Engineering of Cross-Layer Fault Tolerance In Multiprocessing Systems

Junchao Chen[1], Milos Krstic[1, 2]
[1]*IHP, Frankfurt (Oder), Germany*
[2]*University of Potsdam, Germany*
Email:{chen,krstic}@ihp-microelectronics.com

***Abstract* – The increased design complexity and appearance of emerging embedded systems are leading to more pronounced challenges related to errors. Traditional approaches for addressing faults include the redundancy approaches in hardware, time, software, and /or information. But the overhead of these methods is not acceptable for many mixed-criticality applications, and consequently, we need some means to achieve the dynamical trade-off in safety, reliability, performance and power consumption. This paper presents the initial steps of a PhD work focusing on exploring the adaptive use of the fault tolerance mechanisms in multiprocessing architectures and development methods for adaptive cross-layer optimization approaches.**

I. Introduction

Fault tolerance is the property that keeps the system not deviating from the correct operation in the presence of faults [1]. Along with the technology scaling, increasing gate complexity could be integrated into a single chip. Today's embedded systems are susceptible to faults from various sources, i.e. radiation particles, voltage variations, crosstalk, technology defects, etc. Moreover, the failure mechanisms of radiation-induced soft-errors, circuit ageing effects, early-life failures and variability become critical [2]. The high energy particles can cause the radiation-included soft errors by inducing bit flips in the registers or logic [3]. Generally, radiation-induced soft-errors are mainly relevant for mission-critical systems such as space, avionics or certain military applications, but with technology scaling and the decrease of critical charge are starting to be important also for terrestrial applications. Additionally, ageing effects degrade the circuit performance over time, and even eventually the permanent internal faults could occur. There are significant ageing effects in CMOS ICs that are related to the degradation of the gate-oxide, where Bias Temperature Instability (BTI) [5], Hot Carrier Injection (HCI) [6] and Time-Dependent Dielectric Breakdown (TDDB) [5] are the dominant drift-related ageing effects. Moreover, there are early-life failures caused by defective ICs which pass manufacturing tests but fail in the infant mortality period. Furthermore, the burn-in tests for screening early-life failures are becoming more and more important but also make the tests more expensive [9]. The errors could also be generated due to the variability, and according to [7][8], the major concerns come from threshold voltage variations, channel length variations and voltage/thermal variations, etc.

There are a lot of publications on the techniques to address the above issues, and most of them focus on the improvement on single abstraction layers in the system design. Instead of improving the efficiency of the fault tolerance mechanisms in the individual layers, the more promising way to achieve an effective and reliable system operation is to utilize the different available approaches or parameters at multiple abstraction layers and combine them to optimize the design in a cross-layer manner. For example, the structural integrity checking [11], addressing both architecture and software layers, can have less hardware cost than the traditional code checkers. Also, the error detection (e.g. using logic parity checking and residue code) and instruction-level retry [12] include improvements at the circuit, logic and architecture layers, and this approach can lead to higher reliability features of the RISC processors. Cross-layer fault tolerance systems have the potential to achieve higher performance, more reliable operation, lower cost and lower power consumption by taking advantage of the information and capabilities available across different layers in the system stack [10].

However, it's a challenge to develop an integrated cross-layer fault tolerance system concept. Not only that designer needs to develop the techniques for breaking through the current abstraction layers, but it also requires to perform comprehensive and thorough analysis and optimization of the existing techniques at the different layers. This paper reports the initial investigations in the author's PhD research in the field of evaluating and developing methods to achieve cross-layer fault tolerance and try to integrate and characterize these methods into an adaptive multiprocessing platform which has been developed at IHP [26].

The paper is organized as follows. Section II introduces the methods for the layered fault tolerance. Section III presents an overview of the current technologies in cross-layer fault tolerance. Section IV studies the multiprocessing architecture. Section V presents the ongoing and future work. The conclusions are summarized in Section VI.

