Assessing SEU Vulnerability via Circuit-Level Timing Analysis

Kypros Constantinides Stephen Plaza Jason Blome Bin Zhang[1]
Valeria Bertacco Scott Mahlke Todd Austin Michael Orshansky[1]

Advanced Computer Architecture Lab
University of Michigan
Ann Arbor, MI 48109
{kypros, splaza, jblome, valeria, mahlke, austin}@umich.edu

[1]Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX, 78712
bzhang@ece.utexas.edu
orshansky@mail.utexas.edu

ABSTRACT

Recently, there has been a growing concern that, in relation to process technology scaling, the soft-error rate will become a major challenge in designing reliable systems. In this work, we introduce a high-fidelity, high-performance simulation infrastructure for quantifying the derating effects on soft-error rates while considering microarchitectural, timing and logic-related masking, using realistic workloads on a CMP switch design. We use a gate-level model for the CMP switch design, enabling us to inject faults into blocks of combinational logic. We are then able to track logic-related and time-related fault masking, as well as microarchitectural-related fault masking, at the architecture level. We find out that for complex designs, logic-and time-related fault masking account for more than 50% of the masked faults. We also observe that only 3-4% of the injected faults propagate an error at the design's output and cause an error in the application's execution, resulting in a derating factor of 30. From our experiments, we also demonstrate that soft-error derating effects highly depend on the design's characteristics and utilization.

1. INTRODUCTION

As silicon technologies scale into the nanometer regime, there is a growing concern among system designers about the susceptibility of future-generation semiconductor-based digital systems to the effects of transient faults caused by energetic particles (such as neutrons and alpha particles) [24, 12, 25]. Historically, the major concern about transient faults effects was focused on memory structures. This was because memory structures dominated chip area and unstructured combinational logic exhibited low soft-error rates. Recently, numerous works [20, 7] argue that in future-generation systems, the soft-error rate due to transient faults in combinational logic will grow and will therefore constitute a major challenge for system designers to provide high-reliability designs.

Numerous techniques already exist to deal with the effects of transient faults, most typically by providing soft-error detection via redundancy [1, 16, 3, 21, 17]. However, all of these techniques have a significant impact on performance, power dissipation, die size and design time. Consequently, early in the design cycle, system designers need to trade-off between the reliability level provided by these approaches and their implementation cost. On the one end, a design with inadequate soft-error protection may prove useless due to its poor reliability. On the other end, a design with excessive protection may make the product uncompetitive in cost and/or performance. The best way for the system designers to balance this trade-off is by having accurate soft-error estimates for their designs. Although much work has been done by the device community to get raw soft-error rates for current and future technologies devices [20, 5, 4, 8], system designers still lack an accurate assessment of soft-error rates for their designs. This is mainly due to the lack of analysis tools that accurately model the possible masking of transient faults as they enter into a complex computing system.

The need for accurate soft-error rates of complex circuits places high demands on the analysis infrastructure. The analysis framework must accurately gauge detailed circuit-level phenomena in order to correctly model the introduction, propagation and possible masking of transient faults. For example, a transient glitch which manifests in a block of combinational logic may not affect the latch at the end of the associated logic chain, either because the glitch is time-masked and therefore doesn't reach the input of the latch at the latch's sensitive timing window, or it is logic-masked and does not propagate to the latch's input. Previous approaches used for analyzing much of the work in reliable systems design utilize high-level models of micro-architectural components. The advantages of these simplistic models are flexibility and speed. However, the accuracy of these models is put in question as they cannot capture important aspects of transient fault propagation in complex systems, such as timing and logic masking.

In this work, we introduce a high-fidelity, high-performance simulation infrastructure for estimating soft-error rates of complex computing systems. We permit simulations that react to circuit-level reliability phenomena on a cycle-by-cycle basis, while simulating with sufficient speed in order to examine entire workloads. Our simulation framework is capable of asynchronously injecting voltage pulses of various durations at the gate level and modeling the many possible ways the faults can be masked through microarchitectural, logic and timing masking. For evaluating the simulation framework, we select an essential aspect of a typical chip multiprocessor (CMP) system: a single CMP

router switch. We further decompose the CMP switch into functional modules with different circuit characteristics and study how different circuit characteristics tolerate transient faults and how they affect the soft-error derating factor. We use realistic workloads with various design utilizations and study how they affect fault tolerance and soft-error deration. In addition, we give estimations of the design failure rates for current and future technologies based on derated soft-error rates. Finally, we perform experiments to study the effects on soft-error derating when a single energetic particle strike causes multiple faults to manifest.

