Methodologies for Accelerated Analysis of the Reliability and the Energy Efficiency Levels of Modern Microprocessor Architectures

Emmanouil Kaliorakis[1]

National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
manoliskal@di.uoa.gr

Abstract. The evolution in computer architecture leads to increase in performance of modern microprocessors, which is also accompanied by decrease in products' reliability that is defined as their ability to avoid service failures that are more frequent and more severe than is acceptable. Thus, designers apply different techniques throughout microprocessors life-time in order to ensure the high reliability requirements of the delivered products.

This thesis proposes novel methods to guarantee the high reliability and energy efficiency requirements of modern microprocessors that can be applied during three phases of the processors' life-cycle: (a) MeRLiN methodology to accelerate (from 1 up to 3 orders of magnitude) the reliability assessments of hardware structures at the microarchitectural level against transient faults, (b) a methodology to accelerate the online permanent fault detection of many-core architectures providing up to 47.6X speedup using the Intel SCC chip as experimental vehicle, and (c) a comprehensive statistical analysis method based on linear regression with different feature selection methods to predict the safe voltage operation margins of the ARM-v8 cores of the enterprise X-Gene 2 micro-server.

Keywords: Reliability, transient faults, permanent faults, energy efficiency, statistical analysis

1 Introduction

The evolution in semiconductor technology and computer architecture give designers the opportunity to boost the performance of modern computing systems that are used in several domains of information and communication technology systems. Despite the changes in Moore's Law [1], computer architects and designers are still able to improve processor performance by using more aggressive and sophisticated techniques. However, the scaling in performance is also accompanied by increase in the vulnerability (or decrease in reliability) of microprocessors due to: (a) the strict deadlines that are required to minimize Time-to-Market (TTM) (minimizing also the time needed to test the circuits), (b) the modern device integration techniques that make processors more vulnerable to the radiation and also increase the occurrence of manufacturing defects,

[1] *Dissertation Advisor: Dimitris Gizopoulos, Professor.*

and (c) the increased design complexity that makes the testing process of the microprocessor products very difficult and unaffordable for the available TTM.

Specifically, the modern microprocessors face serious reliability issues during their entire life-cycle due to: (i) the errors that come from *transient faults* caused by cosmic rays, alpha particles and electromagnetic interference and are manifested as instantaneous flips of the values of real hardware bits, (ii) aging that leads to errors that appear at regular time intervals (*intermittent errors*) or exist indefinitely (*permanent errors*), and (iii) manufacturing defects that can either be manifested as permanent errors or lead to timing errors when the chips operate beyond their nominal voltage and frequency conditions. These manufacturing imperfections usually force computer architects to adopt pessimistic operation margins in terms of voltage in order to protect the chips, while sacrificing the energy efficiency of the delivered product.

2 Contributions during microprocessors life-cycle

Manufacturers use several validation techniques that are implemented throughout the processor life-cycle in order to protect the chips from the different types of malfunctions, which are very important to ensure the design requirements in terms of functionality, performance, power and reliability of the delivered products. The goal of this dissertation is to provide solutions to different validation challenges during the products' lifetime. The contributions of this thesis can be grouped in the two following categories (*Pre-Silicon* and *Post-Silicon Reliability Analysis* techniques) according to the time interval they can be used during the microprocessor life-cycle:

Pre-Silicon Reliability Analysis: A very important task during the early design phases is the reliability estimation of the hardware structures and the entire chips against transient faults. The reliability and performance requirements that are defined during the planning design phase can guide several design decisions in the next phases of the processor life-cycle, such as implementation of protection mechanisms or even determination of several microarchitectural features (size of hardware structures, policies, etc.) that can influence not only the vulnerability of a chip but also its performance. Statistical fault injection of transient faults (flips of real hardware bit values) on microarchitectural structures modeled in performance simulators is a state-of-the-art method to accurately measure the reliability, but suffers from low simulation throughput.

