System-level analysis of soft error rates and mitigation trade-off explorations

Zhe Ma*, Francky Catthoor*, Frank Vermunt† and Teun Hendriks‡
* IMEC, Kapeldreef 75, 3000 Leuven, Belgium
† NXP Semiconductors, Nijmegen, The Netherlands
‡ Embedded System Institute, Eindhoven, The Netherlands

***Abstract*—This paper presents a novel system-level analysis of soft error rates (SER) based on the Transaction Level Model (TLM) of a targeted System-On-a-Chip (SoC). This analysis runs 1000x faster than the conventional SoC analysis using a gate-level model. Moreover, it allows accurate prediction in the early design phase of a SoC, when only limited application details are available. Preliminary validation results from accelerated SER tests on the physical system have shown that the analysis can predict the SER with a reasonable accuracy (within 5x of the results from tests on physical systems). This system-level analysis is particularly suitable to handle the black-box models for industrial semiconductor IP libraries. Based on this system-level analysis, we also propose a SE mitigation solution using selective protection of SRAM of a SoC. This solution provides a series of trade-offs between the system dependability and cost (in terms of silicon area).**

I. Analysis technique

Our analysis provides estimation of SER at the system-level for embedded systems. Such embedded systems usually include several DSP processors, hardware accelerator IPs and a relatively large on-chip SRAM. SE can take place in both on-chip memories (SRAM) and flip-flops. When data-intensive applications are running on such systems, the SEs originating in an unprotected SRAM have a dominant influence on the system-level failures (as reported for 65-nm or larger CMOS technology by [1]). Therefore, our analysis presented in this paper focuses on the SE from SRAM.

A. System-level SE derating ratio and its relevance to dependability cost trade-offs

The SER at system-level is typically much lower than the SER at the memory level, this effect is referred to as SE derating. This derating is similar to the Timing Derating and the Logic Derating [7] at the circuit level (also known as the TVF (Timing Vulnerability Factor) and AVF (Architecture Vulnerability Factor), see [5]), but at a higher level of abstraction. That is, our system-level SE derating is concerned with the data transfers and consumptions in memories. At the circuit level, the TVF is caused by the fact that a circuit element is not susceptible to SE in its entire clock cycle; while the AVF is caused by the fact that all circuit elements in a chip are not working simultaneously. At the transaction level modeling, the primary reasons for our system-level SE derating are that (1) only corrupted data that have yet to be consumed could cause a system failure and (2) algorithms/data-structures used in the applications may have inherent fault tolerance. It is important to have an accurate estimation of the SER at the system-level for a product with high dependability requirements. Such an estimation must be able to handle different working modes of the target system (e.g. AM mode vs. FM mode for a radio system) as very different SERs at the system-level could be exhibited when a system works with different modes [11] (this phenomenon is opposite to the cases of scientific applications on general purpose CPUs as reported by [9]). In practice, the SER at the system-level can be expressed as the product of the raw SER in memories and the derating ratio of a specific system. For example, if the raw SER of the SRAM used in a target embedded system is 100 FIT (Failure In Time), and the derating ratio of this system is 5%; then the SER at the system-level is 5 FIT. The essential purpose of our SE analysis is to give the SE derating ratio of the system running with a real-life workload; then a system architect is able to use this ratio and the raw SER at the memory level to calculate the user-perceived SER at the system-level.

For products under well defined levels of safety regulations (e.g. automotive electronics), such system-level SER estimation is crucial at an early stage of system design to allow for a trade-off between the final product's dependability and its cost. For the SER of a system running its real-life workload, NXP has developed its estimation model based on empirical data as well as accelerated tests on physical systems [11]. Nevertheless, as systems become more complex and workloads become more dynamic, the accuracy of estimations based on empirical data becomes less reliable. In the meantime, to use SE counter measures in a cost-effective manner requires an early identification of a system's SE vulnerability. Therefore, our high-level analysis based on a Transaction Level Model (TLM) becomes increasingly important.

