monitoring a single or multiple system components. We have included separate monitors for each register of the register file, as shown in Figure 1, which sample the monitored signals every 10 ms and calculate the consumed energy in each register. Then, in addition to the shared buffer where the sniffers write the statistics of each interval, a Microblaze was used to provide synchronization for statistics extraction between the sniffers and the host PC. Finally, the communication between the statistics extraction engine and the host PC executing the thermal and reliability models was done through an standard Ethernet connection available on the FPGA where the emulation was performed.

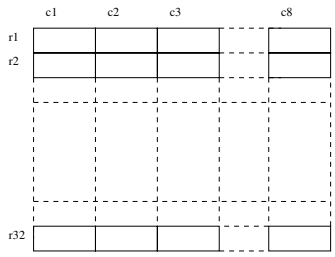

Figure 2. Layout considered for the register file of the Leon 3

The host PC contains software to provide thermal estimation and reliability characterization of the register file based on switching activity of its individual registers. More specifically, the host PC uses the gathered statistical data with the energy consumed in each register, coming from the statistics extraction engine, and incorporates it into the thermal models to determine the run-time thermal behavior of each register. Then, the temperature, power and energy results are included in the reliability models to calculate MTTF for each register of the emulated Leon 3 microarchitecture. From these results the MTTF estimate can be given for the entire register file.

V. Experimental Profiling and Reliability Enhancement of the Register File

The emulation platform described in the previous section has been used to perform complete reliability analysis for the register file of Leon 3 processing core. In this analysis, the effect of the application domain, as well as the code transformations regulated by the compiler, have been analyzed. Moreover, using the outcome from this analysis, we have defined a reliability-aware register assignment policy to enhance the MTTF of the register file.

A. Experimental setup

To analyze the effects that the application domain has on the reliability of the register file a set of common embedded applications from MiBench [20] and CommBech [21] suites have been selected. Among these applications, data-processing (FFT, reed), mathematical and graph theory (basicmath, dijkstra) and ordering/searching (bitcount, qsort, stringsearch, etc.) algorithms can be found.

These applications have been compiled with a cross-generated version of gcc 3.2.3 for the Sparc architecture. Also, four different versions of the compiled benchmark have been generated attending to the four optimization levels found in gcc (-O0, -O1, -O2 and -O3). The following section details the collected results for the described setup.

B. Reliability profiling

The first set of experiments studies the effect of the target application on the MTTF of the register file. Figure 3 shows the evolution of the MTTF (in percentage with respect to the nominal 20 years), where the X-axis represents the nominal

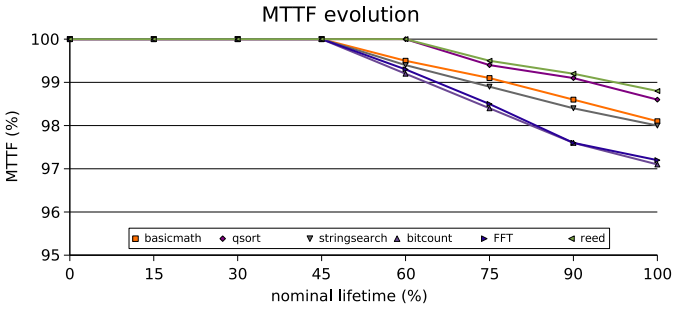

Figure 3. MTTF evolution for various benchmarks.

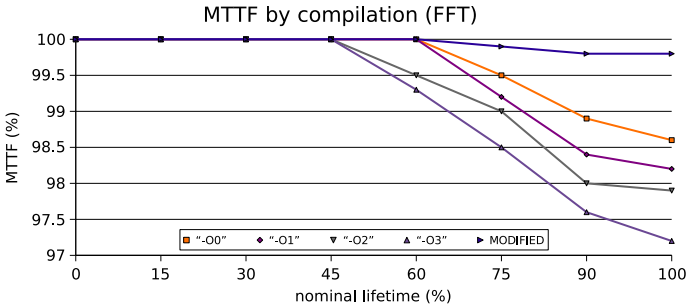

Figure 4. MTTF evolution for the FFT benchmark under different compiler optimizations.