This thesis presents several contributions in the research field of *pre-silicon reliability analysis* phase of processors life-cycle. Firstly, in [2] we present a novel fully-automated versatile architecture-level fault injection framework (called MaFIN) that is built on top of a state-of-the-art x86-64 microprocessor simulator (called Marssx86), for thorough and fast characterization of a wide range of hardware components with respect to various fault models (transient, intermittent, permanent faults). Next, by using the same tool and focusing mainly on the transient faults we executed two reliability evaluation studies. In the first study, we evaluated the reliability and performance tradeoffs for major hardware components of an x86-64 microprocessor across several important parameters of their design (size, associativity, write policy, etc.) [3]. In the second study [4], we used MaFIN in conjunction with a different tool (called GeFIN) that is also used for early reliability assessments at the microarchitecture level to evaluate in a differential way: (a) the reliability sensitivity of several microarchitecture structures for the same ISA (x86-64) implemented on two different simulators, and (b) the reliability of workloads and microarchitectures for two popular ISAs (ARM vs. x86-64). The conclusions of studies [3] and [4] can guide design decisions during the early design phase of the microprocessors concerning reliability and performance, while avoiding any costly redesign phase.

A major challenge of the early reliability assessments to soft errors at the microarchitecture level using statistical fault injection is that the campaigns that provide estimations of high statistical significance require excessively long experimental time. This thesis addresses this challenge by proposing two methodologies. Firstly, we propose to accelerate the individual fault injection runs by using several techniques that are implemented in the simulator and take place after the fault is actually injected in the hardware structure [5]. Secondly, to further accelerate the microarchitecture level fault injection campaigns we propose MeRLiN [6] that provides a final speedup of several orders of magnitude, while keeping the accuracy of the assessments unaffected even for large injection campaigns with very high statistical significance. The core of this methodology is the pruning of the initial fault list by grouping the faults in equivalent classes according to the instruction that finally accesses the faulty entry. Faults that belong to the same group are very likely to lead to the same fault effect; thus, fault injection is performed only in a few representatives from each group. MeRLiN methodology constitutes a major breakthrough in the field of accelerating the reliability estimations of hardware components at the microarchitecture level with negligible loss of accuracy.

Post-Silicon Reliability Analysis: Another important phase during the processor reliability life-cycle is the *Post-Silicon Reliability Analysis* that consists of the manufacturing testing and the in-field verification that take place during the fabrication process and after the release of the microprocessors to the market, respectively. Note that in contrast to *Pre-Silicon Reliability Analysis*, in this phase the validation targets implemented circuits and especially after their release to the market when the designers have no longer interaction with the design. The contributions of this thesis in this phase of the life-cycle cover two important research fields:

a) **Acceleration of permanent faults online detection in many-core architectures**: The extreme complexity of many-core processor architectures and the pressure for reduced time-to-market renders even the most comprehensive verification and testing process before and during mass production incomplete. A significant population of manufacturing faults escape in the field of operation and jeopardize correctness of the chip. Online functional testing is an attractive low-cost error detection solution, but it should be fast enough in order to not impact the system performance. This thesis faces this challenge by proposing an effective parallelization methodology [7] to accelerate online error detection for many-core architectures by exploiting the high-speed message passing on-chip network to accelerate the parallel execution of the test preparation phase of memory-intensive test programs.

To demonstrate the efficiency of the proposed methodology we used a 48-core real hardware chip, Intel's Single-chip Cloud Computer (SCC).

b) **Statistical analysis to predict the safe voltage margins in multicore CPUs for energy efficiency**: Reduction of the voltage operation margins of multicore chips is a major challenge for the designers to gain in terms of power. Unfortunately, this reduction leads to several reliability issues due to the manufacturing defects that make hardware cores of the same chip to present variations in their safe voltage and frequency operation limits. These variations that remain constant after the release of the chip to the market are classified as static variations. On top of that, transistor aging and dynamic variations in supply voltage and temperature, caused by different workload interactions can also affect the correct operation of a microprocessor. Thus, the designers choose to insert conservative guard-bands in the operating voltage (and frequency) to protect the chips from the effects of the static and the dynamic variations, despite the induced cost in terms of energy (and performance). The contribution of this thesis to this challenge is to propose a detailed statistical analysis methodology [8] [9] to accurately predict at the system level the safe voltage operation margins of the eight ARMv8 cores of the X-Gene 2 chip fabricated on 28nm technology. Our analysis uses as inputs the microprocessor's performance counters values of benchmarks that were collected in nominal voltage conditions execution and the results of the characterization phase when the chip operates in scaled voltage conditions.

