Revisiting Vulnerability Analysis in Modern Microprocessors

Michail Maniatakos, *Member, IEEE*, Maria Michael, *Member, IEEE*, Chandra Tirumurti, *Member, IEEE*, and Yiorgos Makris, *Senior Member, IEEE*

Abstract—The notion of Architectural Vulnerability Factor (AVF) has been extensively used to evaluate various aspects of design robustness. While AVF has been a very popular way of assessing element resiliency, its calculation requires rigorous and extremely time-consuming experiments. Furthermore, recent radiation studies in 90 nm and 65 nm technology nodes demonstrate that up to 55 percent of Single Event Upsets (SEUs) result in Multiple Bit Upsets (MBUs), and thus the Single Bit Flip (SBF) model employed in computing AVF needs to be reassessed. In this paper, we present a method for calculating the vulnerability of modern microprocessors -using Statistical Fault Injection (SFI)- several orders of magnitude faster than traditional SFI techniques, while also using more realistic fault models which reflect the existence of MBUs. Our method partitions the design into various hierarchical levels and systematically performs incremental fault injections to generate vulnerability estimates. The presented method has been applied on an Intel microprocessor and an Alpha 21264 design, accelerating fault injection by 15×, on average, and reducing computational cost for investigating the effect of MBUs. Extensive experiments, focusing on the effect of MBUs in modern microprocessors, corroborate that the SBF model employed by current vulnerability estimation tools is not sufficient to accurately capture the increasing effect of MBUs in contemporary processes.

Index Terms—AVF, microprocessor vulnerability, statistical fault injection, multiple bit upset

1 Introduction

The increasing threat of soft errors in nanometer technologies has resulted in a plethora of design solutions for protecting latches from SEUs [1], as well as combinational logic from Single Event Transients (SETs) [2], [3]. Despite the demonstrated effectiveness of these solutions, applying them blindly across an entire design incurs prohibitive cost. As a result, various methods for assessing the susceptibility of individual latches or logic gates have also been proposed [4], [5], in order to support partial hardening approaches [6], [7], [8], [9], [10]. Susceptibility evaluation and the corresponding ranking of latches or logic gates typically takes into account a number of factors, including electrical device characteristics, timing issues, as well the actual logic function implemented. These factors reflect the circuit-level and gate-level reasons that may prevent an SEU or an SET from causing a soft error in a circuit. However, they are unable to capture error masking causes at higher levels and, therefore, they prove rather insufficient when applied to modern microprocessors.

Modern microprocessors exhibit a high degree of architectural-level and application-level masking, resulting in many errors being suppressed or having a low probability of affecting the workloads that are typically executed. Indeed, the multitude of functional units and stages in deeply-pipelined superscalar microprocessors, along with advanced architectural features such as dynamic scheduling and speculative execution, imply that rather complex conditions need to be satisfied in order for an error to affect the architectural state of the microprocessor or the outcome of an application. In an effort to capture these additional masking factors, vulnerability analysis methods have been developed specifically for microprocessors [11], [12], [13], [14], [15]. These methods typically employ SFI and actual workload simulation using an architectural performance model, a Register Transfer (RT-Level), or a Gate-Level microprocessor model, and seek to assess the probability that a transient error in a state element will affect workload execution, commonly known as the AVF. As we discuss in the next section, however, the use of performance models limits vulnerability analysis accuracy, while the use of RT-Level or Gate-Level models of an entire microprocessor requires prohibitive simulation time.

At the same time, the use of AVF for accurate microprocessor vulnerability analysis is also challenged by the dramatic increase of multi-bit failure rates [16], [17]. AVF captures the effect of an SEU through a SBF model. However, as process feature sizes continue to shrink, it has become more likely that adjacent cells may also be affected by a single event [18], thereby causing a MBU.

- *M. Maniatakos is with the Department of Electrical and Computer Engineering, New York University Abu Dhabi, Abu Dhabi, UAE. E-mail: michail.maniatakos@nyu.edu.*
- *M. Michael is with the Department of Electrical and Computer Engineering, University of Cyprus, Cyprus. E-mail: mmichael@ucy.ac.cy.*
- *C. Tirumurti is with the Validation and Test Solutions Group, Intel Corporation, Santa Clara, CA 95050, USA. E-mail: chandra.tirumurti@intel.com.*
- *Y. Makris is with the Department of Electrical Engineering, The University of Texas at Dallas, Richardson, TX 75080-3021, USA. E-mail: yiorgos.makris@utdallas.edu.*

Manuscript received 15 Nov. 2012; revised 14 Oct. 2013; accepted 31 Oct. 2014. Date of publication 25 Nov. 2014; date of current version 12 Aug. 2015. Recommended for acceptance by A. Zomaya.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2014.2375232

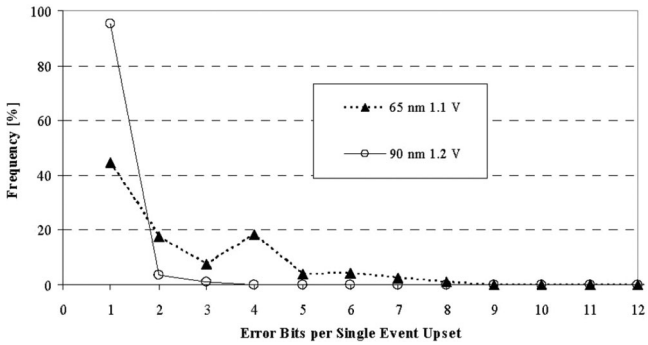

Fig. 1. Frequency distribution of number of faulty bits generated per SEU for different process sizes [16].

An MBU is defined as an event that causes more than one bit to be upset during a single measurement [19]. During an MBU, multiple bit errors in a single word, as well as single bit errors in multiple adjacent words may be introduced [20]. While MBUs have been encountered in the past in space applications [18], advanced memory structures exhibit an increasing multi-bit failure rate [16], [17], pinpointing the need for incorporating multi-bit upsets in accurate vulnerability analyses. Specifically, in [16], experiments show that only 45 percent of the upsets affect only 1 bit; the rest may affect up to 7 bits, as shown in Fig. 1. This failure rate is further accelerated by reduced power supply voltage, increased clock frequency, crosstalk and electromigration effects [21].

Considering both single-bit and multi-bit upsets becomes particularly important when assessing vulnerability of modern microprocessors. A large area percentage of such circuits is occupied by several in-core memory arrays, typically implemented using SRAM cells. To the best our knowledge, while extensive studies have been performed on increasing reliability of caches and register files [22], [23], only a few MBU studies target modern microprocessor modules [21], [24], [25].

Therefore, in order to provide accurate vulnerability estimates in modern microprocessors, the following should be addressed:

- *Fast and accurate modern microprocessor fault simulation:* An RT-Level model is required, as functional simulators[1] overestimate AVF [12]. Using an RT-Level model, however, is a major bottleneck as thousands of simulations are required in an extensive SFI campaign to assess vulnerability. This problem is further accentuated when MBUs are introduced.
- *Proper modeling of multiple-bit upsets:* Such models depend on detailed radiation-induced studies. Blindly assuming multiple bit flips in adjacent cells is misleading, as there may be no series of electrical events that can result to such outcome following an SEU.

Accordingly, in this paper we present a thorough MBU vulnerability analysis study in two complex modules of two modern microprocessors. This study is facilitated by two key contributions:

1) Enhanced fault models, as presented in Section 3, in order to provide reliable vulnerability figures in the presence of multiple bit errors. This study reveals that using the conventional SBF fault model results in inaccurate estimations of module vulnerability, as compared to using more realistic fault models which are crafted based on the findings of recent proton and neutron irradiation tests.
2) A novel methodology for calculating vulnerability in modern microprocessors using Hierarchical Fault Pruning (HFP), presented in Section 4. The proposed method is faster, yet maintains the accuracy of the traditional SFI approach, which employs the entire microprocessor monolithically. HFP leverages the high masking factors of microprocessor modules to quickly prune the fault list as it progresses towards larger partitions where simulation is slower and, thereby, accelerate vulnerability analysis.