MTFF in months. As can be seen, independently from the application domain, the key differentiator to identify the worst benchmarks from the reliability viewpoint is the analysis of which ones make intensive use of a reduced number of registers, namely, FFT and bitcount in our results. Thus, they are the benchmarks that experience a most severe MTTF reduction (nearly 35% in 10 years) due to the hotspots found in the highly-accessed registers. On the other hand, those data-processing benchmarks with an extended number of assigned registers (i.e., qsort and reed) experience a lower impact on the MTTF evolution (14% in 10 years).

The second set of experiments evaluates the effect of the different compiler optimizations (-O0, -O1, -O2) and the modified register assignment policy on the MTTF for the FFT benchmark, which is one of the applications that uses a larger number of registers. As can be seen in Figure 4, the less optimized policy (-O0 option) regarding the number of used registers is the one that provides a lower impact on the MTTF reduction (2% on average), while the register reuse conducted by the more extensive compiler optimization options impact negatively the MTTF (6% and 9% for the -O2 and -O3 options, respectively) in the sampled interval.

In the next set of experiments, the effect of every reliability factor on the final MTTF is evaluated. Figure 5 shows the evolution of the four main reliability factors for the FFT benchmark, compiled under the -O3 optimization. As seen, SM is the dominant factor in the reduction of the MTTF. The faster slope of SM is due to the fast thermal dynamism of

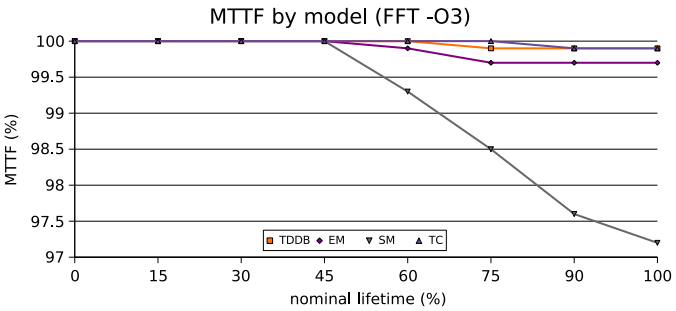

Figure 5. MTTF model evolution for the FFT benchmark compiled with -O3.

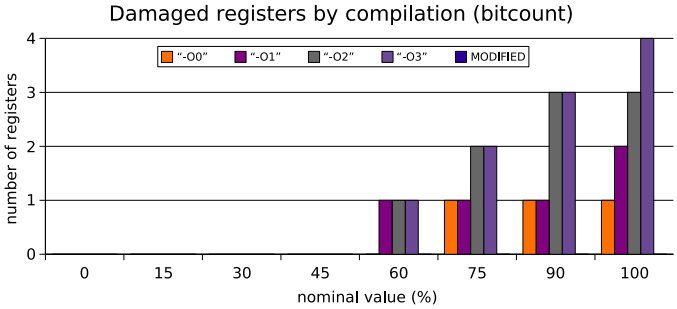

Figure 6. Number of damaged registers under different compiler optimizations and our reliability-aware algorithm (`MODIFIED`).

the system in different execution phases (i.e., approximately 12°C of differences can occur in few seconds), as predicted by different thermal models in the literature [2], [5], [19].

Finally, the number of damaged registers has also been estimated in order to quantify the degree of device failure. A register is considered to be damaged if its MTTF is below 2% of the nominal value. This information is very useful for the microarchitecture designer to understand the consequences of the optimization policies applied by the compiler in the register file lifetime. The number of damaged registers at the end of a sample interval of 3 years is depicted in Figure 6. As this figure shows, the amount of damaged registers on average for the bitcount benchmark, one case study with high pressure in the register file, varies between 1 and 4 for the studied interval, and between 15 and 40 in 20 years, depending on the optimization level used by the compiler. Moreover, the maximum optimization level (-O3) is the one with worse reliability, showing in our results that the probability of having at least 4 registers damaged in the first 2 years in the worst case reaches a probability of 99.5%, making critical the development of reliability-aware register assignment policies by microprocessor designers to increase processors lifetime.