In the next section, we present in more details the methodologies and the evaluation results of the major contributions of this dissertation.

3 Acceleration of reliability assessments against transient faults

The methodologies that were presented in this dissertation to accelerate the statistical fault injection campaigns that are used to assess the reliability of the hardware structures at the microarchitectural level can be summarized in two categories. In the first category, we accelerate the individual injection runs after the actual injection of the fault in the structure based on the faults lifetime (Section 3.1), while in the second category we further accelerate the fault injection campaigns of high statistical significance using MeRLiN methodology by pruning the faults of the initial fault list (Section 3.2).

3.1 Acceleration of injection campaigns based on the faults lifetime

In [5], we extended the baseline mode of an out-of-order cycle accurate full-system x86-64 fault injection framework (MaFIN) [2] with two extra modes of operation in order to speed up the statistical fault injection campaigns at the microarchitecture level. The common characteristic of the two proposed techniques of [5] is that they are implemented after the actual injection of the fault in the hardware structures during its lifetime. In the first mode, an injection experiment is forced to completion when the fault is overwritten before it is read and thus we classify it early and accurately as Masked. In the second mode, an injection experiment is forced to completion before the end of the application in two cases: (a) when the fault is overwritten before it is read, or (b) when an x86 instruction reads the fault from the faulty entry and reaches the commit stage and before the actual termination of the benchmark. The second method provides a tradeoff between speedup and accuracy in order to deliver a fast but less accurate solution in the early reliability estimation problem.

For evaluation, we used MaFIN to carry out extensive fault injection campaigns of transient faults in six structures of the microprocessor that hold the majority of chip's area: L1 Data cache, L1 Instruction cache, L2 unified cache, Physical Integer Register File, LSQ (data field) and LSQ (address field). We used seven benchmarks from the MiBench suite [10], while we injected 2000 faults per campaign that corresponds to 2.88% error margin and 99% confidence level according to [11].

From the results of this study, we concluded that for the intra core structures (physical integer register file, address and data fields of LSQ), the best solution to speed up the statistical fault injection campaign is the second mode of operation with negligible loss of estimation accuracy, leading to a high speedup of 3.38X, 4.06X and 3.37X for the three structures respectively. Except for the second mode, the first mode could be also used for the same structures without any accuracy loss leading to a final speedup of 2.63X, 2.92X and 1.46X respectively.

On the other hand, the best choice for an architect to estimate the reliability of caches is the first mode operation. This conclusion comes from the fact that the inaccuracy of caches' reliability assessment using the second mode is not negligible (from 8.47 percentile units for L2 cache to 20.13 units for L1 Data cache) and the speedup is not as high as in the intra core structures (for instance only 1.06% increase of speedup for the L2 cache). Consequently, the first mode is the best choice for caches to speedup campaign (with 1.37X, 1.48X and 1.05X speedup for the L1 Data, L1 Instruction cache and L2 cache respectively) and to ensure the final estimation accuracy.

3.2 Acceleration of injection campaigns based on fault pruning (MeRLiN)

Exhaustive fault injection at the microarchitecture level using the entire statistically significant fault list (i.e. a list of all the flips for every bit of all hardware structures and for every program execution cycle) is infeasible. Thus, designers in the industry resort to statistical fault sampling to boost the throughput of massive fault injection campaigns, while maintaining a reasonably high accuracy of the final estimations. Despite the acceleration provided by the statistical fault sampling, the total simulation time that is needed to estimate the vulnerability of multiple hardware structures with different configuration parameters is still unaffordable and it leads to larger conservative design decisions. Consequently, the acceleration of the microarchitectural fault injection of array-based structures that occupy the majority of chip's area during the first steps of its design cycle is of major importance for the engineers that work in the industry.

