Cross-Layer Early Reliability Evaluation: Challenges and Promises

Stefano Di Carlo, Alessandro Vallero, Dimitris Gizopoulos, Giorgio Di Natale, Antonio Gonzales, Ramon Canal, Riccardo Mariani, Mauro Pipponzi, Arnaud Grasset, Philippe Bonnot, et al.

Cross-Layer Early Reliability Evaluation: Challenges and Promises

S.Di Carlo, A.Vallero
Politecnico di Torino
Italy

D.Gizopoulos
University of Athens
Greece

G.Di Natale
LIRMM
France

A.Gonzalez, R.Canal
UPC
Spain

R.Mariani, M.Pipponzi
YOGITECH
Italy

Arnaud Grasset, Philippe Bonnot
Thales
France

Frank Reichenbach, G.Rafiq, T.Loekstad
ABB
Norway

***Abstract*—Evaluation of computing systems reliability must be accurate enough to provide hints for the required fault protection mechanisms that will guarantee correctness of operation at acceptance costs. To be useful, reliability evaluation must be performed early enough in the design cycle when, however, the available details of the system are largely unknown. This inherent contradiction in terms: early vs. accurate, requires a cross-layer approach for reliability evaluation. Different layers of abstraction contribute differently in the overall system reliability; if this contribution can be assessed independently, the reliability of the system can be evaluated at the early stages of the design. We review the state-of-the-art in the area and discuss corresponding challenges**

Keywords—reliability evalution, computing systems

I. INTRODUCTION

Information technologies are literally changing the way people live, work and learn. For the past 20 years, researchers have developed technologies that allow sensing, computing, and wireless communication to be embedded in complex systems as well as every day objects. However, while in the past the development of different computing segments such Embedded Systems (ES) or High Performance Computing (HPC) systems has been driven by separated communities and players exploiting different technologies, a radical paradigm shift is expected in the upcoming years leading to a true *computing continuum* [1] ranging from smartphones to mission-critical datacenter machines, and from desktops to automobiles (Figure 1). The key characteristic of this computing continuum is that the same hardware and software technologies and industrial players will act across all computing segments introducing a radical change compared to past business and technical development models. On aggregate, this computing continuum represents a total addressable market approaching a billion processors a year, which is expected to explode to more than two billion per year before 2020 [1].

In this exciting scenario, very critical technological challenges are creating serious threats for the successful development of next generation computing systems in all segments. Future device integration technology is expected to dramatically reduce the device quality, and therefore the operational reliability of circuits due to higher device variability, manufacturing defects, aging, and higher susceptibility to transient faults (soft errors) and permanent faults (device degradation, wear-out) [2]. One of the biggest challenges for future technologies is the implementation of "dependable" systems on top of significantly unreliable components, which will degrade and even fail during normal lifetime of the chip.

Figure 1: The computing continuum

Digital systems reliability is not a new issue for the scientific community. However, current practices to guarantee reliability of electronic systems are based on pessimistic (worst case) scenarios and expend significant development costs, energy and resources to tolerate the device unreliability by adding fault tolerance mechanisms at different levels (technology, architecture, software). The rising energy costs required to compensate for increasing unreliability are rapidly becoming unacceptable in today's environment where power consumption is often the limiting factor on integrated circuit performance, and energy efficiency is a top concern. In this era, where low reliability threatens to seriously slow down or even to end the benefits of feature size reduction, a *holistic* approach to guarantee high reliability is required. Such an approach aims at ranging across different computing disciplines, across computing system layers and across computing market segments to have a unique reliability assessment methodology. Such a holistic approach in which several techniques can be applied at different levels in a complex system (e.g.,

technology, architecture, software, etc.) cannot be successful if not complemented with methods and tools enabling the assessment of the actual reliability level of a system starting from the early stages of its design. Nowadays, system time-to-market (TTM) is already pivotal for the market success of both HPC and ES designs. An increasing number of projects whose reliability was assessed at very late stages of the design cycle, miss their announced market entry dates due to major design changes that in many cases are not affordable. Therefore, early budgeting for reliability has the potential to save significant design effort and resources and has a profound impact on the TTM of a product. These considerations are supported by initiatives such as the CLERECO European Project [3] that investigate new methodologies to accurately assess system reliability through all stages of the design cycle for the future systems of the computing continuum.

This paper reviews current approaches for reliability evaluation at different design levels and discusses motivations and requirements for the research of new methods for early and precise reliability evaluation at system level.