II. Single-Layer approaches

Firstly, let us look at the basic concepts of fault tolerance. Fault tolerance is a way to exploit and manage redundancy [1] to mask faults or errors when they appear. Basically, there are four types of redundancies: hardware, software, information and time. Hardware redundancy is achieved by involving extra hardware to detect or mask the effects of the failed component.

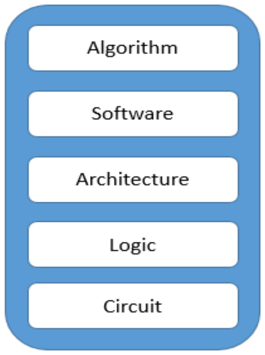

Fig. 1 Abstraction layers of a computing system

The well-known example is the triple modular redundancy (TMR), in which three components perform the same operation and a voter is selecting an output. However, the hardware redundancy always incurs high overhead. The information redundancy is usually applied with the error detecting and correction coding approaches, such as adding one or more extra check bits to the original data. Nowadays, the information redundancy is widely used in various storage devices, due to their regular structure. Time redundancy attempts to reduce the amount of extra hardware by adding extra computational time. It is effective for the mitigation of transient faults through re-execution the same program on the same hardware. Finally, the software redundancy attempts to mitigate the software failures.

Every computing system can be abstracted into several layers (Figure 1). Traditionally, the faults are dealt in the corresponding abstraction layers where they are detected. The following list summarizes the different important techniques for fault tolerance corresponding on different layers:

- **Circuit layer:** At the lowest layer, faults are depending on the circuit design and basic physical properties [15]. As one of many examples of such techniques, hardened flip-flop design called LEAP-DICE [16] uses a Dual-interlocked cell (DICE) and Layout design through Error Aware transistor Positioning (LEAP) technology. The LEAP-DICE is able to tolerate single-event upsets (SEUs) and single-event multiple upsets (SEMUs) at the nominal or near threshold voltage. The use of this technique allows a dramatic decrease of the single-event error rates without significant influence on the performance. Nevertheless, the drawback is the incensement of the power and area overheads (mainly due to complex DICE architecture)
- **Logic layer**: In this level, the digital system is assembled with various gates and memory elements, and the signals are represented as binary values. As an example, the SRAM-based caches and DRAM-based main memory usually use the Error Correcting Codes (ECC) [17] to protect their data. The various Single-bit-Error-Correcting and Double-bit-Error-Detecting (SEC-DED) codes are widely used in many systems. Additionally, the Double-bit-Error-Correcting and Triple-bit-Error-Detecting (DEC-TED) codes are the robust way to cope with multi-bit errors. However, depending on the implementations, the ECC induces significant storage overheads.
- **Architecture layer**: Here the faults are typically detected as observed changes in one or more modules' behavior. The nice example is the Dynamic Implementation Verification Architecture (DIVA) [18] presents a monitor core which is a specialized checker in the cores to validate executed instructions. Furthermore, this novel technique can dramatically reduce the error rate in microprocessor design. On the other hand, the costs associated with the DIVA checker are silicon area, power consumption and slowing down the core processor.
- **Software layer**: Software-based fault tolerances typically duplicate the code and data segments, run and compare the results. One example is the Duplication with Comparison (DWC) technique. This technique can detect the error by executing an algorithm twice and compare the result. However, this method cannot identify the correct result [14]. Error Detection by Duplicated Instructions (EDDI) can duplicate instructions via compilation and arrange the different registers and variables for the duplicated instructions. Additionally, it does not need extra hardware overhead [14].
- **Algorithm layer**: The Algorithm-Based Fault Tolerance (ABFT) can detect or correct faults by modifying the algorithms [19]. For example, in the paper [20], the author presents an ABFT solution for dynamic molecular applications. This algorithm can map the kernel to a matrix and recover from the error state after it is detected. Furthermore, this application is able to recover from the latest checkpoint or repeating the corrupted computation.

The substantial disadvantage of the layered approach is that the different system layers are considered separately. Due to the fact that upper layer cannot specify the requirements and achieve the cooperation with the lower layer, the unnecessary error correction and resource underutilization is possible.