MeRLiN methodology [6] accelerates the reliability estimation of hardware structures at the microarchitecture level from 1 up to 3 orders of magnitude without compromising the final accuracy of the assessment. MeRLiN consists of three main phases (see Fig. 1): *Preprocessing*, *Fault List Reduction* and *Fault Injection Campaign*. Next, we briefly describe these three phases:

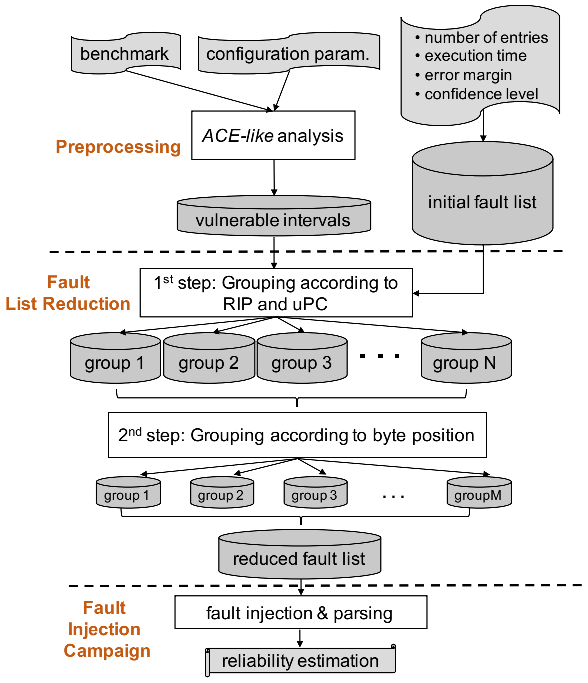
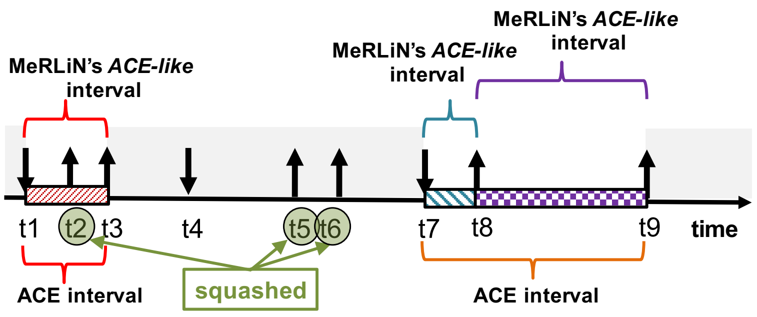


Fig. 1. Flowchart of MeRLiN.

Preprocessing: This phase takes place off-line and is responsible for two main tasks: the *ACE-like analysis* and the *Initial Fault List Creation*:

a) The *ACE-like analysis* is responsible to identify in a single pass all the *vulnerable intervals* of all hardware entries of the targeted structure (physical registers, store queue entries, cache words, etc.). For our analysis, a vulnerable interval of an entry starts with a write operation and ends with a committed read of the same entry or starts with a committed read and ends with another committed read of the same entry; all other intervals are non-vulnerable. This is different than classical ACE analysis [6] (see Fig. 2). All the faults that hit non-vulnerable intervals are excluded from the procedure of the actual injection and are classified as *Masked* as it is definite that they will not affect program execution. During the ACE-like analysis task, all the information concerning the vulnerable intervals is stored along with the information of the instruction pointer (*RIP*) and the microprogram counter (*uPC)* of the microarchitectural instruction that accesses the entry at the end of the vulnerable interval. This information is needed in the second phase of our algorithm (*Fault List Reduction*).

b) The *Initial Fault List Creation* is the second task of the first phase of MeRLiN in which the initial fault list of each injection campaign is created according to the typical statistical fault sampling formula of [11]. The initial fault list guarantees high statistical significance for all our results. The initial faults population is defined by: (1) the size (in bits) of the hardware structure, (2) the total execution (in cycles) of the benchmark that is already known from the *ACE-like* task of the method, (3) the statistical confidence level and (4) the statistical error margin. The execution time required for the exhaustive fault injection campaigns using this initial population of faults at the microarchitecture level for all the combinations of benchmarks, hardware components and configuration parameters of each component is months or even years of execution, which is infeasible to take place especially in the early design phases of a chip. With MeRLiN we reduce this time by 1 to 3 orders of magnitude.