II. RELATED WORK

The global research community has been very active in researching technological and architectural methods to improve reliability of specific components and systems. However, when focusing on the reliability evaluation, especially when moving at high abstraction levels, significant work still needs to be carried out. This section is an overview of how reliability is nowadays evaluated at different design stages.

A. Reliability prediction at circuit and gate-level

Circuit-level reliability estimation tries to estimate the probability of a given failure mode at the output of a logic gate due to intrinsic phenomena like aging, or external perturbations like heavy ions [4],[5]. Today, reliability device simulators have become an integral part of the design process. These simulators successfully model the most significant physical failure mechanisms in modern electronic devices. Moreover, some analysis pointed out that, the performance improvement of a full technological generation step can be lost due to process variations [6]. In response to the need to analyze designs under process variations, researchers have developed statistical timing analysis techniques to be applied for deep sub-micron chip designs [7].

Gate-level reliability estimation moves the focus to the nodes of a netlist [5]. Estimating the error susceptibility of a node requires computing the probability of sensitizing the node with an input vector able to propagate the erroneous value to one of the outputs of the circuit [6]. This however is a very computational intensive task and requires the simulation of several random vectors whose number significantly increases with the size of the circuit [4], [5], [8], [9], [10], [11]. Reliability prediction tools now model the failure probability of chips at the end of life by analyzing only the single dominant wear-out mechanisms. Modern prediction tools do not predict the random, post burn-in, failure rate that would be seen in the field.

B. Reliability prediction at architectural level

The research community has provided a lot of results in the area of Architectural Vulnerability Factor (AVF) calculation and AVF estimation. The AVF is the probability that a fault in a processor structure will result in a visible error in the final output of a program. Most attempts are offline analysis and AVF calculation with complex simulators [12], [13], [14], like ACE-analysis (Architectural Correct Execution). This offline estimation is a complex process, requiring many resources to track values and instructions as they travel through a processor. Normally, only a limited number of instructions can be analyzed in a reasonable amount of time. There have been some works on estimating the AVF in real time [15], [16]. Walcott et al. [16], apply statistical analysis using a detailed simulator to analyze the AVF behavior at large scale. Then they use linear regression to explore the relationship between and various microarchitecture level variables such as structure occupancy, number of instructions executed, etc.

AVF prediction has been also based on the study of various microarchitecture level variables. Duan et al. [17] proposes the use of boosted regression trees as a predictive model for AVF. Later, Biswas et al. [18] extend this work by calculating and estimating AVF over short windows of time, providing better opportunities for hardware components error protection during design. Soundararajan et al. [15] propose a method to estimate AVF for the reorder buffer (ROB) in the processor. Fu et al. [19] explore program reliability/vulnerability phase behavior to predict AVF. They observe that a methodology based on the study of the software code structure is promising in classifying program reliability phase behavior. They also explore the use of performance counters similar to previous works [16]. However, they only explore the AVF estimation for the issue queue and the reorder buffer in an out-of-order processor.

C. Reliability prediction of caches

Memory structures reliability is hard to predict and deserve specific solutions for vulnerability evaluation [20]. Following Duan et al. [17] work on using boosted regression trees as a predictive model, Ma et al. [21] developed a model based on Bayesian additive regression trees for the cache memories. Cheng et al. [22] study the variability in AVF for different cache configurations. Li et al. [23] propose to use simple modifications to the processor to estimate AVF.

D. Reliability prediction at system level

When analyzing the lifetime reliability of processor-based systems, it is essential to investigate the impact at system level. Srinivasan et al. [24] describe a model for lifetime analysis for microprocessors and conducted dynamic reconfigurations based on the model. In this model, authors assume identical vulnerability of devices and uniform device density over the whole chip. Later, Shin et al. [25] develop a fine-grain model where different structures of the processor (e.g., register file, functional units) have different failure mechanisms. However, the model suffers from the same inaccuracies as AVF analysis methods based on architectural simulators.

Other works predict lifetime reliability based on simulations [26]. Similar to previous work [24], [25], the failure mechanisms do not consider aging effects, which lead to

inaccuracies in the simulation results. Huang and Xu [27] have proposed AgeSim, a simulation framework for evaluating the lifetime reliability of SoCs at system level.