Section 5 describes the employed simulation-based experimental infrastructure. Results, along with a comparative analysis between the various models are presented in Section 6, followed by conclusions in Section 7.

2 Related Work on Vulnerability

The vulnerability of a microprocessor, expressed as its Soft Error Rate (SER) [26], [27], [28], [29], is a function of several elements that can affect the FIT (Failures In Time) rate of the microprocessor. Common elements include elevation, technology generation, supply voltage, etc., and typical FIT rate numbers for latches used in literature vary between 0.001 - 0.01 FIT/bit [30].

When a fault appears in a circuit, it may be masked by the logic, the architecture, or the application itself. Therefore, designers focus on faults that, with high probability, result in a user-visible error. Calculating this probability, however, is not trivial and researchers have followed various approaches to obtain accurate estimates.

The notion of AVF has been extensively used in the past to characterize the criticality of flip-flops and latches to program execution correctness. AVF expresses the probability of a bit-flip resulting in a visible system error. Previously proposed methods for performing AVF analysis employ either performance models [11], [31], [32] or RT-Level models [12], [13], [14] of a microprocessor. As we explain below, the former enable fast AVF estimation but suffer in terms of accuracy, while the latter offer far more accurate results but require prohibitive simulation times.

The performance model-based method described in [11] introduces the concept of Architecturally Correct Execution (ACE) and defines ACE bits as those that can cause corruption in the final output of the program. In contrast, un-ACE bits are those which under no conditions may produce a discrepancy in the final program outcome (i.e. branch predictor bits). ACE analysis is performed and evaluated on an IA-64 performance simulator, using which the authors can generate deterministic AVF estimates for the ACE bits and rank the corresponding state elements based on their

1. Since estimation of AVF using statistical fault injection relies on comparing the system output in the presence of a fault to a golden model output, preserving correct functionality is a requirement of the simulation. Therefore functional simulators, rather than performance simulators are needed.

criticality. For this purpose, workload is simulated to completion and the impact of faults in these state elements is analyzed in a single simulation pass, which is performed rapidly. Furthermore, another major benefit of ACE analysis is that it can be performed early in the design cycle. The major drawback of ACE analysis and AVF estimation using the architectural performance model, however, is the lack of detail about the actual hardware structures of the microprocessor. Therefore, the analysis is only performed for the modeled components, which in the case of [11] includes only components that affect the performance of a microprocessor. This results in significant loss of accuracy in AVF estimation. Furthermore, extensive manual effort is required to classify a bit as ACE or un-ACE.

The methods described in [12], [13], [14] resolve the AVF estimation accuracy problem by performing SFI in the RT-Level model, which reflects the actual hardware structures of the microprocessor. Recent radiation studies with actual proton and neutron irradiation tests showed that SFI measurements at the RT-Level closely match in-field exposure [33]. This accuracy, however, comes at a cost: RT-Level simulations are far slower than performance models and fault simulation tools are not readily available at this level. Furthermore, a rigorous transient fault injection campaign requires excessive simulation times in order to provide statistically significant results. In [12], the authors provide a qualitative comparison of the AVF estimation accuracy of their extensive RT-Level simulations to the ACE analysis presented in [11]. Their findings conclude that ACE analysis overestimates soft error vulnerability by about 3.5× and that this discrepancy stems from the model's lack of hardware detail and the single-pass simulation methodology. In [32], Wang et al. [12] replied that added detail to the ACE analysis can lead to tighter AVF bounds. Still, the limited detail available at the performance model remain the main contributor to AVF overestimation.

A different approach to compute the vulnerability of state elements, called Global Vulnerability Factor (GSV) analysis, is presented in [34]. GSV aims to approximate AVF using single stuck-at fault simulations. GSV can provide, in much shorter time, the same relative ranking of memory elements in terms of their criticality, as compared to AVF. However, the acquired GSV figures are dependent on the experimental setup and are not transferrable across designs and experiments.

All the aforementioned vulnerability analysis techniques employ single error models to estimate the reliability of a microprocessor. Multiple, non-concurrent faults are discussed in [35] in order to evaluate the efficiency of design diversity. Exhaustive characterization of multi-bit errors in 90/130 nm memories is presented in [17], while in [25], the authors perform multi-bit error campaigns in an embedded cache and discuss how cache scrubbing can reduce the double-bit error rate. Another investigation of multi-bit failure rate in advanced memories appears in [16], targeting 65 nm processes. The latter study triggered the definition of a new probabilistic framework for incorporating vulnerability of memories to different fault multiplicities into AVF [24]. Finally, Touloupis et al. [21] investigate the effects of multiple non-concurrent faults on the operation of a microprocessor. Both [24] and [25] conclude that the probability of non-clustered double faults is negligible, and can be eliminated with simple scrubbing techniques [25].

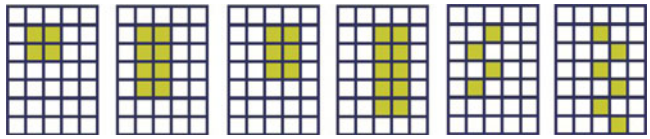

Fig. 2. Typically observed fail bit patterns [16].

In contrast, the work presented herein seeks to accelerate and enhance vulnerability analysis in modern microprocessors with contemporary MBU fault models.

3 MBU Fault Model

Recent radiation tests [16], [24] presented real-life evidence of spatial MBUs in technologies below 130nm. Fig. 1 shows the observed MBUs for two different technologies. For 65nm, almost 20 percent of the SEUs generated four soft errors, highlighting the neccessity of taking into account MBUs during SER analysis.

During a single event upset, ionizing radiation releases charge in a device. This can happen either because of direct ionization by the incident particle or because of ionization by generated particles from the reaction between the particle and the device itself. Both effects induce charge collection, possibly affecting the state of the device. Depending on the properties of the semiconductor device, the charge collection can affect multiple nodes[2].

Typically observed fail bit patterns, as presented in Fig. 2, indicate that multi-bit upsets do not manifest as multiple bit flips spread across rows or columns; instead, they are clustered in double stripes perpendicular to the wordlines and manifest as 'force-to-0' or 'force-to-1' effects. Fig. 3 shows a highly compact layout of bit cells widely used in the design of such arrays. Since the p-well is shared among every pair of columns, in case a particle strikes and causes charge collection, the generated charge raises the potential of the bulk and turns on a parasitic bipolar transistor. Hence, the circuit node is shorted to the bulk and the contents of the cell are flipped. There is also a probability that parasitic bipolar transistors in neighboring cells sharing the same p-well will turn on, effectively generating an MBU. Depending on the node hit by the particle, the value of the cell may or may not change. For example, in case node Q is struck when the bit is holding 0, the bit cell will not be affected; the same applies when node QB is struck when the bit has a value of 1. Consequently, about 50 percent of the upsets will not result into bit flips.

This model is the basis of the experiments performed in Section 6. A similar MBU fault model was also used in [24]. It should be noted that, the discussed effects apply only to dense SRAMS; therefore, combinational logic is immune to multiple-bit upsets [37].

4 Hierarchical Fault Pruning

In this section, we propose a methodology that employs statistical fault injection in a hierarchical manner in order to

2. The interested reader can find an extensive discussion on the physics of such charge collection in [36].

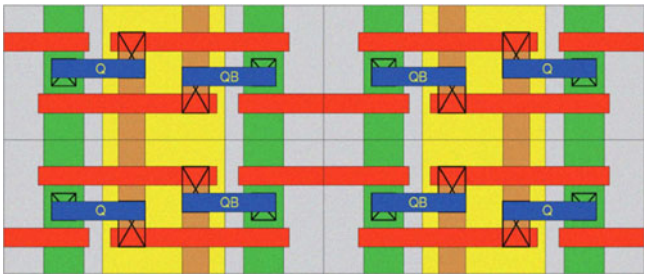

Fig. 3. Compact mirror layout of arrays [24].

accelerate vulnerability analysis. This method is demonstrated through the traditional AVF computation. In this context, by "hierarchical" we imply that fault simulation is incrementally performed in gradually expanding partitions of the design. Assume, for example, that we are interested in computing the AVF of the instruction scheduler module of the Alpha 21264 processor, shown in Fig. 4. In this case, a possible hierarchy could involve the instruction scheduler by itself in Level 1, the out-of-order execution cluster, which includes the instruction scheduler, in Level 2, and the entire microprocessor in Level 3. As can be observed, successive levels always include the preceding ones, therefore fault simulation becomes increasingly more expensive. At the same time, the size of the fault list that needs to be simulated is drastically reduced due to masking, as we proceed from each level to the next. Indeed, modern microprocessors exhibit high masking factors [38], [39], reported to be up to 98 percent (AVF = 2%) [38]. Based on these two observations, HFP aims to accelerate AVF computation by reducing the simulation effort without sacrificing accuracy.