MeRLiN's *ACE-like* interval
MeRLiN's *ACE-like* interval
MeRLiN's *ACE-like* interval
t1
t2
t3
t4
t5
t6
t7
t8
t9
time
ACE interval
squashed
ACE interval

Fig. 2. ACE and *ACE-like* intervals definition example.

Fault List Reduction: This phase of MeRLiN consists of a two-step grouping algorithm:

a) In the *1st step of the group creation algorithm*, the remaining faults that hit ACE-like vulnerable intervals are stored in different subdirectories according to the *RIP* and the *uPC* of the instruction that reads the entry at the end of the interval (see Fig. 3). Each of the created groups consists of transient faults on the same or different entries of the hardware structure being analyzed, during the same or different ACE-like vulnerable intervals that are read by a micro-instruction with the same *RIP* and the same *uPC*. The classification of the faults in groups according to the *uPC* and the *RIP* of the microarchitectural instruction that accesses the faulty entry at the end of the vulnerable interval is necessary because different micro-instructions of the same instruction (x86 in our case study but generally applicable) can lead to different fault effects; thus, they should be classified separately in different groups (our experimental results validate this assumption).

b) In the *2nd step of the group creation algorithm*, to maximize MeRLiN's accuracy especially for groups with hundreds of faults, we select more than one fault for the actual fault injection runs in cases that faults hit a different byte of the entry. Moreover, faults in different bytes are selected from different dynamic instances of the same static instruction to increase time diversity (see Fig. 4). In this way, MeRLiN finally creates groups of equivalent faults at the byte granularity ensuring the accuracy of the final estimation. This can be further extended to separate faults hitting different nibbles or bits, but our experiments verify that this is not necessary and the byte granularity is sufficient.

Fault Injection Campaign: At the end, all the selected faults from all the created groups are stored in the *reduced fault list repository*. Only these group representative faults are actually injected using the microarchitecture level fault injector.

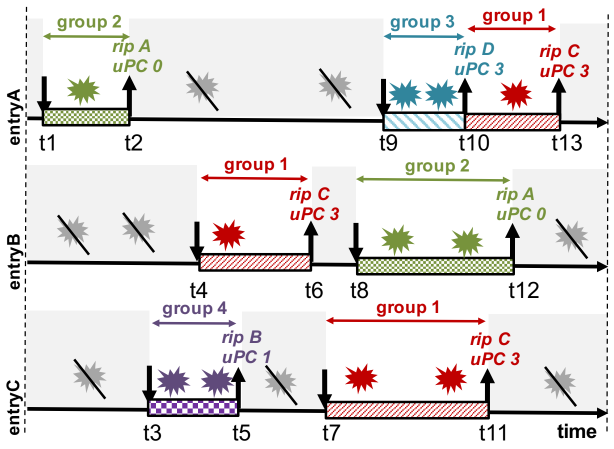


Fig. 3. 1st step example of the grouping algorithm.

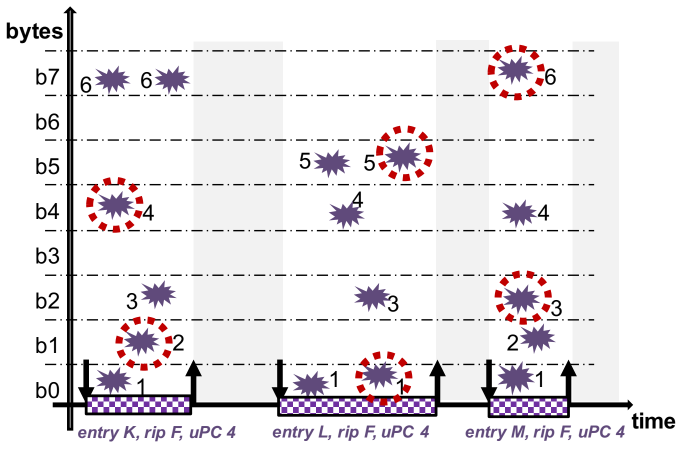


Fig. 4. 2nd step example of the grouping algorithm.