Recently, researchers have begun to explore the system-level impact of variations on power, performance, and reliability. Humenay et al. [28] and Romanescu et al. [29] developed models of process variation on pipelined processors. They show that globally-asynchronous, locally synchronous (GALS) design techniques may offer ways to mitigate the impact of correlated within-die variations, but random variations, particularly within memory structures, cannot be easily addressed with these coarse-grained approaches.

Reliability assessment is a critical topic, especially in large and heterogeneous systems. In fact, reliability evaluation, at system level, often relies on statistical models. Thus, model selection is crucial to obtain trustworthy results, depending on the application field. The research activity in this field is active even if not specifically related to the design flow of complex digital systems. Extreme value theory is often the theoretical foundation to build statistical models to evaluate both lifetime and reliability in several fields [30].

E. Reliability impact of software

Scarce works have focused on systematically including the software into the reliability evaluation process. The work published in [31] analyzes various compiler optimization effects on the AVF of an embedded processor. Similar approaches at the compiler level have been also proposed in [32] and [33]. In [34] the authors proposed a first attempt of performing static analysis of a computer system including its software. Sridharan and Kaeli [35] propose to compute a Program Vulnerability Factor (PVF) for a set of benchmarks exploited to improve AVF computation for several microprocessors. However, neither the final software workload, nor the full stack is explicitly considered. Savino, et al. [36] propose a solution that considers the impact of the application software running on embedded microprocessors. However, it does not consider the operating system (OS) and multi-processing solutions in the overall evaluation. One of the few attempts to consider the full software stack including the operating system has been proposed in [37]. Nevertheless, only very preliminary results are shown and it is uncertain how it can be extended to modern systems employing massive parallelism.

F. Impact of errors

The issue and impact of soft errors is an important emerging concern in the design and implementation of future microprocessors. A considerable amount of research at the microarchitecture level has conducted fault injection studies using software-based methods [38] [39] [40] [41] [42] [43]. They focus on understanding how errors in low-level circuits or hardware structures manifest at the architecture level. Inside this large body of work about fault injections and soft errors, there are studies that characterize the fault tolerant behavior of the system. The susceptibility of commodity operating systems running on IA-32 and IA-64 microprocessors to soft errors was investigated in [44]. The results indicated that with improved microprocessor support like the Machine Check Architecture (MCA), and a little application knowledge, the system could reduce the need for reboots due to the detected soft errors. Additionally, previous work [38] noticed that not all faults at the low levels affect program correctness.

Identifying the effects of faults in program behavior and which programs are more fault-resilient at higher levels has also been deeply studied. Some works [45], [46], [47] have analyzed certain aspects related to program output requirements, algorithm features, and cognitive resilience of various applications from the application's standpoint.

Finally, another research vector has explored the possible anomalous symptoms or atypical events caused by faults, and used this information to perform error detection at different levels [48], [49], [50]. This class of research tries to characterize and catch selected events, symptoms, to diagnose the likely presence of a failure caused by soft errors.

III. Early Reliability Estimnation: Challenges

Traditionally, reliability estimation performed at different stages of the design cycle can lead to worst-case decisions and over-designed systems. While the required system reliability can be guaranteed, the cost of the employed reliability mechanisms (in terms or area, energy/power, and performance) and the design time required for their integration and evaluation are both excessive. Moreover, standard reliability evaluation approaches strongly rely on massive and time-consuming simulations and/or fault injection campaigns, which are becoming a bottleneck due the increasing complexity of computing systems.

Early, fast, and accurate evaluation of computing systems reliability to support design decisions for hardware and software reliability enhancing mechanisms in the system is a key enabler for the continuation of technology scaling benefits harnessing for several decades. The benefits of an early and accurate reliability estimation methodology for the computing continuum are many. In particular, design time, energy efficiency and system performance are the main design dimensions that can benefit from this early analysis.

A. Design time

Early reliability evaluation supports the system design process and avoids under-estimations or over-estimations of the final system's reliability that can adversely affect the system development cost and/or its TTM constraints. Early and accurate identification of reliability weaknesses of the computing systems leads to suitable decisions for the employment of hardware and software mechanisms against them. When the reliability weaknesses are identified late in the system design, major redesign costs may be imposed and TTM can be significantly affected. Moreover systems are often over-designed to ensure the fulfillment of the reliability requirements. This is due to the lack of accurate and early estimation methods. Instead the system should be developed to fulfill its purpose at minimal design costs.

