Reliability-Aware Design for Nanometer-Scale Devices

David Atienza[1,2], Giovanni De Micheli[1], Luca Benini[3],
José L. Ayala[2], Pablo G. Del Valle[2], Michael DeBole[4], Vijay Narayanan[4]

[1] LSI/EPFL, EPFL-IC-ISIM-LSI Station 14, 1015 Lausanne, Switzerland.
[2] DACYA/UCM, Avda. Complutense s/n, 28040 Madrid, Spain.
[3]DEIS/UNIBO, Viale Risorgimento 2, 40134 Bologna, Italy.
[4]CSE/PSU, University Park, PA 16802, USA.
e-mail: {david.atienza,giovanni.demicheli}@epfl.ch, lbenini@deis.unibo.it,
{jayala,pgarcia}@fdi.ucm.es, {debole,vijay}@cse.psu.edu [*]

Abstract— Continuous transistor scaling due to improvements in CMOS devices and manufacturing technologies is increasing processor power densities and temperatures; thus, creating challenges to maintain manufacturing yield rates and reliable devices in their expected lifetimes for latest nanometer-scale dimensions. In fact, new system and processor microarchitectures require new reliability-aware design methods and exploration tools that can face these challenges without significantly increasing manufacturing cost, reducing system performance or imposing large area overheads due to redundancy. In this paper we overview the latest approaches in reliability modeling and variability-tolerant design for latest technology nodes, and advocate the need of reliability-aware design for forthcoming consumer electronics. Moreover, we illustrate with a case study of an embedded processor that effective reliability-aware design can be achieved in nanometer-scale devices through integral design approaches that covers modeling and exploration of reliability effects, and hardware-software architectural techniques to provide reliability-enhanced solutions at both microarchitectural- and system-level.

I. Introduction

The relentless scaling of technology and increase in transistor densities are primary reasons for *Multi-Processor System-on-Chips (MPSoCs)* to have become possible [10]. However, power requirements have not scaled accordingly, causing power densities to skyrocket and on-chip temperatures to increase at alarming rates [16]. Thus, the International Technology Roadmap for Semiconductors [1] has predicted that traditional design constraints centered around product cost and performance requirements will soon be overtaken by system wear-out failures and lifetime reliability issues [18].

The ability to model and evaluate thermal and reliability effects (e.g., electromigration, stress migration, time-dependent dielectric breakdown or thermal cycling) in a very early stage of the design flow of nanometer-scale devices has proven to be critical to enhance system lifetime [4, 18]. However, acquiring realistic reliability estimates in latest technology nodes is very complex because the sources of failures and thermal degradation phenomena in nanometer-scale MPSoCs are poorly understood, and span different degradation dynamics. Thus, well-tuned reliability models for the different windows of application of each degradation factor are needed to characterize systems switching activity and reliability behavior of final MPSoC architectures. In addition, very long accurate simulations (e.g., hundreds of millions of cycles [6]) are required to realistically model the variations and failures that manifest themselves in different phases of the lifetime of MPSoCs in the target working environments. As a result, novel (fast and accurate) thermal-reliability analysis methods need to be developed and incorporated in the early stages of the design of nanoscale devices to explore the existing trade-offs between area and performance in reliable and variability-tolerant design.

Indeed, reliability is an old concern that has been already pursued in high-end military and aerospace markets with very tight *Mean Time To Failure (MTTF)* requirements [4]. Three major trends can be found to affect system reliability according to different time-span intervals. The first trend is coming from silicon manufacturing process variations that result in a large percentage of produced devices to operate below the minimum acceptable speed; thus, creating a large number of time-zero yield loses, and requiring a number of prevention measures at the device level (e.g., proximity correction, phase-shifting masks, etc.) [5, 13]. The second trend of unreliable behavior is originated by *single event upsets (SEUs)* produced by transient phenomena throughout the utilization of the final system, such as, radiations, cross-talk, or ground bounce among others. In this case, circuit- and system-level techniques can reduce these transient errors (e.g., redundant execution with voting schemes, double sampling flip-flops, etc.) [3, 7]. The third trend refers to non-reversible failures in advanced periods of system lifetime due to thermal effects and aging of the devices [18, 19]. In the latter case, both HW (e.g., duplication or triple modular redundancy [3]) and software approaches (e.g., dynamic voltage and frequency scaling or static instruction-window selection [8, 18] have been proposed. Nevertheless, since the growing rise of random non-uniformity in microscale elementary devices is creating significant reliability effects in large-scale electronics systems, and thus instances of the same design behave differently in the same final conditions, the application of each of the aforementioned techniques in new MPSoCs need to be carefully evaluated.