To evaluate the accuracy of the *Fault List Reduction* phase of MeRLiN we defined the *homogeneity metric* that expresses the effectiveness of our group creation algorithm to classify faults in groups that finally manifest the same fault effect. On average, the *homogeneity* of the created groups is very high (more than 91% for all our campaigns). In our study, we used the state-of-the-art microarchitectural injection tool (GeFIN) that is based on Gem5 simulator and extended it to implement and evaluate MeRLiN methodology on three data-related structures and one instruction-related structure of an x86-64 out-of-order processor model:

- The physical integer Register File for three sizes: 256, 128, 64 registers.
- The data field of the Store Queue of the Load/Store Queue for three sizes: 64 load and 64 store, 32 load and 32 store, and 16 load and 16 store entries. Gem5 does not implement data fields in the Load Queue.
- The data array of L1 Data cache for three sizes: 64KB, 32KB and 16KB.
- The destination register of the Issue Queue for two sizes: 32 and 60 queue entries.

For all our fault injection campaigns, we used 60,000 faults that correspond to 99.8% confidence level and 0.63% error margin. The key contributions of MeRLiN methodology are summarized below:

- It accelerates statistical microarchitecture level fault injection from 1 to 3 orders of magnitude. Our experiments with full runs of 10 MiBench benchmarks [10] show 93X, 225X, 68X and 28X speedup on average for different sizes of the register file, the store queue, the first level data cache and the issue queue, respectively. When applied to 10 SPEC CPU2006 benchmarks [12], MeRLiN reveals larger average speedups of 1644X, 2018X and 171X for the register file, the store queue and the first level data cache, respectively.
- It reports virtually the same reliability estimations as exhaustive (and infeasible) microarchitectural fault injection with extremely high statistical significance.
- It delivers fine-grained insights of the fault effects (Silent Data Corruptions – SDC, Detected Unrecoverable Errors – DUE, crashes, locks) unlike lifetime-analysis methods. This can be used to evaluate different protection schemes or to identify benchmarks more prone to SDCs or DUEs.

4 Acceleration of permanent faults online detection in many-core architectures

Functional testing techniques have gained increasing acceptance for microprocessor error detection during the last years. Functional online error detection approaches for many-core architectures are based on the application of the test programs during normal system operation and should adhere to the following requirements: (a) *reduced test program execution time*, (b) *small memory footprint*, (c) *test program replication* as all processor cores have to execute the same test program to detect faults and guarantee high fault coverage levels (this is different from traditional parallel programs).

In [7], we propose a functional online approach to accelerate the error detection of the Intel's Single-chip Cloud Computer (SCC) that contains 48 in-order Pentium cores. The SCC architecture consists of 24 tiles (two cores per tile) and 4 integrated DDR3 memory controllers supporting up to 64GB DRAM. Each processor SCC core has a 16KB L1 instruction cache, a 16KB L1 data cache and a 256KB L2 cache. Moreover, each tile has a 16KB message passing buffer (MPB) that bypasses L2 cache during communication.

For the experiments, we developed two test programs with different characteristics which represent typical test program formats used in functional online testing:

- *Load-Apply-Accumulate (LAA)* test program. It applies ATPG-generated test patterns stored in the off-chip DRAM. An LAA test program first reads two test vectors from the DRAM (assuming two-operand operations are being tested); it applies the target instruction (i.e. an arithmetic or logic instruction) and finally accumulates the results. We experiment with a loop-based LAA test program which applies a certain amount of test patterns (192KB or 384KB). LAA test program is memory-intensive and stresses the memory system of the SCC.
- *Linear-Feedback-Shift-Register* (LFSR) test program. This CPU-intensive test program applies pseudorandom patterns generated by a 32-bit LFSR. Similarly to LAA program, it first generates two pseudorandom test patterns, applies the target instruction and accumulates the results. LFSR test program generates either the same number of test data with LAA or 10 times more (e.g. for 384KB LAA, LFSR generates 3840KB test data).