Hierarchical approaches have previously been used in the literature in order to extract various design parameters, such as timing, energy consumption and soft error rate figures [40], [41]. These approaches, however, only perform static analysis of the circuit and do not dynamically consider any actual workload. In contrast, our HFP approach includes workload analysis, as this is essential for proper AVF estimation.

The HFP method is presented in detail in Algorithm 1. First, we partition the design into the various hierarchy levels and create the list of latches to be injected. Then, we generate a fault list with random transients for each latch in the first level. After initializing the algorithm, we fault simulate all faults in this fault list. At the end of this simulation, any fault which causes some discrepancy at the primary outputs (POs) of the first level is stored and set to be injected again at the next level. We define the fraction of the faults that appear at the POs of the partition as Module Vulnerability Factor (MVF), to avoid confusion with AVF which is defined based on faults propagating to the POs of the entire design. When all faults are simulated at the first level, all stored faults become the input fault list of the second level and the same process is applied. These faults are simulated again, since the next partition contains the previous one, but now the design is much larger and more faults will be masked and will not appear at the POs of this new level. This process repeats for each of the defined levels. At the end of the algorithm, the AVF of each latch is calculated as the product of the MVFs of this latch across all the different levels.

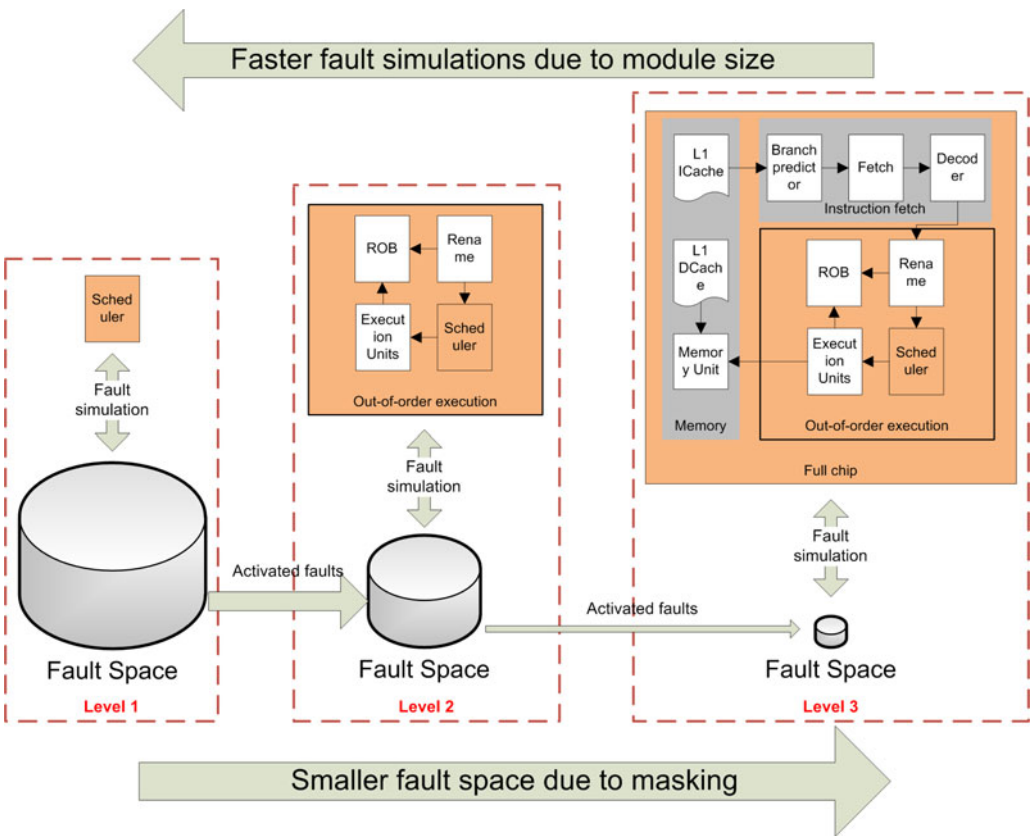

Fig. 4. Sample partitioning (Alpha 21264).

Algorithm 1. Hierarchical Fault Pruning algorithm

```
Assume T is the list of latches to be injected;
Assume n is the number of hierarchical levels;
Initialize Level = 1;
for each Latch in T do
  Generate random fault list F(Level, Latch) for Level = 1;
end
for Level = 1 to n do
  for each Latch in T do
    Initialize activated fault list A(Level, Latch) = ∅;
    for each Fault in F(Level, Latch) do
      Fault simulate Fault;
      if Fault propagates to partition's POs then
        Add Fault to A(Level, Latch);
      end
    end
      MVF(Level, Latch) = |A(Level, Latch)|/|F(Level,
      Latch)|;
      F(Level + 1, Latch) = A(Level, Latch);
  end
end
for each Latch in T do
    AVF(Latch) = MVF(1, Latch) * MVF(2, Latch) * ··· *
    MVF(n, Latch)
end
```

For example, in the three-level hierarchy shown in Fig. 4, let us assume that the MVF of a latch in the scheduler (i.e., MVF_1) is 25 percent at the first level, i.e., 1 out of 4 transients in this latch makes it to the POs of the scheduler. Let us also assume that out of these faults, 25 percent make it to the POs of the out-of-order cluster (e.g., $MVF_2 = 25\%$) and, out of those, 25 percent leave the full-chip model and are stored in memory (i.e., $MVF_3 = 25\%$). Evidently, the probability that a fault will reach the POs of the second level (out-of-order cluster) is $MVF_{1-2} = MVF_1 * MVF_2 = 25\% * 25\% = 6.25\%$, while the probability that it will reach the POs of the third level (entire design) is $AVF = MVF_{1-3} = MVF_1 * MVF_2 * MVF_3 = 25\% * 25\% * 25\% = 1.5\%$, consistent with what is reported in [38]. As a result, assuming an initial fault list size of 100 K faults in the scheduler, only 25K faults will be simulated at level 2, and only 6.25K faults will be simulated at the entire design level. In contrast, without the proposed hierarchical approach, all 100K faults would have to be simulated at the entire design level.

Defining the design hierarchy is a key part of the process where designer expertise is important. A small number of large partitions require fewer iterations of the HFP algorithm, yet each iteration takes longer to complete. A larger number of smaller partitions results in faster iterations of the HFP algorithm, yet many faults are repeatedly injected at successive levels. A balanced approach used herein, which serves as a good starting point, is to define a hierarchy consisting of (i) sub-block level, (ii) layout block level, and (iii) full-chip level.

Furthermore, given the HFP methodology, we can employ several more techniques to further accelerate fault injection, such as:

- *Correlation of vulnerability factors between different partitions.* As a fault is being injected across levels, there may be a correlation between higher levels and the final vulnerability estimate. For example, if a fault escapes the Out-Of-Order cluster, there is a very high probability that it will affect the user (thus, the out-of-order partition will provide similar vulnerability estimates to the full-chip partition).
- *Fault list size reduction.* Since HFP enables rapid fault injection, the optimal fault sampling size can be identified early in the design process, and used for accurate vulnerability estimation.

These extra speed-up factors are extensively evaluated in Section 6.

5 Experimental Setup

In order to evaluate the effectiveness of our methods, we performed extensive experiments on a contemporary Intel microprocessor implementing the P6 architecture[3], as well as on the Alpha 21264 microprocessor.