B. Related analysis techniques

Many techniques exist at different abstraction levels to estimate the SER. For memories, SER is usually derived by testing the memory alone without running the functionalities on processor or other hardware IP blocks. Many raw SE at the memory system level are eventually filtered out [3]. When the user-experienced SER at the system level should be identified for a specific system, the complete system must be analyzed. If the target system has already been fabricated, then a physical system running the real-life workload can be tested

with SER acceleration by applying radioactive sources [3]. If SER acceleration by a radioactive source is unavailable or inconvenient, a random fault injection can be used to accelerate the test [8], [10]. This fault injection is a cheaper alternative than a real test. But it requires knowledge of the raw SER in the memories. It also requires the instrumentation of the memory of a system such that memory faults can be injected at run-time when the system is executing. When the size of the memories of the system under test is large, such random fault injections at a low abstraction level require a long process. When the physical system is unavailable, then the analysis can be performed by either a hardware architecture-level emulation [6] or a gate-level emulation [2]. The system models from both levels are available later than TLM which is a higher-level abstraction model.

C. A fast and accurate high-level SER analysis

Our high-level analysis provides a fast TLM simulation of a system. A system is modeled as a set of connected components. It treats all components as black-boxes and hence no detailed specifications are required inside each component. This is important from an industrial point of view as some IP blocks are only available as black-boxes. We are mostly interested in the I/O and (real-)time behaviors of each component when memory faults are injected, and we will simulate the propagations of errors in a system. This high-level analysis provides a balance between modeling a system entirely as a black-box and modeling a system as a white-box with all intra-component details.

D. Required input for the high-level SER analysis

We work with a customized transaction level model [4] called Thread Node Graph (TNG). A TNG is a directed acyclic graph where each vertex is a Thread Node (TN) and each edge between two TNs represents a data transfer.A TN is the basic unit for computation in our model and hence cannot be divided further. Each TN represents a self-contained function that can be executed without calling functions in other TNs. A TN can start after all data associated with the incoming edges are ready; and it finishes with all data associated with its outgoing edges become ready. Therefore, an edge between two TNs represents an ordering relation that the succeeding TN can only start when the preceding TN has finished. Each schedule has the timing information on the executions of TNs, including start/end times and memory accesses schedules. Such schedules present a deterministic view on the reads and writes to the memories. A valid schedule of a TNG is defined as one in which each TN is executed once and all orders on TNs are satisfied. A target embedded system can be specified with one or many different TNGs; and a TNG can have different schedules.

To evaluate the SER at system-level, we first need to have the raw SER at the memory level to calculate the memory faults number. Each memory fault is a single bit error taking place in a SRAM. The number of SE-induced memory faults in a particular memory can be calculated as:

$$MemFaults = SER \times MemSize \times ElapsedTime \quad (1)$$

In most cases, the raw SER is determined by whether ECC (Error Correction Code) protection is used for a SRAM with a specific process technology.

The vulnerability of each TN (as a black-box) to the SE-induced data corruptions is measured by both its failure probability to bit errors and its silent data corruption probability to bit errors.

The TN failures caused by memory faults from a particular memory is:

$$TNFailures = TNFailProbability \times MemFaults \quad (2)$$

where $MemFaults$ is the number of memory faults from formula (1). This $TNFailProbability$ can be identified by fault injection simulations in Instruction-Set Simulator (ISS) for software IPs or in VHDL simulator for hardware IPs. And a TN can have different failure probabilities to bit errors from different data objects (e.g. input buffer, internal data or control parameters).