III. CROSS-LAYER APPROACHES

In contrast to the traditional single-layer fault tolerance approach, the cross-layer way can provide better performance and effectiveness for the system [10]. Since the layer barriers are not a limitation for fault-mitigation, the implementation of some appropriate combination of methods to meet the design constraints is possible. However, performing the cross-layer approaches requires broad knowledge and understanding of the whole system, such as when and where faults appear, how faults can generate errors, how errors can propagate and be mitigated across layers, and how errors impact the system performance.

One of the initial studies of cross-layer approaches is provided in [25]. The authors use the Simulated Annealing optimization algorithm to evaluate the overall system reliability by selecting the best-identified combination of software and hardware components, with a certain cost constraint. Although this technique can be efficient and generate the satisfactory results when the difficult-to-satisfy restrictions are needed, the reliability models are too idealistic. Only one single failure probability is considered for each component, and the cost function is merely the sum of cost in each element.

The inter-layer information flow can make the runtime performance (such as power, ageing) analysis possible, and let the investigation of cross-layer and dynamic runtime adaptation techniques possible. In [21], the authors show a cross-layer ageing analysis platform, which can perform the ageing modelling, simulation and mitigation by using multi-objective cross-layer approaches. Two platforms are proposed, and these platforms cover the circuit, logic and architecture layers. The first platform proposes an inaccurate ageing model, which is developed at the architecture level. It takes architecture information of power, temperature (collect from the on-die power and thermal sensors) and usages of architecture blocks (e.g. ALU, decoder) as input. The ageing model can estimate the ageing rate of different block and the usage (switching activity, ON time, OFF time) of each block. However, the above model is hard to evaluate circuit-level ageing mitigation techniques. Then, the authors propose an RTL-platform, which is based on EDA (electronic design automation) process such as circuit simulators, gate-level ageing models, etc. This platform allows very accurate analysis of power, ageing and area cost, but the flexibility compared to the first approach is significantly reduced.

Design of cross-layer fault tolerance system doesn't need to develop new fault tolerance techniques. It can be achieved by combining the existing techniques related to different layers. Cross-Layer Exploration for Architecting Resilience (CLEAR) [10] is the first framework to propose an impressive and accurate simulation campaign exploring how to achieve desired reliability goals with the minimal costs (area, execution time, power, energy) through combining resilience techniques [26] on different abstraction layers. It can explore the vast space of comprehensive resilience techniques automatically and systematically, and form 586 cross-layer combinations across different layers in the system stack. In total, ten different error detection and correction techniques and four hardware error recovery techniques are used in the resilience library for different layers. Finally, the top-down approach, which resilience techniques are applied at highest layer firstly, is used to achieve the cost-effectiveness of various combinations. The result shows that a proper combination of hardening circuit-level, parity checking in logic-level, and micro-architectural recovery for the general-purpose processor core can achieve a high cost-effective radiation-induced soft errors resilience method. However, if every new product is performing the same simulation, the cost would be high in the early stage. Currently, this framework only concentrates on radiation-induced soft errors.

IV. FAULT TOLERANCE IN MULTIPROCESSING SYSTEMS

A single computer system with more than one Central Processing Units (CPUs) that share the various hardware resources is called multiprocessing system, or, multiprocessor [13]. The concept of multiprocessing system has been known for decades. In the last years, multiprocessing system has become the primary architecture because of two reasons. Firstly, the performance of single processors has already reached the upper limit, which is the point of diminishing returns. Secondly, because of the excessive power consumption, the working frequency cannot be increased [22]. Nowadays, the use of the multiprocessing architectures starts to be the dominant trend in all computing segments, such as desktop, server and embedded. Because of these computing segments can present an excellent processing power to programs, which could be divided into smaller procedures and processed in parallel. Moreover, the multiprocessing architecture can also have the advantages of high throughput, high energy efficiency, long-lasting battery life [27] etc.