The experiments utilize SFI campaigns to estimate the soft error of the microprocessors under test. SFI has been demonstrated to closely match actual radiation studies with proton and neutron irradiation tests [33]. Furthermore, the distributions of MBU faults used in the SFI campaigns have been presented in [16] and have been extensively used in the literature.

Fig. 5 shows the basic P6 architecture [42]. Instructions flow from the instruction cache to the decoders, where they enter the out-of-order cluster and are stored in the Reservation Station (RS). The Reorder Buffer (ROB) guarantees in-order retirement after out-of-order instruction execution. The Memory Reorder Buffer (MOB) interacts with the data cache in order to fetch the required information.

The Alpha 21264 microprocessor full chip model appears on the right side of Fig. 4. The Alpha 21264 instruction flow is similar to the P6, featuring a 32-slot instruction array scheduler for storing the incoming instructions from the renaming logic, and the ability to dispatch up to 6 instructions to the corresponding functional units. The scheduler also incorporates a scoreboard to resolve data hazards.

Our study focuses on the control units of the out-of-order cluster, namely the *Instruction Scheduler* (known as *Reservation Station* in P6) and the *Reorder Buffer*. Following the guidelines provided in Section 4, our hierarchy includes 3 levels. Level 1 corresponds to the module (i.e., either the RS or the ROB). Level 2 corresponds to the partition (i.e., the out-of-order execution engine, shown as the dark gray area in Fig. 5 for the P6 and as presented in the middle of Fig. 4 for the Alpha 21264). Level 3 corresponds to the entire chip.

Our simulation experiment flow is shown in Fig. 6. Before starting the HFP algorithm, certain initialization steps are needed so that workload may be executed:

- *Workload execution.* A variety of workload should be selected for effective AVF analysis. A typical workload requires several million cycles to be completed, rendering full workload execution at the gate-level

3. A non-disclosure agreement with Intel Corporation prevents us from providing details about the actual microprocessor.

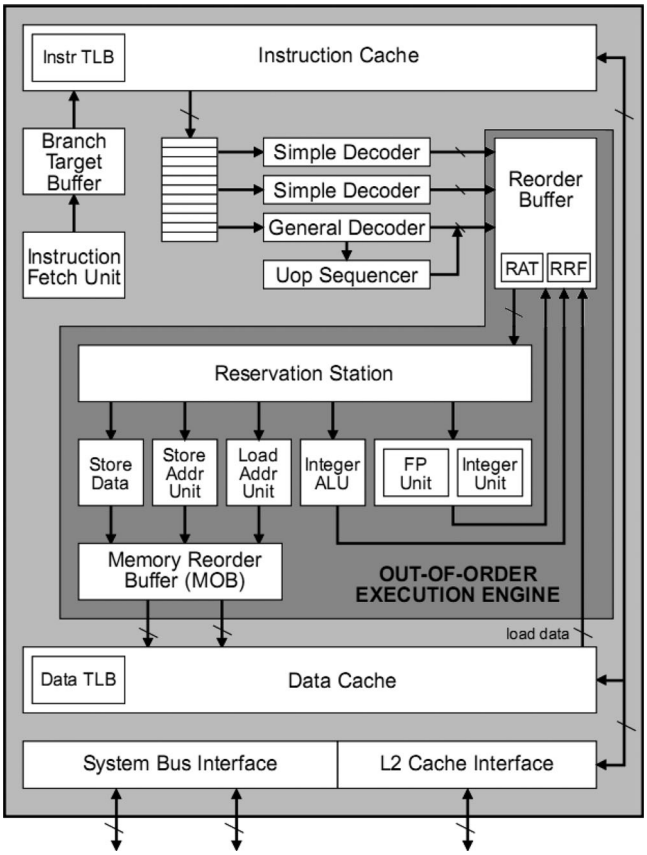

Fig. 5. P6 general architecture [42].

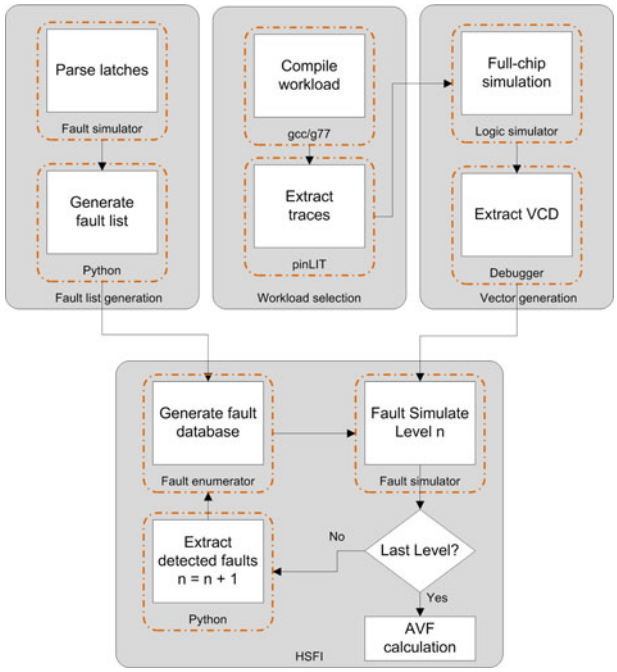

Fig. 6. Experimental flow.

impractical. In order to execute workload on P6, after compiling the workload for the specific x86 architecture, we use pinLIT [43] to extract workloads in the object file format. As for the Alpha 21264, we employ the mixed-mode simulation presented in [44], where a functional simulator is used to warm-up the caches and complete the workload, while the clock cycles of interest are examined using an RT-Level model.

- *Vector generation*: The object file has to be simulated in order to generate appropriate vectors for use during fault simulation. A logic simulator is used to simulate workload and Value Change Dumps (VCDs) are extracted at the input boundary of each level.
- *Fault list generation*: In order to apply SFI, a set of fault locations is needed, on which transient error injections are performed in order to calculate the AVF. Since our focus is the AVF of memory elements, the design is parsed and the fault locations are enumerated. Given this list, transient errors are randomly generated and injected uniformly over time, ensuring that the same number of errors is injected in each element.

The workload used in our experiments comprises various SPEC benchmarks, namely `astar` (path-finding algorithms), `bzip2` (compression), `h264` (video compression), `lucas` (primality checking), `mcf` (combinatorial optimization), `gzip` (compression), `parser` (syntactic parser) and `gap` (group theory). The benchmarks represent different workload types, which is needed for accurate AVF calculation. Workloads were executed to completion using minimal inputs. For the P6, all experiments were performed within Intel's environment, on machines with similar capabilities. For the Alpha 21264, the experiments were performed on a Xeon 3.4GHz server with 16GB of RAM. The times reported in Section 6 are averages of several simulations to ensure fair comparison in terms of required resources.

6 Experimental Results

In the first part of this section, we demonstrate the AVF calculation acceleration obtained by the proposed HFP method over the traditional SFI approach and we investigate possible options for gaining further speed-up at the expense of some minor accuracy loss. In the second part, we compare the results of different MBU models, while in the third part we evaluate vulnerability for realistic MBU distributions.

6.1 AVF Calculation Speed-Up

6.1.1 Hierarchical Fault Pruning

Table 1 compares the simulation time of SFI to that of HFP for our fault-injection campaign of 175K faults. The reported times are averaged over 20 fault simulation runs on a single machine. The simulation time needed for each level in the HFP method is also shown in the table. Specifically, HFP_1 refers to the simulation time needed to obtain results at the boundary of the first level (i.e., the RS or the ROB module), HFP_{1-2} refers to the cumulative simulation time to obtain results at the boundary of the second level (i.e., the out-of-order execution engine), and HFP_{1-3} refers to the cumulative simulation time to obtain results at the boundary of the full chip. Evidently, $HFP_{1-3} > HFP_{1-2} > HFP_1$.