In addition to the failure probabilities, we also need the probabilities of silent data corruption at the output data. These probabilities can be obtained in the similar manner as the failure probabilities. The only difference is that for silent data corruptions, we need to calculate them for each output data object. For a TN with a set of output data objects $D = \{d_0, d_1, ...\}$, we define $\forall d_i \in D$,

$$\begin{aligned} DataCorruptions_i = {} & CorruptionProbability_i \\ & \times MemFaults \end{aligned} \quad (3)$$

II. Selectively protecting memory/data

Once a baseline analysis (without any counter measures against SE) is complete, the designers can find a derating ratio of their system and calculate the SE-induced system failure rate. If the failure rate is higher than an acceptable number, designers can use this high-level analysis to have a quick evaluation of trade-offs between the failure rate and the size of protected memories. Our high-level analysis allows a user to specify the SER at the granularity of a TN's memories. The designers can specify a set of memory address ranges for a TN and give either a raw SER or a SER with memory-protecting for each range. Due to the combinatorial nature of selective memory protection, further optimizations are still required to have a scalable exploration with a large number of TNs. Our experiments have shown that when the number of TNs is 10 or less, and the number of memory address ranges of each TN is 3 or less, our high-level analysis can complete the exploration of selective protection in several minutes on an ordinary desktop PC.

In Section III, we will show a simple case where SRAMs can be protected by ECC. And our analysis can help to find a number of optimal selectively protecting configurations.

III. Experimental results

A. Baseline analysis for NXP car audio processing system

We show a simple case study of applying our high-level analysis on a audio processing system. The TNG of this audio processing system is depicted in Fig.1. The bubbles are TNs. Each TN is annotated with numbers of memory accesses and execution times (not shown in the figure). This system is running on a single DSP processor with two SRAMs: a data memory and a coefficient memory (the program memory is immune to SE and hence is excluded in this case study). The numbers of read and write accesses of each individual TN for a single invocation are measured. Also, the execution times (in terms of clock cycles at a clock frequency of 130MHz) of TNs are measured on the physical systems.

Our analysis calculated a SE derating ratio of 5% for TN failures at system-level. This means that for every 100 SEs in the memory used by this audio processing system, only 5 TN failures would take place. In addition to this derating ratio of TN failures, our analysis also provided the probabilities of (SE-induced) bit errors in the memory address ranges of TN output buffers. The preliminary validations based on accelerated tests of the physical system have shown that our 5% derating ratio is relatively close to test results.

B. Selectively protecting memories

The on-chip SRAM can be partly protected by an Error Detection And Correction (EDAC) method based on the Hamming code with a minimum distance of 4. This EDAC method is also known as the Single Error Correction-Double Error Detection (SEC-DED) Error Correction Code (ECC). In general, increasing the percentage of ECC protected on-chip SRAMS can reduce the probability of SE-induced chip failures. Since ECC protection incurs a SRAM area overhead, a chip architect may need to have SRAMs partly protected to obtain an optimal trade-off between the silicon area cost and the reliability of a target chip. Our analysis of system-level failure rate can identify such trade-offs. The workflow of this identification is actually an iteration of many system-level failure rate analyses. Each analysis gives a failure rate for an unique configuration of partial ECC protection. If an architect can list all relevant configurations of partial ECC protection, then this iteration of our workflow can automatically go through the list and reports a failure rate for each configuration. At the end, the workflow can prune all non-Pareto optimal trade-offs and reports a Pareto curve which contains the optimal trade-off points of all explored configurations. The overall workflow for this exploration is depicted in Fig. 2.

We have explored the trade-offs between the dependability cost (in terms of protected memory) and the system failure rate for the target audio processing system. The pruned trade-offs from the exploration are shown in Fig.3. Each point in the figure represents a specific allocation of ECC-protection and the corresponding failure rate. This result has identified that when the protected SRAM area grows from 0 to 15kbit (extra silicon area due to the protection is about 13% of the protected area), this audio processing system's failure rate drops from 5% to 0.5% (the remaining 0.5% is due to the fact that communicating buffers between TNs are not protected).