[*]This work is partially supported by the Swiss FNS Research Grant 20021-109450/1, and the Spanish Government Research Grant TIN2005-5619.

In this work we present an analysis of the techniques and requirements needed to explore aging effects and provide reliability-aware design in nanoscale MPSoCs. We illustrate the application of novel HW-SW thermal and reliability analysis frameworks to enhance the resilience to aging of processing cores of MPSoCs at the microarchitectural level. In particular, we provide for illustration purposes the consequences of different compiler optimizations and registers assignment techniques in the shared register file architecture of a Leon 3 processor [15] while executing an extended set of real-life benchmarks. The switching activity and temperature of the register file makes it a critical thermal hotspot, and its failures are very difficult to detect during testing. Thus, it is a key component to improve the lifetime of microprocessors [11, 17]. Finally, we show how we can exploit this exploration to propose a reliability-aware register file assignment policy that consistently improves MTTF (20% on average) for the various benchmarks.

This paper is organized as follows. In Section II, we overview methods to study thermal and aging-related reliability effects in nanoscale devices, and proposed techniques to enhance system reliability. In Section III we present main reliability factors that need to be modeled in new MPSoCs. In Section IV we discuss the application of novel techniques to rapidly perform reliability explorations for MPSoCs. In Section V we show the possible benefits of reliability-aware design applied at the microarchitectural level of processing architectures. Finally, in Section VI we summarize our conclusions in the area of reliability-aware design.

II. Reliability-Aware Design Approaches

Although the literature on aging-related reliability study and design in CMOS is very broad to provide an exhaustive review in this paper [4, 5, 18], it is feasible to classify the related work on two main research lines. On the one hand, different works tackle the problem of studying aging effects in new nanoscale devices. On the other hand, approaches exist that deal with reliability enhancement against aging and thermal effects.

Reliability modeling: A very important effort has been done to create statistical models of aging effects in system components, and defining possible approximations of upper bounds for CMOS devices reliablity at different levels of abstraction. Chip wide MTTF can be modelled as a function of the failure rates of individual structures on chip due to different failure mechanisms [8, 18]. Also, [20] models failures probabilities in SRAM cells and proposes the use of statistical design for nanoscale CMOS devices. Since thermal effects affect reliability and process variations, [16] presents thermal/power models for super-scalar architectures. These models predict upper-bound temperature variations in processor components and need to be linked to aging models to suitably analyze reliability figures in architecture-level MPSoCs. Also, analytical approaches have shown how large temperature variations cause increased leakage currents, which affect reliability in future nanoscale devices [8, 13, 16]. In addition, using the previous models, different architecture-level MPSoC and microarchitecture simulators and emulators have appeared [2,6,17,22], which provide bounds of temperature and reliability variations in processor microarchitectures and MPSoC components, and enable accurate explorations of overall thermal and reliability

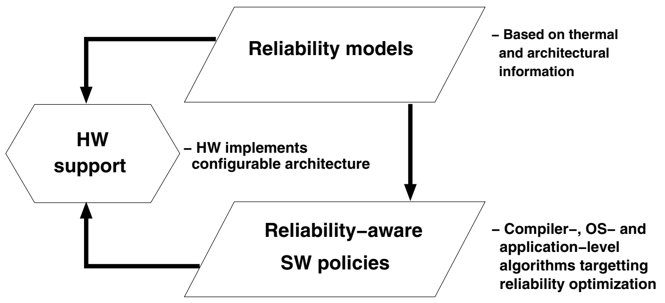

Reliability models
– Based on thermal and architectural information
HW support
– HW implements configurable architecture
Reliability–aware SW policies
– Compiler–, OS– and application–level algorithms targetting reliability optimization

Fig. 1. Reliability-aware design flow for nanoscale MPSoCs

figures. Both simulation and emulation approaches outline the importance of fast and accurate validation methods of reliability enhancement techniques, according to final working conditions of forthcoming nanoscale MPSoC designs.