In order to compare the simulation times between HFP and traditional SFI, we need to compare the simulation time of HFP_{1-3} and the simulation time of SFI. The corresponding entries in Table 1 show that HFP outperforms SFI by more than an order of magnitude. For example, calculating AVF for the latches of the P6 RS while running the `astar` workload on 175K faults via SFI requires 3 days (263,802

TABLE 1
Simulation Time (in Minutes) Comparison between HFP and SFI for 175K Faults for P6

		astar	bzip2	h264	lucas	mcf
RS	HFP_1	68	67	68	55	44
	HFP_{1-2}	166	137	130	89	153
	HFP_{1-3}	284	302	312	528	397
	SFI	4,396	3,023	6,839	6,209	2,029
	Speed-up	**15×**	**10×**	**21×**	**12×**	**5×**
ROB	HFP_1	110	117	111	122	103
	HFP_{1-2}	131	142	133	145	125
	HFP_{1-3}	281	421	426	440	492
	SFI	5,247	3,779	9,438	7,645	5,876
	Speed-up	**18×**	**9×**	**22×**	**17×**	**12×**

TABLE 2
Simulation Time (in Minutes) Comparison between HFP and SFI for 175K Faults for Alpha 21264

		gzip	bzip2	parser	gap	mcf
Scheduler	HFP_1	321	392	295	475	289
	HFP_{1-2}	798	972	701	1,103	735
	HFP_{1-3}	1,252	1,674	1,358	2,503	1,210
	SFI	23,802	18,417	20,383	37,545	21,792
	Speed-up	**19×**	**11×**	**15×**	**15×**	**18×**
ROB	HFP_1	280	484	791	688	594
	HFP_{1-2}	749	1,345	1,973	1,831	1,002
	HFP_{1-3}	1,274	2,972	3,755	3,670	2,225
	SFI	31,856	26,754	56,338	48,721	35,610
	Speed-up	**21×**	**9×**	**15×**	**13×**	**16×**

seconds), while the same results are obtained via HFP within only 5 hours (17,075 seconds), corresponding to a 15× speed-up. On average, across our simulations, HFP accelerated AVF calculation by 12.6x for RS and 15.6x for ROB. The key implication of the obtained speed-up is that we can now use HFP to generate statistically significant AVF numbers (i.e., 5 benchmarks, 175K faults per benchmark) for a module within a day (on a single machine), as opposed to the two weeks required by SFI.

Similar speed-up is obtained while injecting faults at the Alpha 21264 modules (Table 2). We note that the overall time needed to extract AVF numbers is much higher than for the P6, as one fault is injected per pass; in contrast, the in-house Intel simulator can inject several faults per pass.

6.1.2 AVF to MVF Correlation

Further AVF calculation speed-up can be obtained, at the cost of some minor accuracy loss, by leveraging an interesting observation regarding the correlation between AVF and MVF. Evidently, since a fault may be masked at a subsequent level in an n-level hierarchy, $MVF_1 \geq MVF_{1-2} \geq \cdots \geq MVF_{1-n} = AVF$ for every latch. However, as we approach the full-chip level, masking becomes less likely and depends mostly on the application behavior (application masking) rather than the properties of the injected latch; therefore, it is likely that the MVF will become increasingly correlated to the AVF. For example, the correlation coefficient between $MVF_{1-3} = AVF$ and MVF_{1-2} in our experiment for the Intel microprocessor is 97 percent. This does not come as a surprise, due to the nature of the partitioning chosen. A fault escaping Level 2 guarantees incorrect execution of an instruction, since Level 2 contains all the execution functionality.

The implication of this observation is that we can speed-up AVF calculation by using the –much faster to compute– MVF of preceding levels instead. For example, suppose that AVF is used to rank the state elements of a module in order to select and protect the most vulnerable ones. The top line (MVF_{1-3}=AVF) in Fig. 7 reports the coverage (y-axis) that would be achieved by protecting the corresponding percentage of the state elements shown in the x-axis[4]. This is the point of reference, reflecting an optimal state element ranking. Suppose now that the same state elements are ranked based on MVF_{1-2} instead. In this case, the coverage achieved by protecting the corresponding percentage shown in the x-axis is given by the middle line on the plot of Fig. 7. Evidently, the accuracy lost when selecting based on MVF_{1-2} instead of AVF is fairly small, while the computational gains are very large. For example, the 15× speed-up reported in Table 1 for `astar` becomes 27× if the simulation stops at HFP_{1-2}.

We note, however, that the correlation between MVF_{1-3}=AVF and MVF_1 is only 61 percent, implying that a fault escaping the boundary of a small module, such as the RS or the ROB, is not a good indication of whether it will actually affect execution. As a result, ranking and protecting state elements based on MVF_1, as shown on the bottom line of Fig. 7, yields very sub-optimal results.

6.1.3 Fault Injection Window

AVF calculation may also be accelerated, again at the cost of minor accuracy loss, by limiting the number of simulation cycles after a fault is injected. A similar approach is taken in the RT-Level AVF calculation method described in [39], where faults are not simulated until the end of the workload due to limited resources. Instead, a fault is injected during a

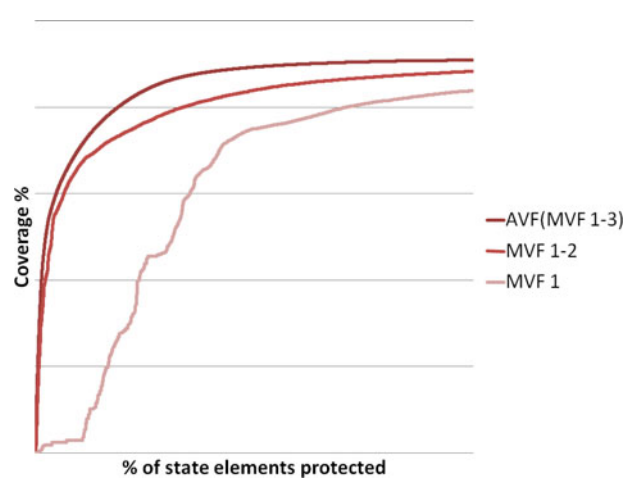

Fig. 7. Correlation between AVF and MVF calculated at various partitions levels.

4. Values on the axis are omitted and results are reported only for a subset of the latches due to the confidential nature of the actual data.

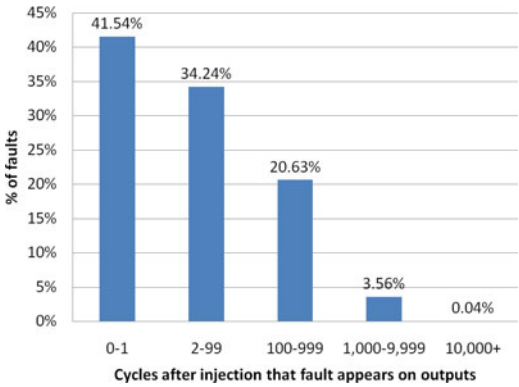

Fig. 8. Statistics driving fault injection window selection.

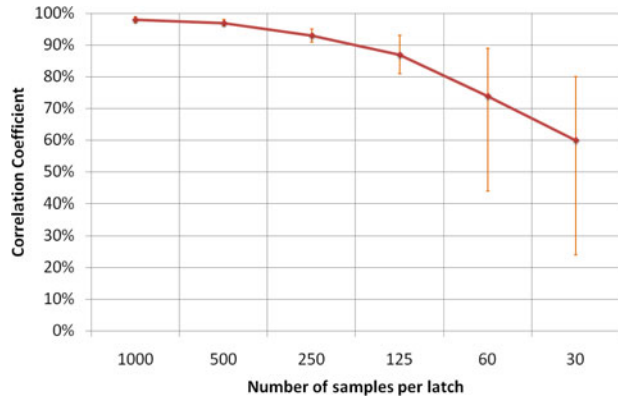

Fig. 9. Correlation between AVFs produced by different number of samples.

limited number of cycles (up to 10,000). Indeed, fault simulation of RT- or Gate-Level models is feasible only for a few thousand cycles.

In the results presented herein, complete workloads were executed exclusively at the Gate-Level, yet with minimal inputs in order to keep the clock cycles below 1 million. Fig. 8 shows how many cycles, after injection of a fault, the fault appears at the primary outputs of the corresponding partition. As can be observed, over 96 percent of the faults appear within 1,000 cycles after injection, and almost all faults (i.e, 99.96 percent) reach the outputs within 10,000 cycles. Thus, we can safely stop the simulation 10,000 cycles after fault injection and still compute very accurate AVF numbers with the data available at that point in time. In our experiments, this approach would result in an additional 10× AVF calculation speed-up, since we would fault simulate 10,000 cycles instead of 100,000+ typically used as a safety margin [39].