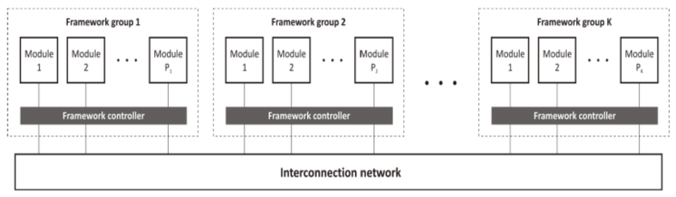

Fig. 2 General architecture of framed multiprocessor

In [22], a flexible and scalable multiprocessor architecture framework is proposed, which can dynamically configure its properties concerning performance, power consumption and dependability. Figure 2 shows the multiprocessing platform which has been developed at IHP. The main idea of this framework is to dynamically adapt the fault tolerance and performance under the constraints of the system ageing and power dissipation. In order to dynamically change the reliability and performance requirements of multiprocessor applications, three basic multiprocessor operation modes (de-stress, fault-tolerance, high-performance) are implemented:

- De-stress mode: The goal of this mode is to increase the multiprocessor lifetime and reduce the power consumption. As a consequence, a minimum required the number of multiprocessor cores is active, while all the others are inactive. The IC ageing monitors could supply the ageing information to core gating patterns. These patterns can systematically power- or clock-off cores in this multiprocessor.
- Fault-tolerance mode: In this mode, the multiprocessor cores are used to form core-level NMR (N-modular redundancy) system, in order to

increases error resilience. Therefore, the entire cores are synchronized to run the same task concurrently. Furthermore, a core-level programmable NMR voter is designed to let the cores vote on each clock cycle. This core-level NMR can mask faults without requesting recovery procedures.

- High-performance mode: In this mode, the multiprocessor boosts multiprocessor performance and act the same as a conventional multiprocessor.

These modes can dynamically be changed under the actual application requirements. Furthermore, this multiprocessor framework consists of:

- Framework controllers: This is the hardware part of this framework, and it's in charge of coupling/decoupling the modules to/from the power supply or clock and forms NMR systems.
- Framework middleware: This is a layer to hide the hardware details and offer the services to the application layer.
- Application layer: This part can call the middleware routines, in order to program the framework controllers and read their status.

The interconnection network can enable scalability and redundant links for these modules. And the modules could be the processing elements, cores with or without cores or memory modules.

A novel environment for automated fault injection and a novel multiprocessor verification platform are used to evaluate this multiprocessing platform. Furthermore, a novel lifetime evaluation approach based on the Weibull distribution shows the advantages of using the core gating patterns can increase the system's lifetime for over 30% compared to the traditional Round-Robin approach.

V. ONGOING AND FUTURE WORK

The ongoing research activities have been focused on the above multiprocessing platform with the additional cross-layer fault tolerance mechanisms, and evaluation the benefits and costs.

The above multiprocessing platform could be deployed in a harsh environment which could include the effects of high energy particles and the expected lifetime could be long. In order to manage available hardware resources and monitor operating environment of the chip, the additional on-chip sensors are required, such as more accurate ageing monitor, voltage and temperature sensors. Such sensors must be seamlessly integrated into the operation of a complex multi-processor system. In the applications where soft errors, induced by radiation, play a significant role, the detection of such errors need to be monitored as well. Moreover, the period of high flux radiation in space could be quite long [24]. Consequently, the integration of an on-chip radiation sensor to trigger the dynamic hardening of the system could increase the reliability.

The optimized task scheduling and adaptive mode switching could be an effective method to increase the lifetime and overall energy consumption in the software layer. These methods could be carefully tailored to fit to dynamically changing reliability requirements of the application. Moreover, exploiting a novel compiler which can automatically redundancy some critical instruction and adapt to the compiler level multiprocessing operation.