B. Energy-efficient reliability

Energy and power consumption is already the primary limitation in the design of computing systems in all domains. Typical reliability solutions based on worst-case scenarios and

massive guard-banding at low abstraction layers add excessive energy overheads and performance delimiters in scaled technologies. For instance, fault tolerance solutions such as Double Modular Redundancy (DMR) or Triple Modular Redundancy (TMR) are typically used in high-reliability systems. This means 100%-200% more area and energy overhead compared to the unprotected system. Early reliability evaluation system will enable the employment of more fine grain reliability solutions (e.g., use DMR only in the very vulnerable parts of the system) with a positive impact on the power budget of the system.

C. *Reducing the performance impact of reliability*

Throughput of systems will also benefit from the capability of early reliability evaluation. Reliability mechanisms based on time redundancy, including error correcting codes or message duplication are computation hungry and they may cause up to 40% of system computational power to be assigned to reliability related tasks. If the faults and their error propagation are known, it is possible to develop just the right mitigations to detect and handle errors in an efficient and accurate way. Again with the capability of performing early reliability evaluations and with the employment of more fine grain reliability solutions, designers will be able to obtain significant improvements in achieved performance.

IV. Early Reliability Estimation: Promises

The system reliability stack reported in Figure 2 summarizes the masking effect in a computer system. The system is split into three main layers: (1) technology, (2) hardware and (3) software. The low-level raw errors of the physical devices are masked in several different ways as their effect is propagated through the hardware layer and the software layer of the system stack towards the final program/application outputs.

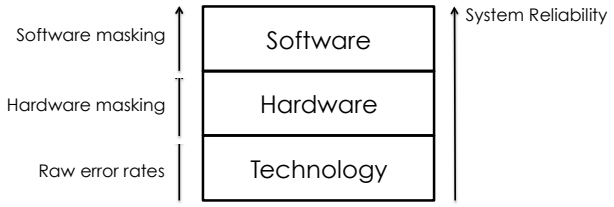

Figure 2: Masking effects in a computer system

A key enabler of early reliability of such systems is the possibility to *analyze these three layers in isolation*, to later combine the outcome of this local analysis in order to infer reliability measures at the system level. Each layer defines an interface with the upper layer that in turns sets how the errors can be propagated from one layer to the next one. The most difficult challenge is the ability of splitting the hardware and the software layers, and to study the two layers independently. While the hardware layer is already deeply analyzed (as shown in Section III), no mature works do exist for the software layer.

For a complete analysis, the software layer must consider both the system software (i.e., the operating system) and the application software. Since the global system outcome is commonly represent by the outcome of the software executed in the system (both application and system software), analyzing the software impact on the system reliability implies analyzing the way the software reacts on faults that reach its interface with the hardware layer. In general, the Instruction Set Architecture (ISA) of the target hardware platform executing the software defines this interface.

The ability of a software component to mask and/or intrinsically tolerate errors coming from the hardware is also referred in the literature as software resilience. To define the resilience of a software application, it is necessary to evaluate the probability of functional correctness of the software in the presence of errors in either the software data or software instructions.

It is important to highlight here that the main interaction point between the hardware layer and the software layer is the ISA of the microprocessors and co-processors (e.g., accelerators such as GPUs or crypto devices) available in the system. A straightforward way to model the error propagation from the hardware layer to the software layer is therefore to map hardware errors into a set of faults model that affect the ISA instructions and their data. This somehow detaches the software analysis form the underlying hardware analysis and moves the work of combining the obtained results later on.

Table 1 provides a potential taxonomy of software fault models defined at the ISA level that could be considered in a reliability estimation tool (either based on fault injection or static/dynamic analysis of the software). This is a first attempt to define a set of software fault models that can be directly linked to the effect of errors arising at the hardware level. All models are defined at the ISA level, they therefore apply both to system and application software.

Table 1: Definition of Software Fault Models

Software Fault Model	**Description**
Wrong Data in a Operand	An operand of the ISA instruction changes its value
Not-accessible Operand	An operand of the ISA instruction cannot change its value
Source Operand Forced Switch	An operand is used in place of another
Instruction Replacement	An instruction is used in place of another
Faulty Instruction	The instruction is executed incorrectly
Control Flow Error	The control flow is not respected (control-flow faults)
External Peripheral Communication Error	An input value (from a peripheral) is corrupted or not arriving
Signaling Error	An internal signaling (exception, interrupt, etc.) is wrongly raised or suppressed.
Execution timing Error	An error in the timing management (e.g. PLL) interferes with the correct execution timing.
Synchronization Error	An error in the scheduling processes causes an incoherent synchronization of processes/tasks.