6.1.4 Fault List Size

Finally, additional AVF calculation speed-up may be obtained by moderating the fault list size, possibly at the expense of minor accuracy loss. In our experiments, we used a sample size of 2,500 injections per latch (5 benchmarks, 500 injections per latch) to compute the AVF numbers, which we use as our baseline. In this section, we examine the impact of reducing this sample size on the accuracy of the computed AVF.

Fig. 9 shows how the AVF computed using smaller fault list sizes (i.e., 1,000, 500, 250, 125, 60 and 30 injections per latch) correlates to the baseline AVF. For each sample size, the average over 10 different fault lists is reported (along with the deviation). For example, a sample size of 60 injections per latch correlates, on average, 78 percent with the optimal list, yet this number may be as low as 40 percent. The key take-away point from this graph is that accurate AVF estimations (i.e., with both minimum and average correlation coefficient >90%) can be obtained using a fault list size of as few as 250 samples. This leads to an additional 10x speed-up, since 250 instead of 2,500 injections are performed.

6.2 Comparison Between Different MBU Models

The usage of the Hierarchical Fault Pruning capability allows the injection of a very high number of faults in a short amount of time. In this section, we examine multi-bit upsets, considering up to four upsets for a single event (i.e. strike). Again, due to NDA, results only for the Alpha 21264 will be presented in this section.

Fig. 10 presents the hypothetical cases where MBUs have a fixed radius. As expected, lower upset radius leads to lower vulnerability estimates, and as the fault effect radius increases, the vulnerability of the module under test increases as well. An interesting observation from Fig. 10 is that, while vulnerability factors for 3BUs and 4BUs are on average higher than those for 1BUs and 2BUs, the merit figures of 3BUs and 4BUs are similar. Additional experiments showed that vulnerability does not increase significantly for an upset radius of more than 4 bits. Even though, intuitively, an MBU of 5+ bits has more chances of affecting the system output, as compared to an MBU of 4 bits, results show that the probabilities are approximately the same. This happens because of 'dead instructions', as described by Mukherjee in [11]. For example, consider a 4-bit MBU at the instruction scheduler. An MBU of 4 bits is wide enough to probably affect the system output, *unless* the scheduler location where the hit occurred did not contain valid instructions. The exact same case applies for MBUs that are 5+ wide (they have to hit dead instructions in order not to affect the execution). Thus, 5+ BUs can be approximated by 4BUs for saving simulation time.

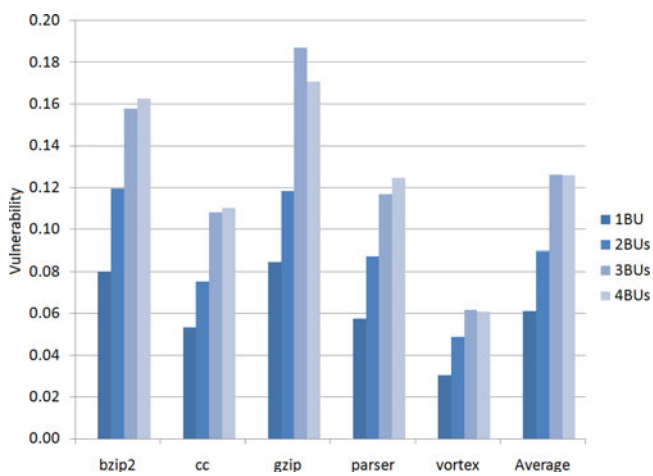

Fig. 10. Comparison between different MBU models.

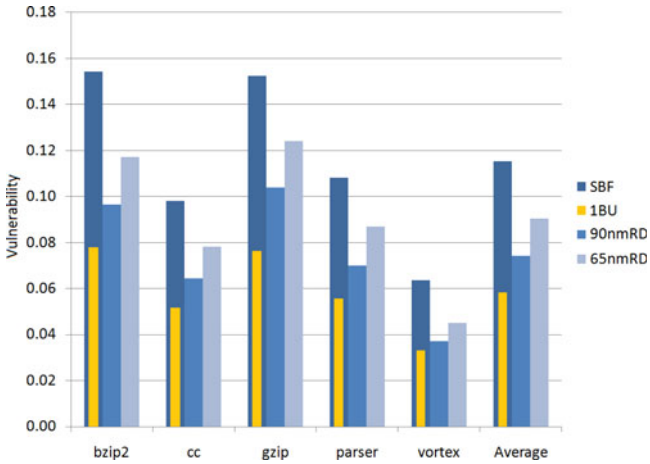

Fig. 11. Vulnerability comparison for representative MBU distributions.

6.3 Vulnerability Comparison for Representative MBU Distributions

Obviously, particle effects will not manifest exclusively as 1, 2, 3 or 4 BUs, but rather as a distribution of different MBUs. Using the distribution extracted by radiation experiments performed in 90 nm and 65 nm processes [16] (appearing in Fig. 1), we can perform a realistic comparison of the vulnerability metric using SBF to the vulnerability estimated by representative distribution of MBUs. Specifically, the 90 nm Representative Distribution (90 nmRD) consists of 95 percent 1BU, 4 percent 2BUs and 1 percent 4BUs. The 65 nm Representative Distribution (65 nmRD) reflects a completely different distribution, consisting of 45 percent 1BU, 18 percent 2BUs, 0.1 percent 3BUs and 27 percent 4BUs (as discussed in the previous section 5+ BUs are approximated by 4BUs).

Fig. 11 presents a comparison of the SBF model to the two MBU distributions of the respective process nodes. The results corroborate that due to the introduction of a significant percentage of 2+ MBUs, the SBF model leads to inflated vulnerability factors, as compared to more realistic distributions of faults. Specifically, the vulnerability estimation of 0.12 based on SBF is 0.05 more than the vulnerability estimation of 0.07 based on 90 nmRD. Thus, SBF *overestimates* vulnerability by 71 percent. Similarly, for the 65 nmRD fault model, SBF overestimates vulnerability by 33 percent (0.09 compared to 0.12).

We should note, however, a major difference between SBF and the newly defined MBU models, namely the data-dependency of the upset. Specifically, in the SBF model an event will *always* flip the target bit, independent of the existing data stored in this bit. In contrast, the xBU model will approximately flip a bit 50 percent of the time, making the actual effect data-dependent. Therefore, one might argue that in order to ensure a fair comparison, the 1BU model (which is approximately 50 percent of the SBF) should be used as the baseline. In this case, which is also presented in Fig. 11, the 1BU model *underestimates* the overall vulnerability by 14 percent, for the 90 nmRD fault model and 34 percent for the 65 nmRD fault model.

To summarize, independent of whether the SBF or the 1BU model is used as a reference point, our results support the conjecture that, in contemporary technologies wherein MBUs are prevalent, vulnerability cannot be accurately estimated without taking into account the distribution of MBU faults. Finally, besides the inaccuracy of vulnerability estimates, we would also like to point out an additional limitation of relying on SBF or 1BU models, namely their inability to correctly assess the effectiveness of error detection and correction schemes. Suppose, for example, that a designer is contemplating between a parity-based protection scheme and a Double Error Detection-Single Error Correction scheme. Using the SBF and 1BU models would fail to differentiate between the robustness achieved by each of these two options in the presence of MBUs. In contrast, using actual MBU distributions would provide estimates that would more accurately match in-field exposure.

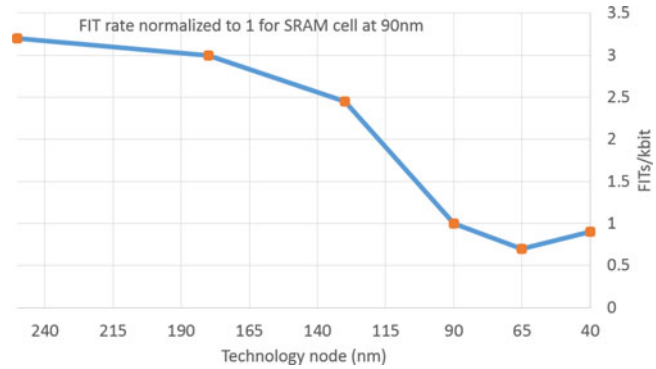

Fig. 12. SRAM SEUs as a function of the technology node [45].