In order to correctly and adequately operate and adjust the system, it is crucial to characterize different settings concerning power, performance, and reliability features. Considering that there is a countless number of possibilities for such settings, the modelling of the parameters based on the characterized and mathematical model, such as a system-level Bayesian reliability estimation model [28], could help us to estimate the features of the not fully characterized states. Moreover, the functional relationship between the parameters and the different technologies across different abstract layers should be investigated.

More modalities can be developed, such as mixed mode which one core work in de-stress mode and the other cores work in fault tolerance mode. Furthermore, some spare module with particular technologies, such as DICE-LEAP design, adaptive voltage scaling can be deployed in the platform.

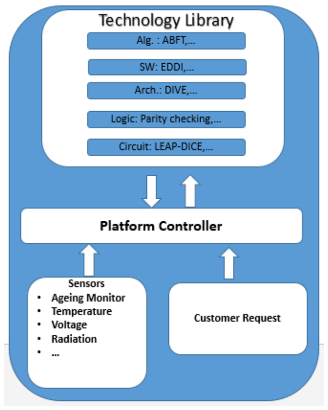

Fig. 3 The conception of the cross-layer fault-tolerance framework

Breaking the abstraction layers divisions will lead to the possibility of building a cross-layer fault tolerance framework, which can comprehensively introduce the above or other techniques corresponding to different layers. Each technique can be modularized, so it is feasible to operate these modules separately and dynamically. This could include activation and inactivation of the module (in hardware or software) at the various levels during the runtime. The integrated platform can automatically explore the different modules combination across multiple layers, in order to achieve the different constraints or targets during operation.

VI. CONCLUSIONS

In the previous sections, we have presented the basic concepts of simple and cross-layered fault tolerance approaches, as well as a baseline adaptive multiprocessing platform. Our short-term plans to verify the ideas presented in the above section. In particular, we would like to characterize and model different schemes of cross-layer fault tolerance, and implement and evaluate it in the provided multiprocessor platform.

ACKNOWLEDGMENT

This work has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 722325.

REFERENCES

[1] Israel Koren, C. Mani Krishna, “Fault-Tolerant Systems”.

[2] S. Mitra et al., "The resilience wall: Cross-layer solution strategies," Proceedings of Technical Program - 2014 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA), Hsinchu, 2014, pp. 1-11.

[3] N. Seifert et al., “On the radiation-induced soft error performance of hardened sequential elements in advanced bulk CMOS technologies”, in IEEE International Reliability Physics Symposium (IRPS), 2010, pp. 188–197.

[4] N. N. Mahatme, N. J. Gaspard, S. Jagannathan, T. D. Loveless, B. L. Bhuva, W. H. Robinson, L. W. Massengill, S. J. Wen and R. Wong, "Impact of Supply Voltage and Frequency on the Soft Error Rate of Logic Circuits," IEEE Transactions on Nuclear Science, vol. 60, no. 6, pp. 4200- 4206, Dec 2013.

[5] Linder, B.P., et al. , "Improving and Optimizing Reliability in Future Technologies with High-K Dielectrics, " Proc. Symp. VLSI Technology, Systems, and Applications (VLSI-TSA) , pp. I-4, 20 1 3.

[6] Ramey, et al. , "Intrinsic Transistor Reliability Improvements from 22nm Tri-Gate Technology," Proc. inti. Reliability Physics Symp. , pp. 4C.5. 1 -4C. 5. 5, 20 1 3.

[7] Joshi, R., et al. , "Variability Analysis for Sub - 1 00nm PD/ SOI SRAM Cell," Proc. European Solid State Circuit Con!, pp.2 1 1 -2 1 4, 2004.

[8] Tschanz, J. w., et al. , "Tunable Replica Circuits and Adaptive Voltage-Frequency T echniques for Dynamic Voltage, Temperature, and Aging Variation Tolerance," Proc. Symp. VLSI Circuits, pp. 1 1 2-1 1 3, 2009.

[9] Vassighi, A., et al. , "Characterizing Infant Mortality in High Volume Manufacturing," Proc. Inti. Reliability Physic Symp. , pp. 7 1 7-7 1 8, 2008.