Reliability enhancement: two major trend lines can be identified in this area. On the one hand, pure HW-based approaches proposing redundancy at the architectural level and the inclusion of spare components and self-diagnosis [3, 7]. On the other hand, as MPSoCs are highly programmable systems, a second trend proposing SW-based reliability enhancement or combined HW-SW approaches have appeared in the last years. In [19], dynamic prediction fault-tolerant microarchitectures have been proposed to improve reliability and hard failures. Also, Dynamic Reliability Management (DRM) [8, 18] tackle performance degradation in nanoscale systems by keeping wear-out profiles within a certain threshold relying on Dynamic Frequency and Voltage Scaling (DVFS) or clock throttling. Similarly, [21] maximizes electromigration lifetime by preserving wire temperature within certain upper-bounds. Then, [7] tries to improve system reliability at the register file level by exploiting statistical analysis to limit architectural overhead and application-level information. Finally, [12] explore reliability vs performance trade-offs by combining hardware and software management techniques in the register file. However, the previous methods imply performance or area overheads to enhance reliability against aging and other factors, and should be included according to the particular reliability requirements of each final system.

The previous approaches outline the foundations of effective reliability-aware design in forthcoming nanometer-scale MPSoCs, which requires a seamless integration in an overall flow of several modeling and architectural techniques at different levels, as shown in Figure 1, namely:

- Combination of reliability models and fast thermal-reliability exploration techniques.

- Inclusion of reliability monitoring mechanisms and tuning knobs at the hardware level.

- Development of software-based reliability management policies, which employ the available hardware support to enhance reliability both at microarchitectural- and system-level with limited performance and area overhead.

In the following sections we illustrate the application of the previous elements for reliability-aware design to analyze and enhance the aging-related reliability of the register file of a typical embedded processing core (the Leon 3 [15]).

III. Aging-Related Reliability Model

The influence of temperature in aging-related reliability of CMOS-based MPSoC components can be analyzed through several mathematical models that define this dependency. Effects that have been observed to have a very strong impact on the MTTF of systems and processor microarchitectures [18] are: *electromigration (EM)*, *time-dependent-dielectric-breakdown (TDDB)*, *stress migration (SM)*, and *thermal cycling (TC)*.

EM appears due to the momenta exchange between the electrons and the aluminum ions 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 a technology constant assumed to be 2 in metal-layered systems [18]. The presented reliability model considers that the effects of EM are non-reversible 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. It 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, the proposed reliability model considers that this effect is a recovery process, non-dependent on the instantaneous temperature, but with a simulation window 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 and 3 [18].

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

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

where q is 2.35 for the considered technology [18] and $T_{ambient}$ is the ambient temperature.

The input temperature considered by the previous equations is provided by a thermal model for MPSoC components [2]. This model, based on HotSpot [16], divides the die and heat spreader in cubic shape cells of several sizes. Each cell has five thermal resistances and one thermal capacitance, four resistances model horizontal thermal spreading and the fifth one covers the vertical thermal behavior. The generated heat is modeled by adding an equivalent current source to the cells on the bottom surface. The heat injected by the current source corresponds to the power density of the architectural component covering the cell multiplied by the surface area of the cell. Finally, each cell interacts only with its neighbors, which results in a linear complexity with respect to the number of cells. This approach can analyze 2 seconds of simulation (in a 660-cell floorplan), in 1.65 seconds on a Pentium 4 at 3GHz, which is fast enough to interact in real-time with current thermal-reliability simulation/emulation frameworks.

IV. HW-SW Reliability Exploration Frameworks

To enable a meaningful exploration of reliability-aware management policies at system and microarchitectural level of MPSoCs, new and fast exploration approaches need to be proposed, as aging effects demand large thermal and reliability simulations (i.e., millions of cycles) of real-life applications in final working conditions. Additionally, these exploration approaches must be able to investigate a large part of the design and manufacturing spectrum of MPSoC implementations (e.g., various floorplan layouts or packaging technologies, multiple frequencies and supply voltages, etc). Hence, we believe that a promising solution to effectively provide reliability studies are combined HW-SW exploration frameworks [2], which merge flexible SW simulation to easily validate the effects on reliability of a wide range of MPSoCs design alternatives (e.g., cheap or high-end packaging solutions), while acquiring cycle-accurate thermal behavior and switching activity of internal components at fast speed (i.e., 100-150 MHz) with respect to pure MPSoC architectural simulators [22].