6.4 Discussion on Future Technology Nodes

The fault models used in this study are based on radiation results on the very commonly used 6T-based SRAM cells. As discussed in [45], while the SEU rate in 6T SRAM cells kept decreasing as technology shrunk form 250 to 65 nm, that trend is reversed below 65 nm, as shown in Fig. 12. The reason behind the decreasing trend was the reduction of the cell area and the improvement of the technology, which counterbalanced the negative effect that critical charge reduction had on SER. Results in 40 nm, however, showed an increase in SER rate, as shown in Fig. 12. Furthermore, independent of the SER of individual cells, the *number* of cells affected by an SEU continues to increase as process shrinks [45], highlighting the need for (i) MBU inclusion in vulnerability analysis and (ii) revisiting vulnerability estimation in future technology nodes.

In order to provide vulnerability estimates in the presence of MBUs, the methodology proposed herein only requires (i) the expected distribution of faults and (ii) the expected workload. Therefore, it can be easily adapted for evaluating vulnerability of future technologies.

7 Conclusion

In this study, we enhance the accuracy of vulnerability analysis in modern microprocessors by incorporating MBUs in the vulnerability estimation process. In order to facilitate such a computationally expensive task, we introduced the concept of Hierarchical Fault Pruning, which exploits the high masking factors of modern microprocessors towards accelerating vulnerability analysis. By hierarchically partitioning the design and incrementally fault

simulating each level only for the subset of faults that evade masking in previous levels, the number of simulation cycles required is drastically reduced. On average, HFP speeds-up vulnerability calculation by 15× as compared to the traditional SFI approach, without sacrificing any accuracy, and may be used for accurate MBU vulnerability analysis. The latter is shown to be a crucial capability, as our experimental results on modern microprocessors demonstrate that the traditional SBF model does not capture the effect of MBUs, leading to inaccurate vulnerability estimates.

ACKNOWLEDGMENTS

The authors' would like to thank the anonymous reviewers for their insightful comments on improving the content and the presentation of the paper. This work was supported by a generous gift from Intel Corp. M. Maniatakos performed part of this research during a graduate student summer internship with Intel Corp. in Santa Clara, CA, USA.

REFERENCES

[1] M. Zhang, S. Mitra, T. Mak, N. Seifert, N. Wang, Q. Shi, K. Kim, N. Shanbhag, and S. Patel, "Sequential element design with built-in soft error resilience," *IEEE Trans. Very Large Scale Integration Syst.*, vol. 14, no. 12, pp. 1368–1378, Jan. 2006.

[2] Q. Zhou and K. Mohanram, "Transistor sizing for radiation hardening," in *Proc. Int. Reliability Phys. Symp.*, 2004, pp. 310–315.

[3] N. Miskov-Zivanov and D. Marculescu, "MARS-C: Modeling and reduction of soft errors in combinational circuits," in *Proc. Des. Autom. Conf.*, 2006, pp. 767–772.

[4] C. Zhao, X. Bai, and S. Dey, "A scalable soft spot analysis methodology for compound noise effects in nano-meter circuits," in *Proc. Des. Autom. Conf.*, 2004, pp. 894–899.

[5] M. Zhang and N. Shanbhag, "Soft-error-rate-analysis (SERA) methodology," *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.*, vol. 25, no. 10, pp. 2140–2155, 2006.

[6] R. Garg, N. Jayakumar, S. Khatri, and G. Choi, "A design approach for radiation-hard digital electronics," in *Des. Autom. Conf.*, 2006, pp. 773–778.

[7] Q. Zhou and K. Mohanram, "Gate sizing to radiation harden combinational logic," *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.*, vol. 25, no. 1, pp. 155–166, Jan. 2006.

[8] S. Almukhaizim, Y. Makris, Y. Yang, and A. Veneris, "Seamless Intergration of SER in rewiring-based design space exploration," in *Int. Test Conf.*, vol. 2, 2006, pp. 29.3.1–29.3.9.

[9] C. Zoellin, H. Wunderlich, I. Polian, and B. Becker, "Selective hardening in early design steps," in *Proc. Eur. Test Symp.*, 2008, pp. 185–190.

[10] S. Krishnaswamy, S. Plaza, I. Markov, and J. Hayes, "Signature-based SER analysis and design of logic circuits," *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.*, vol. 28, no. 1, pp. 74–86, Jan. 2009.

[11] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, 2003, pp. 29–40.

[12] N. J. Wang, A. Mahesri, and S. J. Patel, "Examining ACE analysis reliability estimates using fault-injection," *SIGARCH Comput. Arch. News*, vol. 35, no. 2, pp. 460–469, 2007.

[13] E. Czeck and D. Siewiorek, "Effects of transient gate-level faults on program behavior," in *Proc. Int. Symp. Fault-Tolerant Comput.*, 1990, pp. 236–243.

[14] K. Seongwoo and A. Somani, "Soft error sensitivity characterization for microprocessor dependability enhancement strategy," in *Proc. Int. Conf. Dependable Syst. Netw.*, 2002, pp. 416–425.

[15] M. Maniatakos and Y. Makris, "Workload-driven selective hardening of control state elements in modern microprocessors," in *Proc. VLSI Test Symp.*, 2010, pp. 159–164.

[16] G. Georgakos, P. Huber, M. Ostermayr, E. Amirante, and F. Ruckerbauer, "Investigation of increased multi-bit failure rate due to neutron induced SEU in advanced embedded SRAMs," in *Proc. IEEE Symp. VLSI Circuits*, 2007, pp. 80–81.

[17] J. Maiz, S. Hareland, K. Zhang, and P. Armstrong, "Characterization of multi-bit soft error events in advanced srams," in *Proc. IEEE Int. Electron Devices Meeting*, 2003, pp. 21–4.

[18] T. Criswell, P. Measel, and K. Wahlin, "Single event upset testing with relativistic heavy ions," *IEEE Trans. Nuclear Sci.*, vol. 31, no. 6, pp. 1559–1561, Dec. 1984.

[19] R. Reed, M. A. Carts, P. W. Marshall, C. J. Marshall, O. Musseau, P. J. McNulty, D. R. Roth, S. Buchner, J. Melinger, and T. Corbiere, "Heavy ion and proton-induced single event multiple upset," *IEEE Trans. Nuclear Sci.*, vol. 44, no. 6, pp. 2224–2229, Dec. 1997.

[20] R. Koga, S. Pinkerton, T. Lie, and K. Crawford, "Single-word multiple-bit upsets in static random access devices," *IEEE Trans. Nuclear Sci.*, vol. 40, no. 6, pp. 1941–1946, Dec. 1993.

[21] E. Touloupis, J. Flint, V. Chouliaras, and D. Ward, "Study of the effects of SEU-induced faults on a pipeline-protected microprocessor," *IEEE Trans. Comput.*, vol. 56, no. 12, pp. 1585–1596, Dec. 2007.

[22] F. Faure, R. Velazco, M. Violante, M. Rebaudengo, and M. Reorda, "Impact of data cache memory on the single event upset-induced error rate of microprocessors," *IEEE Trans. Nuclear Sci.*, vol. 50, no. 6, pp. 2101–2106, Dec. 2003.

[23] M. Rebaudengo, M. Reorda, and M. Violante, "An accurate analysis of the effects of soft errors in the instruction and data caches of a pipelined microprocessor," in *Proc. Des., Autom. Test Eur. Conf.*, 2003, pp. 602–607.

[24] N. George, C. Elks, B. Johnson, and J. Lach, "Transient fault models and AVF estimation revisited," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2010, pp. 477–486.

[25] S. Mukherjee, J. Emer, T. Fossum, and S. Reinhardt, "Cache scrubbing in microprocessors: Myth or necessity?" in *Proc. IEEE Pacific Rim Int. Symp. Dependable Comput.*, 2004, pp. 37–42.