The promise for the early evaluation of the software stack is to identify a virtualization environment able to develop software using a Virtual Instruction Set Architecture (VISA),

creating an additional layer between the software stack and the actual hardware architecture (see Figure 3). Both the Operating System and the Application Software shall be described according to this VISA. This additional level of abstraction will enable to investigate the error propagation properties of the software and to efficiently correlate them with errors arising in the actual hardware.

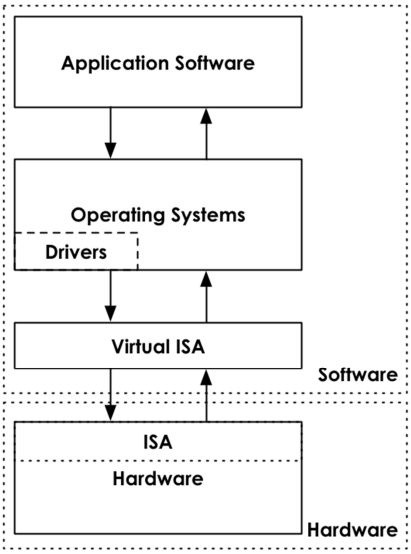

Figure 3: Virtualization of the Instruction Set Architecture

V. Conclusions

Reliability is a key challenge for the next generation computing systems, and its precise evaluation in the early stages of the design process is pivotal for the design of high optimized and efficient future systems.

From the analysis of the state-of-the-art in the area that has been presented in this paper, it is clear that current tools and models are still not mature to provide precise reliability evaluations for a large set of applications as the ones that will be expected in the upcoming computing continuum. By closing this gap, significant improvements in the products performance and quality will be expected. In particular, we expect that improvements in tools and methods for precise evaluation of the software impact on the hardware error propagation may strongly boost our ability of precisely assess the reliability of a full system and we expect to see resources and effort from several research groups driven toward this challenging research topic.

Acknowledgment

This paper has been fully supported by the 7th Framework Program of the European Union through the CLERECO Project, under Grant Agreement 611404. It has been also supported in part by EU's European Social Fund (ESF) and Greek national funds under the "Thales/HOLISTIC" project.

References

[1] D. Buchholz and J. Dunlop, "The future of enterprise computing: Prepare for compute continuum." [Available Online]: http://goo.gl/KYb0H8, May 2011.

[2] S. Guertin and M. White, "Cmos reliability challenges the future of commercial digital electronics and nasa," in *NEPP Electronic Technology Workshop*, 2010.

[3] CLERECO Consortium, "Cross-layer early reliability evaluation for the computing continuum official website." [Available Online]: http://www.clereco.eu, 2013.

[4] M. Omana, G. Papasso, D. Rossi, and C. Metra, "A model for transient fault propagation in combinatorial logic," in *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*, pp. 111–115, July 2003.

[5] A. Maheshwari, I. Koren, and N. Burleson, "Techniques for transient fault sensitivity analysis and reduction in vlsi circuits," in *Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on*, pp. 597–604, Nov 2003.

[6] K. Bowman, S. Duvall, and J. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *Solid-State Circuits, IEEE Journal of*, vol. 37, pp. 183–190, Feb 2002.

[7] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, pp. 900–907, Nov 2003.

[8] K. Mohanram and N. Touba, "Cost-effective approach for reducing soft error failure rate in logic circuits," in *Test Conference, 2003. Proceedings. ITC 2003. International*, vol. 1, pp. 893–901, Sept 2003.

[9] K. Mohanram and N. Touba, "Cost-effective approach for reducing soft error failure rate in logic circuits," in *Test Conference, 2003. Proceedings. ITC 2003. International*, vol. 1, pp. 893–901, Sept 2003.

[10] M. Reorda and M. Violante, "Accurate and efficient analysis of single event transients in vlsi circuits," in *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*, pp. 101–105, July 2003.

[11] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, pp. 389–398, 2002.

[12] S. S. Mukherjee, C. Weaver, J. S. Emer, S. K. Reinhardt, and T. M. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor.," in *MICRO*, pp. 29–42, ACM/IEEE, 2003.

[13] X. Li, S. V. Adve, P. Bose, and J. A. Rivers, "Softarch: An architecture level tool for modeling and analyzing soft errors.," in *DSN*, pp. 496–505, IEEE Computer Society, 2005.