[10] E. Cheng et al., "Tolerating Soft Errors in Processor Cores Using CLEAR (Cross-Layer Exploration for Architecting Resilience)," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[11] D. J. Lu, "Watchdog Processors and Structural Integrity Checking," IEEE Transactions on Computers, Vols. C-31, no. 7, pp. 681-685, July 1982.

[12] H. Ando, Y. Yoshida, A. Inoue, I. Sugiyama, T. Asakawa, K. Morita, T. Muta, T. Motokurumada, S. Okada, H. Yamashita, Y. Satsukawa, A. Konmoto, R. Yamashita and H. Sugiyama, "A 1.3 GHz fifth generation SPARC64 microprocessor," in Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC. 2003 IEEE International, 2003.

[13] M. Ebbers, J. Kettner, W. O’Brien, B. Ogden, „Introduction to the New Mainfame z/OS Basics“

[14] J. F. Tarrillo, C. A. Lisboa, L. Carro, C. Argyrides and D. K. Pradhan, "Evaluation of a new low cost software level fault tolerance technique to cope with soft errors," *2010 11th Latin American Test Workshop*, Pule del Este, 2010, pp. 1-3.

[15] V. P. Nelson, "Fault-tolerant computing: fundamental concepts," in Computer, vol. 23, no. 7, pp. 19-25, July 1990.

[16] L. Hsiao-Heng Kelin et al., "LEAP: Layout Design through Error-Aware Transistor Positioning for soft error resilient sequential cell design," 2010 IEEE International Reliability Physics Symposium, Anaheim, CA, 2010, pp. 203-212.

[17] S. S. Sahoo, B. Veeravalli and A. Kumar, "Cross-layer fault-tolerant design of real-time systems," 2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Storrs, CT, 2016, pp. 63-68.

[18] T. M. Austin, "DIVA: a reliable substrate for deep submicron microarchitecture design," MICRO-32. Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture, Haifa, 1999, pp. 196-207.

[19] Huang, K.-H. and J. A. Abraham, “Algorithm-based fault tolerance for matrix operations,” IEEE Trans. Computers, 1984.

[20] J. Liu and G. Agrawal, "Algorithm Level Fault Tolerance for Molecular Dynamic Applications," 2015 IEEE 22nd International Conference on High Performance Computing (HiPC), Bangalore, 2015, pp. 406 415.

[21] F. Oboril and M. B. Tahoori, "Cross-layer approaches for an aging-aware design of nanoscale microprocessors: Dissertation summary: IEEE TTTC E.J. McCluskey doctoral thesis award competition finalist," 2015 IEEE International Test Conference (ITC), Anaheim, CA, 2015, pp. 1-10.

[22] A. Simevski, "Architectural framework for dynamically adaptable multiprocessors regarding aging, fault tolerance, performance and power consumption", 2014

[23] M. Schölzel, "Self-Testing and Self-Repairing Embedded Processors: Techniques for Statically Scheduled Superscalar Architectures ", 2014

[24] G. Tsiligiannis et al., "An SRAM Based Monitor for Mixed-Field Radiation Environments," in IEEE Transactions on Nuclear Science, vol. 61, no. 4, pp. 1663-1670, Aug. 2014.

[25] [25] N. Wattanapongsakorn and S. P. Levitan, "Reliability optimization models for embedded systems with multiple applications," in *IEEE Transactions on Reliability*, vol. 53, no. 3, pp. 406-416, Sept. 2004.

[26] S. R. Nassif, N. Mehta and Y. Cao, "A resilience roadmap," 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), Dresden, 2010, pp. 1011-1016.

[27] https://www.ihp-microelectronics.com/

[28] R. Ramanathan. Intel multi-core processors: Making the move to quad-core and beyond. Technology@Intel Magazine, Dec 2006.

[29] A. Savino, A. Vallero and S. Di Carlo, "ReDO: Cross-Layer Multi-Objective Design-Exploration Framework for Efficient Soft Error Resilient Systems," in *IEEE Transactions on Computers*.