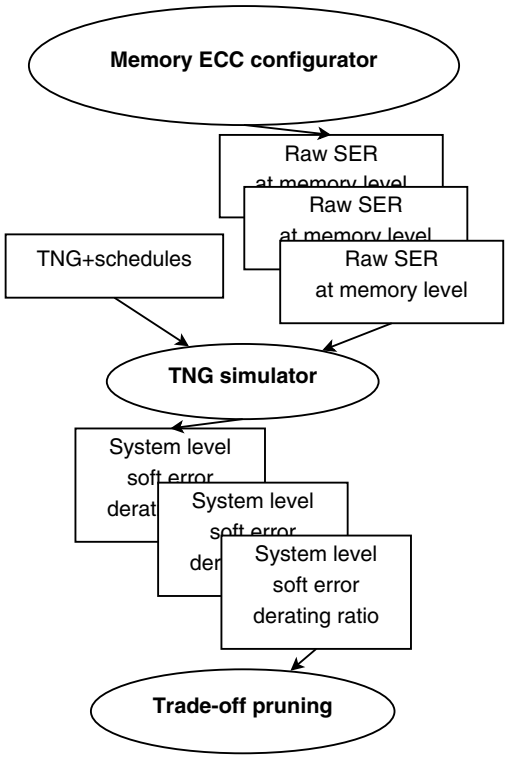

Figure 2. Workflow of the selectively memory protection trade-offs exploration

IV. Conclusions

We have presented a system-level SER analysis technique for predicting the user-perceived SER of a complex SoC. Running with a TLM model of a targeted system, our technique can predict the system's vulnerability to SE in a few minutes. Based on this analysis technique, we have shown a selective SRAM protecting that can give a fine-granularity trade-offs between the system dependability and the memory area cost.

References

[1] Robert Baumann. Soft errors in advanced computer systems. *IEEE Design & Test of Computers*, 22(3):258–266, 2005.

[2] Jean-Marc Daveau, Alexandre Blampey, Gilles Gasiot, Joseph Bulone, and Philippe Roche. An industrial fault injection platform for soft-error dependability analysis and hardening of complex system-on-a-chip. In *IRPS*, pages 212–220. IEEE, 2009.

[3] P. Kudva, J. Kellington, P. Sanda, R. McBeth, J. Schumann, and R. Kalla. Fault injection verification of IBM POWER6 soft error resilience. In *Workshop on Architectural Support for gigascale Integration (ASGI), 2007*, 2007.

[4] Zhe Ma, Paul Marchal, Daniele Scarpazza, Peng Yang, Chun Wong, José Ignacio Gómez, Stefaan Himpe, Chantal Ykman-Couvreur, and Francky Catthoor. *Systematic methodology for real-time cost-effective mapping of dynamic concurrent task-based systems on heterogeneous platforms*. Springer, 2007.

[5] Subhasish Mitra, Norbert Seifert, Ming Zhang, Quan Shi, and Kee Sup Kim. Robust system design with built-in soft-error resilience. *IEEE Computer*, 38(2):43–52, 2005.

[6] Shubhendu S. Mukherjee, Christopher T. Weaver, Joel S. Emer, Steven K. Reinhardt, and Todd M. Austin. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. In *MICRO*, pages 29–42. ACM/IEEE, 2003.