C. Reliability enhancement policy

Using the aforementioned information from our reliability emulation framework about the register file, we have defined

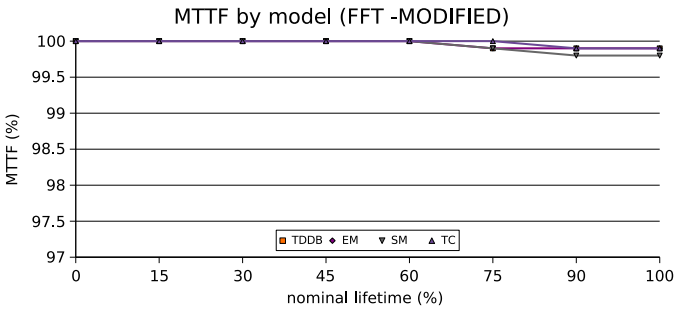

Figure 7. MTTF model evolution for the FFT benchmark compiled with our reliability-aware register assignment algorithm.

a new register assignment policy, implemented in the compiler, which tackles the strong impact of the compilation phase of the different benchmarks and related optimizations on the register file. We have modified the common assignment algorithm performed by gcc, which selects registers for instructions from a pool of unordered registers trying to minimize the number of them according to the level of optimization (-O0 to -O3). In this new proposed algorithm, the selection of registers in the modified gcc is performed by remembering the latest registers accessed in the respective window such that following register assignments follow a *chess board* fashion. Thus, a better diffusion of heat is performed within the different register windows and the risk of hotspots is minimized, trying to improve the reliability figures of the device.

The results of this new register assignment policy regarding the reduction of the number of damaged registers are shown in Figure 6. As can be seen, the spread of the register assignment per window performed by our `MODIFIED` policy included in the gcc compiler eliminates any damaged register in the sampled interval (3 years) for the FFT benchmark.

In a final set of experiments, the effectiveness of our register assigment policy has been evaluated on decreasing the MTTF degradation. Figure 7 shows the evolution in time of the MTFF for the FFT benchmark when this compilation policy is used. As can be seen, when compared with Figure 5, the performed assignment efficiently reduces the MTTF degradation (1% in 10 years).

VI. Conclusions

The alarming rise of power densities and temperatures in the die can seriously affect the reliability of MPSoC processing cores. Therefore, mechanisms to accurately evaluate the reliability features at the microarchitectural level are in great demand nowadays. In this paper we have presented a detailed reliability analysis of the register file architecture of the Leon 3 processor. This analysis was possible due to the use of an accurate HW/SW FPGA-based emulation platform that enables a complete design space exploration of thermal and reliability metrics altogether, in a short amount

of time, during the execution of a complete range of different benchmarks. Our results with the Leon 3 processing core have shown that the target application domain used has a large impact on the reliability of the register file, as well as the use of different compiler optimizations and register assignment policies. Moreover, using our reliability analysis we have proposed a reliability-aware register assignment algorithm, which significantly improves the MTTF of the register file.

REFERENCES

[1] A. Jerraya and W. Wolf, *Multiprocessor Systems-on-Chips*. Morgan Kaufmann, Elsevier, 2005.

[2] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation." *Transaction on Architectures and Code Optimizations (TACO)*, vol. 1, no. 1, pp. 94–125, 2004.

[3] S. S. I. Association, "The international technology roadmap for semiconductors," Tech. Rep., 2006, http://public.itrs.net/.