For the sake of illustration, we have implemented a HW-SW thermal-reliability emulation system around the IEEE-1754 Leon 3 Sparc v8 Processor core [15]. 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 enables a large range of customizations (e.g., size or replacement policy of the register file, caches, and TLBs). Thus, components in our Leon 3-based framework. Thus, the designer can configure the system architecture that would like to test from the reliability viewpoint.

The constructed Leon 3 reliability emulation framework (Figure 2) is made of three primary components: the emulated system, the statistics gathering engine, and the host PC. The emulated system contains the Leon 3 system that is under investigation. The statistics gathering engine monitors events that occur in the Leon 3 register file. The host PC is running a SW library that interacts in real-time with the emulation engine to calculate the register file thermal behavior and reliability while different benchmarks run in the Leon 3.

The emulated Leon 3 core architecture (left side of Figure 2) 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, a serial I/O controller, timers, and interrupt controllers. Finally, the communication interface to load applications is provided through a

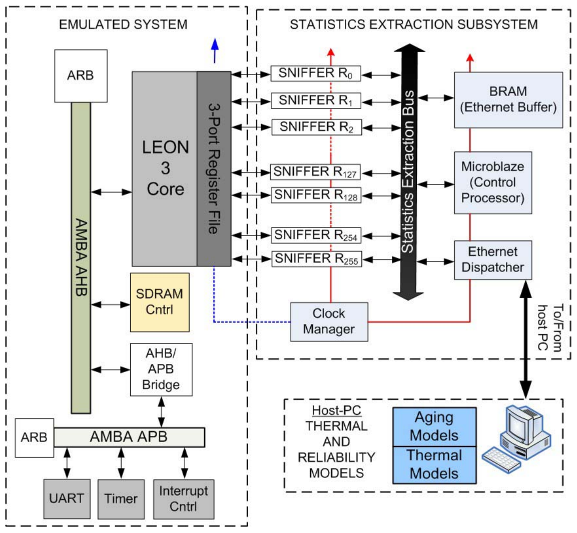


Fig. 2. Overview of reliability emulation framework of the Leon 3 register file

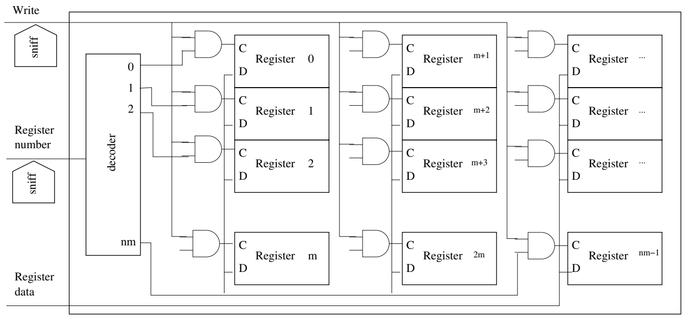


Fig. 3. Layout considered for the Leon 3 register file (m=32;n=8)

serial UART (RS232) port.

A. Register file modeling

The register file found in the Leon 3 is composed of two read ports and one write port, where each port has 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 case study is depicted in Figure 3. The layout of the register file is divided into 32 rows and 8 columns, configuring a device with 256 registers (this number can be configured on user demand).

B. Statistics gathering engine

The statistics gathering engine (on the right side of Figure 2) was modeled based on the framework described in [2]. In this work, it was extended with the necessary components used to control and monitor the emulated Leon 3 system. The first main component was the HW 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 2, which sample the monitored signals (e.g., register identifier, R/W lines, etc.) every clock cycle and calculate the consumed energy in each register every 10 ms (because the temperature evolution is a rather slow process [16]). 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 a standard Ethernet connection available on the FPGA where the emulation was performed.