[14] N. J. Wang, A. Mahesri, and S. J. Patel, "Examining ace analysis reliability estimates using fault-injection.," in *ISCA* (D. M. Tullsen and B. Calder, eds.), pp. 460–469, ACM, 2007.

[15] N. Soundararajan, A. Parashar, and A. Sivasubramaniam, "Mechanisms for bounding vulnerabilities of processor structures.," in *ISCA* (D. M. Tullsen and B. Calder, eds.), pp. 506–515, ACM, 2007.

[16] K. R. Walcott, G. Humphreys, and S. Gurumurthi, "Dynamic prediction of architectural vulnerability from microarchitectural state.," in *ISCA* (D. M. Tullsen and B. Calder, eds.), pp. 516–527, ACM, 2007.

[17] L. Duan, B. Li, and L. Peng, "Versatile prediction and fast estimation of architectural vulnerability factor from processor performance metrics," in *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*, pp. 129–140, IEEE, 2009.

[18] A. Biswas, N. Soundararajan, S. S. Mukherjee, and S. Gurumurthi, "Quantized avf: A means of capturing vulnerability variations over small windows of time," in *IEEE Workshop on Silicon Errors in Logic-System Effects*, 2009.

[19] X. Fu, J. Poe, T. Li, and J. A. B. Fortes, "Characterizing microarchitecture soft error vulnerability phase behavior.," in *MASCOTS*, pp. 147–155, IEEE Computer Society, 2006.

[20] A. Biswas, P. Racunas, R. Cheveresan, J. S. Emer, S. S. Mukherjee, and R. Rangan, "Computing architectural vulnerability factors for address-based structures.," in *ISCA*, pp. 532–543, IEEE Computer Society, 2005.

[21] A.Ma,Y.Cheng,andZ.Xing,"Accurateandsimplifiedpredictionofavf for delay and energy efficient cache design.," *J. Comput. Sci. Technol.*, vol. 26, no. 3, pp. 504–519, 2011.

[22] Y. Cheng, A. Ma, Y. Tang, and M. Zhang, "Accurate vulnerability estimation for cache hierarchy," in *Networked Computing and Advanced Information Management (NCM), 2011 7th International Conference on*, pp. 7–14, IEEE, 2011.

[23] X.Li,S.V.Adve,P.Bose,andJ.A.Rivers,"Onlineestimationofarchi- tectural vulnerability factor for soft errors," in *Computer Architecture, 2008. ISCA'08. 35th International Symposium on*, pp. 341–352, IEEE, 2008.

[24] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," *Proceedings of the 31st Annual International Symposium on Computer Architecture*, vol. 32, no. 2, p. 276, 2004.

[25] J.Shin,V.V.Zyuban,Z.Hu,J.A.Rivers,andP.Bose,"Aframeworkfor architecture-level lifetime reliability modeling.," in *DSN*, pp. 534–543, IEEE Computer Society, 2007.

[26] T. S. Rosing, K. Mihic, and G. D. Micheli, "Power and reliability management of socs.," *IEEE Trans. VLSI Syst.*, vol. 15, no. 4, pp. 391–403, 2007.

[27] L. Huang and Q. Xu, "Agesim: A simulation framework for evaluating the lifetime reliability of processor-based socs.," in *DATE*, pp. 51–56, IEEE, 2010.

[28] E. Humenay, D. Tarjan, W. Huang, and K. Skadron, "Impact of parameter variations on multicore architectures," in *Workshop on Architectural Support for Gigascale Integration (ASGI-06, held in conjunction with ISCA-33)*, 2006.

[29] B. Romanescu, S. Ozev, and D. Sorin, "Quantifying the impact of process variability on uniprocessor behavior," in *Workshop on Architectural Reliability*, 2006.

[30] R. L. Smith, "Statistics of extremes, with applications in environment, insurance, and finance," *Monographs on Statistics and Applied Probability*, vol. 99, pp. 1–78, 2004.

[31] T. M. Jones, M. F. O'Boyle, and O. Ergin, "Evaluating the effects of compiler optimisations on avf," in *Workshop on interaction between compilers and computer architecture (INTERACT-12)*, Citeseer, 2008.

[32] V. Sridharan and D. R. Kaeli, "Eliminating microarchitectural dependency from architectural vulnerability.," in *HPCA*, pp. 117–128, IEEE Computer Society, 2009.