The proposed method shown in [7] focuses on the efficient parallel execution of the test preparation phase. The test patterns are divided into 48 segments each one assigned to the private memory region of a core. The LAA test program is divided into two phases. First, all cores load in parallel the test patterns from their private memories, apply the tests and accumulate the responses. Subsequently, each core copies the corresponding test patterns from the local MPBs of the other 47 cores and applies/accumulates the tests. It is essential that in each cycle of the second phase each MPB serves the memory requests of only one core in order to limit the traffic congestion in the mesh and the routers. The rationale of the proposed method is that having every core to read test patterns from its private memory and distribute them to the other cores is the most efficient way to parallelize the test preparation phase of the LAA program. Our experimental results revealed up to 5.9X and 36.0X speedup when we applied our proposed method to 12 and 48 cores, respectively.

Regarding the LFSR test program, our experiments revealed that its test preparation phase cannot be parallelized in a more efficient way since the time each core requires to run the LFSR code to generate a certain number of test patterns is shorter than the time to copy these test patterns from the local MPB of an adjacent core. Thus, a second improvement in the parallelization of the entire online test program could be the parallel execution of memory-intensive test programs (e.g. LAA test) and CPU-intensive test programs (i.e. LFSR test) in the two cores of the same tile. This improvement increased the final speedup to 10.8X and 47.6X for the 12 and the 48 cores, respectively.

5 Statistical analysis to predict the safe voltage margins in multicore CPUs

Both static and dynamic variations lead microprocessor architects and designers to apply conservative guardbands (operating voltage and frequency settings) to avoid timing failures and guarantee correct operation, even in the worst-case conditions excited by unknown workloads. Predicting safe voltage operation regions of the microprocessor during manufacturing or after microprocessors' release to the market using as input the performance counters provided by the system has recently gained the interest of the computer architecture community.

In [8] and [9], we implemented linear regression models with three different feature selection algorithms aiming to predict both the V_{min} and the Severity (a metric that indicates a region of chip's operation below the safe V_{min} and before the occurrence of any catastrophic for the system error) in the eight ARMv8-based cores of the X-Gene 2 chip. The inputs for our models came from the characterization phase of the chip in scaled voltage conditions and from the performance counters that were collected for each workload during its entire execution in nominal voltage conditions.

In our experiments, we ran all the benchmarks from the SPEC CPU2006 suite with all their inputs (40 programs in total). The evaluation of our models' accuracy targeting either the V_{min} or the Severity, was done by: (a) using the coefficient of determination (R^2) that assesses how well a model explains and predicts the future outcomes; also, it is indicative of the level of explained variability in the dataset. The larger the values of R^2, the better fit the model provides, while the best fit exists when R^2 is equal to 1, (b) using the Root Mean Square Error (*RMSE*) that represents the deviation between the predicted and the observed values (the smaller the *RMSE* the more accurate the model is), (c) comparing our models with the baseline (naïve) model, which is the average of the target values (V_{min} or Severity) of the training dataset.

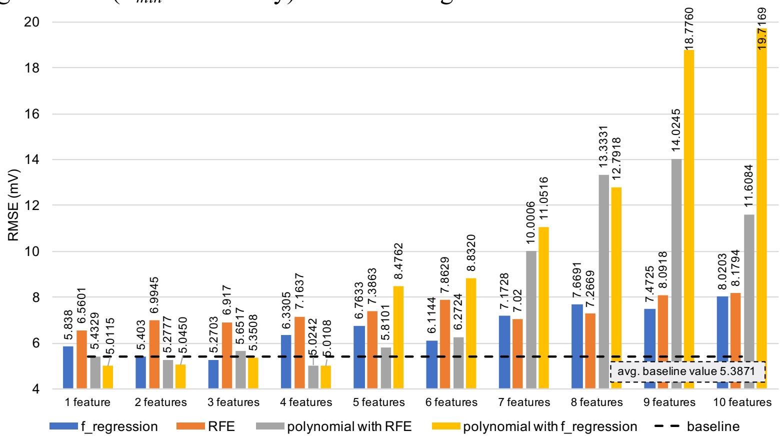


Fig. 5. Accuracy of predicting the V_{min} of the most sensitive core.