The proposed emulation platform not only implements a higher complexity system than the one presented in [2]; thus, showing the scalability of HW-SW reliability exploration frameworks, but also enables thermal analysis at a greater level of granularity. Hence, the temperature for every register is acquired and an accurate and exhaustive register file reliability exploration can be performed.

The host PC contains software to provide thermal estimation and reliability characterization of the register file based on switching activity of its individual registers. 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. 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. Case Study: Register File Reliability Enhancement Design

The register file reliability emulation platform described in the previous section has been used to perform a complete reliability analysis for the register file of the Leon 3 core, implemented in 90 nm process technology. In this analysis, we have explored the effects of the application domain, as well as the code transformations regulated by the compiler. Finally, as an example of the potential benefits of reliability-aware design for nanoscale MPSoCs, 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

A set of embedded applications from MiBench [9] and CommBech [14] suites has been selected to analyze the effects that the application domain has on the reliability. 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 versions of each benchmark have been generated using the four optimization levels of gcc (-O0 to -O3). The results are normalized with respect to a nominal reliability value of 3 years. Thus, the X-axis in all the figures represents the normalized MTTF percentage within this nominal value.

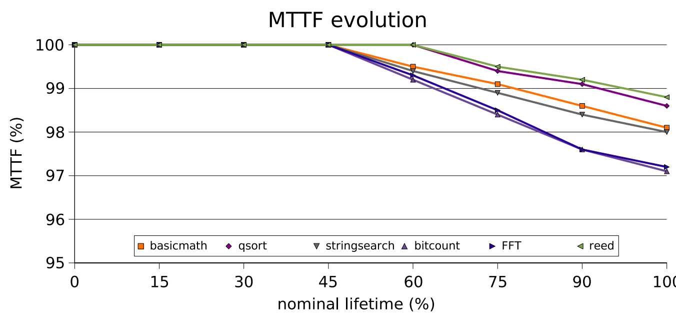


Fig. 4. MTTF evolution for various benchmarks.

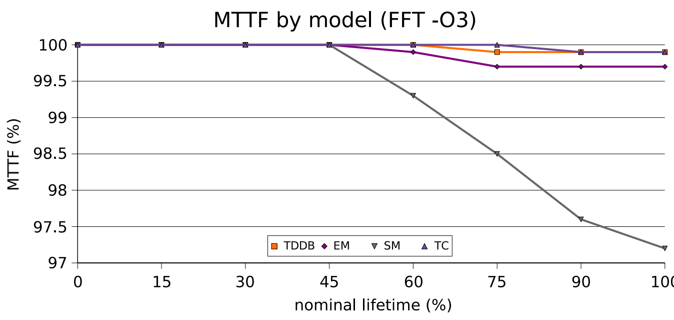


Fig. 6. MTTF model evolution for the FFT benchmark compiled with -O3.

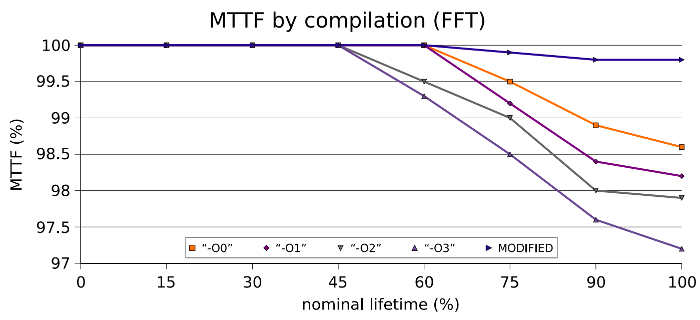


Fig. 5. MTTF evolution for the FFT benchmark under different compiler optimizations.

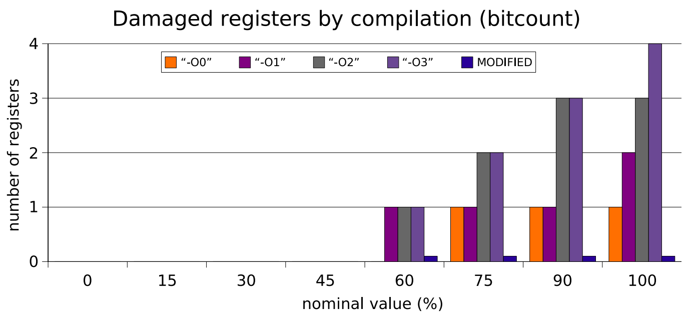


Fig. 7. Number of damaged registers under different compiler optimizations and our reliability-aware algorithm (`MODIFIED`).