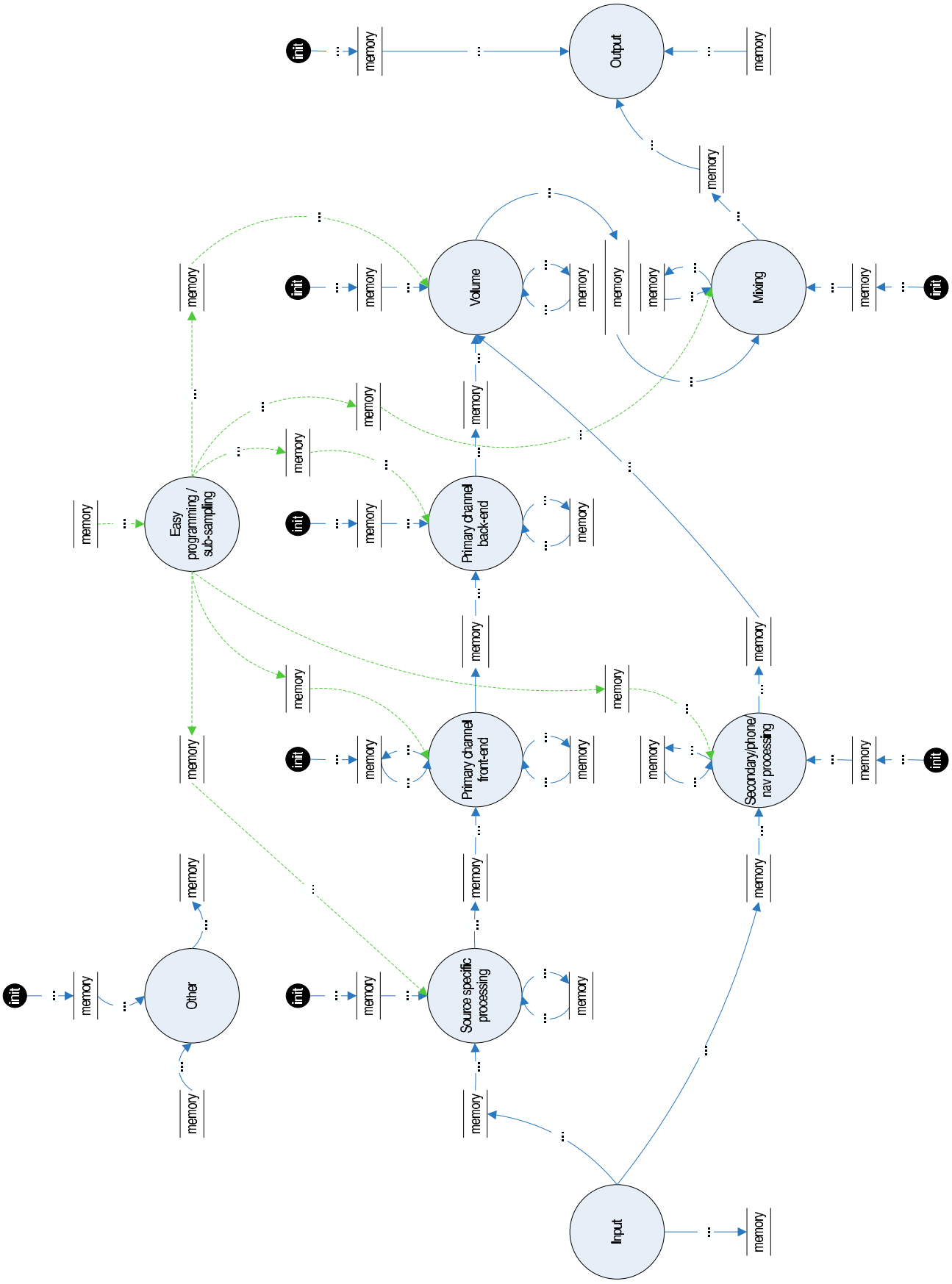

Figure 1. Audio process system TNG

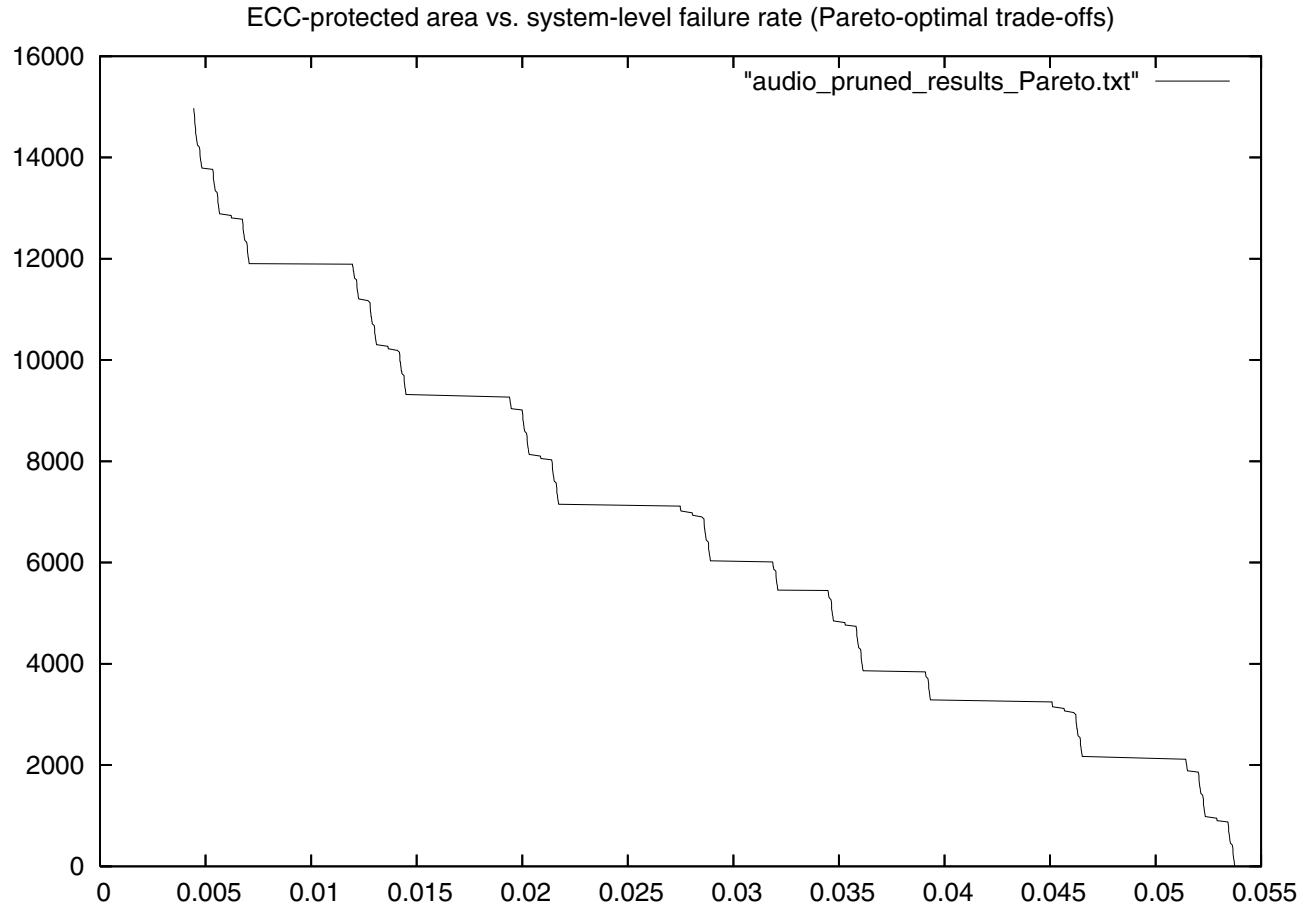

Figure 3. Selectively memory protection trade-offs

[7] Hang T. Nguyen and Yoad Yagil. A systematic approach to SER estimation and solutions. In *IRPS*, pages 60–70. IEEE, 2003.

[8] Pradeep Ramachandran, Prabhakar Kudva, Jeffrey W. Kellington, John Schumann, and Pia Sanda. Statistical fault injection. In *DSN*, pages 122–127. IEEE Computer Society, 2008.

[9] Sonny Rao, Pia Sanda, Jerry Ackaret, Adrian Barrera, Jorge Yanez, and Subhasish Mitra. Examing workload dependence of soft error rates. In *SELSE*, 2008.

[10] Daniel Skarin, Martin Sanfridson, and Johan Karlsson. Impact of soft errors in a brake-by-wire system. In *SELSE*, 2007.

[11] Frank Vermunt. SER experiment 1 (doc. rev. 1.0). Technical Report ExperimentSERonXXXXXX9141.doc, NXP, March 2009.