[4] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Lifetime reliability: Toward an architectural solution," *IEEE Micro*, vol. 25, no. 3, pp. 70–80, 2005.

[5] A. K. Coskun, T. S. Rosing, Y. Leblebici, and G. D. Micheli, "A simulation methodology for reliability analysis in multi-core socs," in *GLSVLSI '06: Proceedings of the 16th ACM Great Lakes symposium on VLSI*. New York, NY, USA: ACM Press, 2006, pp. 95–99.

[6] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *ISCA '00: Proceedings of the 27th annual international symposium on Computer architecture*. New York, NY, USA: ACM Press, 2000, pp. 83–94.

[7] V. Zyuban and P. Kogge, "The energy complexity of register files," in *ISLPED '98: Proceedings of the 1998 international symposium on Low power electronics and design*. New York, NY, USA: ACM Press, 1998, pp. 305–310.

[8] N. S. Kim and T. Mudge, "The microarchitecture of a low power register file," in *ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design*. New York, NY, USA: ACM Press, 2003, pp. 384–389.

[9] M. S. O. Semenov, A. Vassighi, "Impact of self-heating effect on long-term reliability and performance degradation in cmos circuits," *IEEE Transactions on Device and Materials Reliability*, vol. 1, pp. 17 – 27, March 2006.

[10] J. J. Sharkey, D. V. Ponomarev, K. Ghose, and O. Ergin, "Instruction packing: Toward fast and energy-efficient instruction scheduling," *ACM Trans. Archit. Code Optim.*, vol. 3, no. 2, pp. 156–181, 2006.

[11] K. R. Walcott, G. Humphreys, and S. Gurumurthi, "Dynamic prediction of architectural vulnerability from microarchitectural state," in *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*. New York, NY, USA: ACM Press, 2007, pp. 516–527.

[12] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Exploiting structural duplication for lifetime reliability enhancement," in *ISCA '05: Proceedings of the 32nd annual international symposium on Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 520–531.

[13] P. Shivakumar, S. W. Keckler, C. R. Moore, and D. Burger, "Exploiting microarchitectural redundancy for defect tolerance," in *ICCD '03: Proceedings of the 21st International Conference on Computer Design*. Washington, DC, USA: IEEE Computer Society, 2003, p. 481.

[14] P. Stanley-Marbell and D. Marculescu, "Dynamic fault-tolerance management in failure-prone battery-powered systems," in *Proceedings of 12th International Workshop on Logic and Synthesis (IWLS 2003)*, Laguna Beach, CA., May 2003, pp. 370–381.

[15] J. L. Ayala, C. Méndez, and M. López-Vallejo, "Analysis of the thermal impact of source-code transformations in embedded processors," in *Proceedings of ICECS*, 2006.

[16] H. Su, F. Liu, A. Devgan., E. Acar, and S. Nassif, "Full chip leakage estimation considering power supply and temperature variations," in *Proc. IEEE/ACM ISLPED*, Aug. 2003, pp. 78–83.

[17] J. Srinivasan, S. Adve, P. Bose, J. Rivers, , and C. Hu, "RAMP: A model for reliability aware microprocessor design," IBM Research Report, Tech. Rep., 2003.

[18] G. Research, "Leon 3 sparc v8 processor core," Tech. Rep., 2005, http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53.

[19] D. Atienza, P. G. D. Valle, G. Paci, F. Poletti, L. Benini, G. D. Micheli, and J. M. Mendias, "A fast hw/sw fpga-based thermal emulation framework for multi-processor system-on-chip," in *Proceedings of the 43rd annual conference on Design automation (DAC)*. ACM Press, 2006, pp. 618–623.

[20] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *WWC '01: Proceedings of the Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 3–14.

[21] R. Ramaswamy and T. Wolf, "Packetbench: A tool for workload characterization of network processing," in *Proc. of IEEE 6th Annual Workshop on Workload Characterization (WWC-6)*, Austin, TX, USA, Octber 2003, pp. 42–50.