B. Reliability emulation

The first set of experiments studies the effect of the target application on the MTTF of the register file. Figure 4 shows the evolution of the MTTF with respect to the normalized nominal value. As can be seen, independently from the application, the key differentiator used 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. Thus, they are the benchmarks that experience the most severe MTTF reduction (i.e., 35% in 10 years following the normalized pattern of Figure 4) 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).

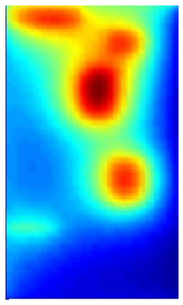
The second set of experiments evaluates the effect of the different compiler optimizations (-O0 to -O3) and the modified register assignment policy on the MTTF for the FFT benchmark. As Figure 5 shows, the less optimized policy (-O0 option) is the one that provides a lower impact on the MTTF reduction (1.5% on average), while the register reuse conducted by the most extensive compiler optimization options impact the MTTF negatively (2.5% and 3% for the -O2 and -O3 options, respectively) in the sampled interval, and up to 24% and 35% respectively in 10 years. Then, Figure 6 shows the evolution of the four main reliability factors for the FFT benchmark under the -O3 optimization. SM is the dominant factor in the reduction of the MTTF due to the fast thermal dynamism of the system in different execution phases (i.e., 12°C difference can occur in few seconds), as predicted by the different thermal models for sub-micron technologies [2, 6, 16].

Finally, the number of damaged registers has also been estimated 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 7. As it 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 12 and 45 in 10 years, depending on the optimization level used by the compiler. 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 99.5%, making critical the development of reliability-aware register assignment policies.

C. Reliability enhancement policy

Using the information about the register file from our reliability emulation framework , we have defined a new register assignment policy. It has been implemented in the `gcc` compiler, which uses two phases to allocate registers: one allocating local pseudos to registers, and one allocating the remaining pseudos used over basic block borders. Both phases only allocate a hard register to pseudos, but do not emit spill code.

The algorithms included in the current versions of gcc assign registers from a pool of free registers. Our proposed register allocation technique modifies the graph coloring algorithm found in [23] by selecting the target register after checking that the neighbors have not been previously assigned, if possible. In this way, the pattern of assigned registers from the register file resembles a *chess board*. Thus, a better diffusion of heat is performed within the different register windows and a broader selection of registers is expected to improve the register file re-


(a) Traditional register allocation.

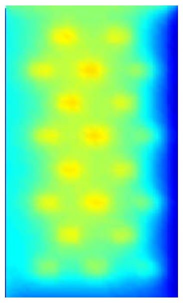

(b) Modified register allocation.

Fig. 8. Thermal distribution for the register file.

liability. Figure 8 shows the better spread of the heat and the reduction of hotspots in the register file when the modified allocation is employed.

As depicted in Figure 7, this new register assignment policy (`MODIFIED`) reduces the number of damaged registers. In fact, the spread of the register assignment per window performed by our policy eliminates any damaged register in the sampled interval (3 years) for the bitcount benchmark. Moreover, Figure 5 indicates that our policy is very effective to minimize MTTF degradation, where MTTF is only reduced by 0.1% in the sampled interval, thus, around 1% in 10 years, much smaller figures than any other policy. In fact, in comparison with -O3 (Figure 6), our results indicate that our policy reduces significantly (by 20% on average) the impact of all factors related to MTTF degradation, especially SM that significantly affects all the other policies.

Finally, we have evaluated the speed of the proposed HW-SW emulation framework in comparison with SW simulators. While the HW-SW emulation framework took 5 minutes approximately for the whole reliability exploration of each tested application, a complete MPSoC architectural simulator [22] required 2 days for 0.18 sec of real execution. Thus, the proposed emulation framework achieves more than three orders of magnitude of speed-ups (1612 $\times$) compared to SW-based thermal simulation, making feasible to perform long thermal and reliability studies in a limited time.