In general, our proposed method can lead to power gains from 11.87% up to 20.28% depending on the aggressiveness (Severity prediction is more aggressive than predicting V_{min}) of the prediction scheme. In this section and due to space limitations, we present in Fig. 5 the results for only one representative case of our analysis on predicting the V_{min} of the most sensitive core of the chip (core with the lowest V_{min} on average for all the experiments). The best accuracy (only 5mV inaccuracy) for this case was observed after using the polynomial transformation with *f_regression* selection and only 4 selected polynomial features leading to 11.87% power savings compared to the case of using the very pessimistic nominal voltage limit. Moreover, the R^2 that was measured for this prediction model is high (close to 0.75) indicating a good fit of the model.

6 Conclusions

In this dissertation, we propose several techniques to accelerate the analysis of the reliability and the energy efficiency levels of modern microprocessors that can be employed throughout the different phases of their life-cycle. For the *pre-silicon reliability analysis* phase, we proposed different methods to accelerate the statistical fault injection campaigns at the microarchitectural level either based on the faults lifetime after their injection that leads to an acceleration of up to 4.06X or by using fault pruning of the initial fault list provided by MeRLiN methodology that accelerates the reliability assessments even further (from 1 up to 3 orders of magnitude). Moreover, for the *post-silicon reliability analysis* phase, we proposed two methodologies. In the first, we proposed a parallelization approach to accelerate the online detection of permanent faults in many-core architectures (with up to 47.6X speedup), while in the second we proposed a statistical analysis approach to accurately predict (with only 5mV inaccuracy) the safe voltage operation margins of the ARMv8 cores of the X-Gene 2 chip.

References

1. G.Moore, "Cramming more components into integrated circuits", In Electronics, April 1965.
2. N.Foutris, M.Kaliorakis, S.Tselonis, D.Gizopoulos, "Versatile architecture-level fault injection framework for reliability evaluation: a first report", IEEE International On-Line Testing Symposium, 2014.
3. S.Tselonis, M.Kaliorakis, N.Foutris, G.Papadimitriou, D.Gizopoulos, "Microprocessor reliability- performance tradeoffs assessment at the microarchitecture level", IEEE VLSI Test Symposium, 2016.
4. M.Kaliorakis, S.Tselonis, A.Chatzidimitriou, N.Foutris, D.Gizopoulos, "Differential fault injection on microarchitectural simulators", IEEE International Symposium on Workload Characterization, 2015.
5. M.Kaliorakis, S.Tselonis, A.Chatzidimitriou, D.Gizopoulos, "Accelerated microarchitectural fault injection-based reliability assessment", IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2015.
6. M.Kaliorakis, D.Gizopoulos, R.Canal, A.Gonzalez, "MeRLiN: Exploiting dynamic instruction behavior for fast and accurate microarchitecture level reliability assessment", ACM/IEEE International Symposium on Computer Architecture, 2017.
7. M.Kaliorakis, M.Psarakis, N.Foutris, D.Gizopoulos, "Accelerated online error detection in many-core microprocessor architectures", IEEE VLSI Test Symposium, 2014.
8. G.Papadimitriou, M.Kaliorakis, A.Chatzidimitriou, D.Gizopoulos, P.Lawthers, S.Das, "Harnessing voltage margins for energy efficiency in multicore CPUs", IEEE/ACM International Symposium on Microarchitecture, 2017.
9. M.Kaliorakis, A.Chatzidimitriou, G.Papadimitriou, D.Gizopoulos, "Statistical analysis of multicore CPUs operation in scaled voltage conditions", IEEE Computer Architecture Letters, Jan. 2018.
10. M.R.Guthaus et al., "MiBench: A free, commercially representative embedded benchmark suite", International Workshop on Workload Characterization, 2001.
11. R.Leveugle, A.Calvez, P.Maistri, P.Vanhauwaert, "Statistical fault injection: Quantified error and confidence", ACM/IEEE Design, Automation & Test in Europe Conference, 2009.
12. Standard Performance Evaluation Corporation, https://www.spec.org [accessed 13/11/2017]