[33] J. A. Butts and G. S. Sohi, "Dynamic dead-instruction detection and elimination.," in *ASPLOS* (K. Gharachorloo, ed.), pp. 199–210, ACM Press, 2002.

[34] A. Benso, S. Di Carlo, G. Di Natale, and P. Prinetto, "Static analysis of seu effects on software applications.," in *ITC*, pp. 500–508, IEEE Computer Society, 2002.

[35] V. Sridharan and D. R. Kaeli, "Using pvf traces to accelerate avf modeling," in *Proceedings of the IEEE Workshop on Silicon Errors in Logic-System Effects*, pp. 23–24, 2010.

[36] A. Savino, S. Di Carlo, G. Politano, A. Benso, A. Bosio, and G. Di Natale, "Statistical reliability estimation of microprocessor-based systems.," *IEEE Trans. Computers*, vol. 61, no. 11, pp. 1521–1534, 2012.

[37] V. Sridharan and D. R. Kaeli, "The effect of input data on program vulnerability," in *Workshop on System Effects of Logic Soft Errors (SELSE-5)*, 2009.

[38] [38] N. J. Wang, J. Quek, T. M. Rafacz, and S. J. Patel, "Characterizing the effects of transient faults on a high-performance processor pipeline.," in *DSN*, pp. 61–, IEEE Computer Society, 2004.

[39] [39] G. P. Saggese, N. J. Wang, Z. Kalbarczyk, S. J. Patel, and R. K. Iyer, "An experimental study of soft errors in microprocessors.," *IEEE Micro*, vol. 25, no. 6, pp. 30–39, 2005.

[40] [40] D. Thaker, D. Franklin, J. Oliver, S. Biswas, D. Lockhart, T. Metodi, and F. Chong, "Characterization of Error-Tolerant Applications when Protecting Control Data," in *Workload Characterization, 2006 IEEE International Symposium on*, pp. 142–149, IEEE, 2006.

[41] [41] S. Z. Shazli, M. A. Abdul-Aziz, M. B. Tahoori, and D. R. Kaeli, "A field analysis of system-level effects of soft errors occurring in microprocessors used in information systems.," in *ITC* (D. Young and N. A. Touba, eds.), pp. 1–10, IEEE, 2008.

[42] [42] M. Rahman, B. R. Childers, and S. Cho, "Stealth works: Emulating memory errors," in *Proceedings of the First International Conference on Runtime Verification*, RV'10, (Berlin, Heidelberg), pp. 360–367,

[43] Springer-Verlag, 2010.[43] S. Pan, Y. Hu, and X. L. 0001, "Ivf: Characterizing the vulnerability of microprocessor structures to intermittent faults.," in *DATE*, pp. 238–243, IEEE, 2010.

[44] A. Messer, P. Bernadat, G. Fu, D. Chen, Z. Dimitrijevic, D. J. F. Lie, D. Mannaru, A. Riska, and D. S. Milojicic, "Susceptibility of commodity systems and software to memory soft errors.," *IEEE Trans. Computers*, vol. 53, no. 12, pp. 1557–1568, 2004.

[45] [45] M. Breuer, "Multi-media applications and imprecise computation," in *Digital System Design, 2005. Proceedings. 8th Euromicro Conferenceon*, pp. 2–7, IEEE, 2005.[46] X. Li and D. Yeung, "Application-level correctness and its impact on

[46] fault tolerance," in *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*, pp. 181–192, Feb 2007.

[47] L. Leem, H. Cho, J. Bau, Q. A. Jacobson, and S. Mitra, "Ersa: Error resilient system architecture for probabilistic applications.," in *DATE*, pp. 1560–1565, IEEE, 2010.

[48] P. Racunas, K. Constantinides, S. Manne, and S. S. Mukherjee, "Perturbation-based fault screening.," in *HPCA*, pp. 169–180, IEEE Computer Society, 2007.

[49] N.J.WangandS.J.Patel,"Restore:Symptom-basedsofterrordetection in microprocessors.," *IEEE Trans. Dependable Sec. Comput.*, vol. 3, no. 3, pp. 188–201, 2006.

[50] P. Ramchandran, S. Adve, V. Adve, and Y. Z. M.-L. Li, "Swat: An error resilient system," in *4th Workshop on Silicon Errors in Logic - System Effects (SELSE - IV)*, 2008.