VI. Conclusions

Transistor scaling toward nanometer-scale devices is rapidly exacerbating reliability problems in large-scale MPSoCs. Thus, new reliability-aware design methods are required. As a result, a new generation of software-based reliability-enhancement techniques with limited area and performance overhead, thanks to the underlying hardware support, need to be proposed. Moreover, novel and fast exploration approaches must be deployed to validate the provided resilience to aging by the developed reliability-enhancement techniques for MPSoCs at different levels of abstraction (e.g., system-level, processor microarchitecture, etc.).

In this paper we have illustrated the feasibility and benefits of reliability-aware design by performing a complete reliability analysis of the register file architecture of a Leon 3 processor. Since this type of analysis is very time-consuming for pure SW simulators, we have presented the application of a HW-SW emulation framework, which enables an exhaustive exploration of the various reliability factors for a complete range of different benchmarks. Our experiments have shown that reliability-aware design is key to provide long-lasting nanoscale devices with very low overheads in terms of area and performance. In fact, the obtained results outline that the target application domain can have a very negative impact on the reliability of the register file, as well as the use of aggressive compiler optimizations and register assignment policies. However, effective reliability-aware register assignment algorithms can significantly enhance the MTTF of the register file (20% on average) for different kinds of applications.

References

[1] S. S. I. Association. The international technology roadmap for semiconductors. http://public.itrs.net/.

[2] D. Atienza, et al. HW-SW Emulation Framework for Temperature-Aware Design in MPSoCs *ACM TODAES*, 2007.

[3] D. Sylvester, et al. ElastIC: An Adaptive Self-Healing Architecture for Unpredictable Silicon *IEEE D&T*, 2006.

[4] S. Borkar, Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation, *IEEE Micro*, 2005.

[5] D. Sylvester, et al. Computer-aided design for low-power robust computing in nanoscale CMOS. *Proc. of IEEE*, 2007.

[6] A. K. Coskun, et al. Analysis and Optimization of MPSoC Reliability. *JOLPE*, 2006.

[7] J. Blome, et al. Cost-Efficient Soft Error Protection for Embedded Microprocessors. *Proc. CASES*, 2006.

[8] T. S. Rosing, et al. Power and Reliability Management of SoCs. *Trans. on VLSI*, 2007.

[9] M. R. Guthaus, et al. Mibench: A free, commercially representative embedded benchmark suite. *Proc. WWC*, 2001.

[10] A. Jerraya, et al. *Multiprocessor SoCs*. Elsevier, 2005.

[11] N. S. Kim, et al. The microarchitecture of a low power register file. *Proc. ISLPED*, 2003.

[12] G. Memik, et al. Engineering over-clocking: Reliability-performance trade-offs for high-performance register files. *Proc. DSN*, 2005.

[13] M. S. O. Semenov, et al. Impact of self-heating effect on long-term reliability and performance degradation in CMOS circuits. *Trans. on DMRR*, 2006.

[14] R. Ramaswamy, et al Packetbench: Tool for workload characterization of network processing. *Proc. WWC*, 2003.

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

[16] K. Skadron, et al. Temperature-aware microarchitecture: Modeling and implementation. *ACM TACO*, 2004.

[17] K. Patel, C Wonbok, M. Pedram. Active bank switching for temperature control of the register file in a microprocessor *Proc. GLSVLSI*, 2007.

[18] J. Srinivasan, et al. Lifetime reliability: Toward an architectural solution. *IEEE Micro*, 2005.

[19] K. R. Walcott, et al. Dynamic prediction of architectural vulnerability from microarchitectural state. *Proc. ISCA*, 2007.

[20] S. Mukhopadhyay, et al. Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS. *Trans. on CAD*, 2005.

[21] Z. Lu, et al. Interconnect Lifetime Prediction under Dynamic Stress for Reliability-Aware Design. *Proc. ICCAD*, 2004.

[22] L. Benini, et al. Mparm: Exploring the MPSoC design space with SystemC. *Journal of VLSI*, pp. 169-182, 2005.

[23] J-Y. Choi, et al. Low power register allocation algorithm using graph coloring. *Proc. TENCON*, 2000.