[26] Y. S. Dhillon, A. U. Diril, and A. Chatterjee, "Soft-error tolerance analysis and optimization of nanometer circuits," in *Proc. Des., Autom.Test Eur.*, 2008, pp. 389–400.

[27] B. Zhang, W. Wang, and M. Orshansky, "Faser: Fast analysis of soft error susceptibility for cell-based designs," in *Proc. Int. Symp. Quality Electron. Des.*, 2006, pp. 755–760.

[28] R. Rao, K. Chopra, D. Blaauw, and D. Sylvester, "An efficient static algorithm for computing the soft error rates of combinational circuits," in *Proc. Des., Autom. Test Eur.*, 2006, pp. 1–6.

[29] S. Krishnaswamy, G. Viamontes, I. Markov, and J. Hayes, "Accurate reliability evaluation and enhancement via probabilistic transfer matrices," in *Proc. Des., Autom. Test Eur.*, 2005, pp. 282–287.

[30] E. Normand, "Single event upset at ground level," *IEEE Trans. Nuclear Sci.*, vol. 43, no. 6, pp. 2742–2750, Dec. 1996.

[31] S. Mukherjee, J. Emer, and S. Reinhardt, "The soft error problem: An architectural perspective," in *Proc. 11th Int. Symp. High-Performance Comput. Arch.*, 2005, pp. 243–247.

[32] A. Biswas, P. Racunas, J. Emer, and S. Mukherjee, "Computing accurate AVFs using ACE analysis on performance models: A rebuttal," *IEEE Comput. Arch. Lett.*, vol. 7, no. 1, pp. 21–24, Jan.-Jun. 2008.

[33] P. Sanda, J. Kellington, P. Kudva, R. Kalla, R. McBeth, J. Ackaret, R. Lockwood, J. Schumann, and C. Jones,"Soft-error resilience of the IBM POWER6 processor," *IBM J. Res. Dev.*, vol. 52, no. 3, pp. 275–284, 2008.

[34] M. Maniatakos, C. Tirumurti, R. Galivanche, and Y. Makris, "Global signal vulnerability (GSV) analysis for selective state element hardening in modern microprocessors," *IEEE Trans. Comput.*, vol. 61, no. 9, pp. 1361–1370, Aug. 2012.

[35] S. Mitra, N. Saxena, and E. McCluskey, "A design diversity metric and analysis of redundant systems," in *Proc. IEEE Int. Test Conf.*, Sep. 1999, pp. 662–671.

[36] J. Black, P. Dodd, and K. Warren, "Physics of multiple-node charge collection and impacts on single-event characterization and soft error rate prediction," *IEEE Trans. Nuclear Sci.*, vol. 60, no. 3, pp. 1836–1851, Jun. 2013.

[37] N. Seifert, P. Slankard, M. Kirsch, B. Narasimham, V. Zia, C. Brookreson, A. Vo, S. Mitra, B. Gill, and J. Maiz, "Radiation-induced soft error rates of advanced CMOS bulk devices," in *Proc. Int. Reliability Phys. Symp.*, 2006, pp. 217–225.

[38] G. Saggese, N. Wang, Z. Kalbarczyk, S. Patel, and R. Iyer, "An experimental study of soft errors in microprocessors," *IEEE Micro*, vol. 25, no. 6, pp. 30–39, Nov./Dec. 2005.
[39] 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 *Proc, Int. Conf. Dependable Syst. Netw.*, 2004, pp. 61–70.
[40] R. Rajaraman, J. Kim, N. Vijaykrishnan, Y. Xie, and M. Irwin, "Seat-la: A soft error analysis tool for combinational logic," in *Proc. Int. Conf. VLSI Des.*, 2006, pp. 409–502.
[41] K. Ramakrishnan, R. Rajaramant, N. Vijaykrishnan, Y. Xie, M. Irwin, and K. Unlu, "Hierarchical soft error estimation tool (HSEET)," in *Proc. Int. Symp. Q. Electron. Des.*, 2008, pp. 680–683.
[42] L. Gwennap, "Intel's P6 uses decoupled superscalar design," *Microprocessor Report*, vol. 9, no. 2, pp. 9–15, 1995.
[43] S. Narayanasamy, C. Pereira, H. Patil, R. Cohn, and B. Calder, "Automatic logging of operating system effects to guide application-level architecture simulation," in *Proc. Int. Conf. Measurement Model. Comput. Syst.*, 2006, pp. 227–238.
[44] M. Maniatakos, N. Karimi, A. Jas, Tirumurti, and Y. Makris, "Instruction-level impact analysis of low-level faults in a modern microprocessor controller," *IEEE Trans. Comput.*, vol. 60, no. 9, pp. 1160–1173, Sep. 2011.
[45] A. Dixit and A. Wood, "The impact of new technology on soft error rates," in *Proc. Int. Reliability Phys. Symp.*, 2011, pp. 5B–4.

Michail Maniatakos received the BSc and MSc degrees in computer science and embedded systems from the University of Piraeus, Piraeus, Greece, in 2006 and 2007 respectively, as well as the MSc, MPhil and PhD from Yale University, New Haven, CT, in 2008, 2009, and 2012, respectively. He is currently an Assistant Professor of Electrical and Computer Engineering at New York University Abu Dhabi, Abu Dhabi, UAE, and a Research Assistant Professor at the NYU Polytechnic School of Engineering. He is the Director of the Modern Microprocessor Architectures (MoMA) lab in NYU Abu Dhabi and an author of multiple publications in the IEEE Transactions and conference papers. His research interests include robust microprocessor architectures, hardware security and heterogeneous microprocessor architectures. He is a member of the IEEE.

Chandrasekharan (Chandra) Tirumurti is currently a Research Scientist with the Validation and Test Solutions group at Intel Corporation based in Santa Clara, CA. His current focus is on strategic manufacturing test initiatives for mainstream CPUs. An alumnus of Indian Institute of Technology, Kharagpur, India, he has wide experience in many areas of CAD and design, including simulation, datapath synthesis, defect oriented testing and fault tolerance. He has published several papers in the areas of Test and Fault Tolerance. He mentors funded research and SRC projects actively for Intel and is an avid cricketer. He is a member of the IEEE.

Maria K. Michael received the BS and MS degrees in computer science and the PhD degree in electrical and computer engineering from Southern Illinois University, Carbondale, CA, in 1996, 1998 and 2002, respectively. She taught as a Lecturer at the ECE Department at Southern Illinois University from 2001 to 2002, and as an Assistant Professor of Computer Science and Engineering at the University of Notre Dame from 2002 to 2003. She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Cyprus, Cyprus. Her current research interests are in the area of test and reliability of modern digital VLSI circuits, embedded systems and multi-core architectures, including on-line/adaptive testing for multi-core designs, test and diagnosis for various faults (including timing and other deep-submicron/nanometer induced faults), single- and multi-bit upset analysis and protection, symbolic techniques for test and verification (BDDs and SAT) and parallel methods/algorithms for EDA tools. She has served on the technical program committees of several international conferences and workshops and is a reviewer for a number of scholarly journals and international conferences. She is a co-recipient of a Best Paper Award of MSE2009. She is a member of the IEEE.

Yiorgos Makris received the Diploma of computer engineering and informatics from the University of Patras, Patras, Greece, in 1995, and the MS and PhD degrees in computer science and engineering from the University of California, San Diego, CA, in 1997 and 2001, respectively. After spending over 10 years as a faculty of electrical engineering and of computer science at Yale University, he moved to The University of Texas at Dallas where he is currently an Associate Professor of electrical engineering, leading the Trusted and Reliable Architectures (TRELA) research group. His current research interests include the application of machine learning and statistical analysis methods towards developing reliable and trusted integrated circuits, with particular emphasis in the analog/RF domain. He is also investigating error detection and correction methods for modern microprocessors, as well as novel computational modalities using emerging technologies. He serves on the organizing and program committees of many conferences in the areas of test, security, and reliability and he is the program chair of the 2013 and 2014 IEEE VLSI Test Symposium (VTS). He is a recipient of the the Best Paper Award from the 2013 Design Automation and Test in Europe (DATE) conference. He is a senior member of the IEEE.