The remainder of the paper is organized as follows. Section 2 details the various ways a transient fault might get masked. Section 3 describes the simulation framework and the transient fault model used. In Section 4, we outline the CMP switch architecture used. In Section 5, we describe our experimental methodology and the experimental results. In Section 6, we go through the related work and finally, Section 7 gives conclusions and suggests directions for future work.

2. SOFT ERROR MASKING

Fortunately, not all transient faults affect the final outcome of a program. In order for a transient glitch caused by a particle strike in combinational logic to affect correct computation, it must first change the value of the latch at the end of the associated logic chain and then propagate to the design's output. There are five basic phenomena that might prevent the glitch from affecting the design's output thereby masking the transient fault:

- **Logic Masking:** A faulty glitch is logically masked when it fails to affect the input value of a latch because it gets blocked by a following gate whose output is completely determined by its other input values (i.e. a faulty logical zero fitting a two input OR gate when its other input value is a logical one). It's clear that the fewer levels of logic between two latches, the lower the probability that a faulty glitch will be logically masked. Therefore, it is expected that as the pipelines of microprocessors get deeper and clock frequencies get shorter, the levels of logic in a microprocessor's pipeline stages will become fewer and logic masking within a given pipeline stage will occur less frequently [20]. Furthermore, as the pulse duration of the faulty glitch gets larger, the probability that it will be logically masked is lower. This is because the values of the other inputs of the blocking gate that determine its output value must remain unchanged for an extended period of time.

- **Timing Masking:** A faulty glitch is timing masked if it affects the input of a latch, only in the period of time that the latch is not sensitive to its input value. It is clear that larger pulse durations lessen the probability of timing masking. Furthermore, the period of time that the latch is sensitive to its input value is determined by the technology's setup and hold times. Therefore, as microprocessors' clock frequencies get shorter and setup and hold times become a larger fraction of the clock period, it is expected that the timing masking will become a less frequent phenomenon.

- **Electrical Masking:** A faulty glitch is electrically masked if its pulse is attenuated by subsequent logic gates due to electrical properties, and as a result it does not affect the input value of a latch. As with logic masking, electrical masking depends on the number of levels of logic between two latches. Hence, as the length of the faulty glitch gets larger (or the number of levels of logic become fewer), the probability that the transient fault will get masked is less. Furthermore, as transistors get smaller and faster, the effects on pulse attenuation by logic gates are reduced, and electrical masking is also expected to reduce.

- **Microarchitectural Masking:** Even when a latch's value is altered by a transient fault (either due to a faulty transient glitch manifested in a combinational logic block that didn't get masked and, therefore, successfully changed the latch's value or due to a particle strike directly flipping the latch's value) the transient fault can still be masked and be transparent to the application's correct execution as a result of microarchitectural masking. For example, if a register's bit is flipped by a particle strike and subsequently the register's value gets overwritten by a new value without the wrong value ever having been read, then the fact that, for a period of time, the register's value was incorrect is transparent to the application's correct execution and the transient fault is successfully masked by microarchitectural phenomena. There is also the case where an incorrect value latched in a flip-flop is subsequently masked either by electrical, logic or time-related masking in the next stage, thereby is prevented from propagating to the design's output. Since the state of the design is incorrect for at least one cycle, we consider this as microarctitectural-related masking, no matter the way the transient fault was subsequently masked.

- **Software Masking:** Even when a transient fault propagates an error to the output of the microprocessor, the error can be masked at the software level [13]. For example, when an error is propagated outside of the microprocessor's domain and causes an incorrect value at a memory location, which is then overwritten by the application or the operating system without having been used, then the error is software-masked and it is transparent to the correct execution of the application or the operating system. The quantitative analysis of software masking is out of the scope of this paper, as it occurs outside of a microprocessor's domain.

These five masking phenomena significantly derate the estimated raw soft-error rates for complex circuit designs but, at the same time, because their analysis need to accurately model and track cycle-by-cycle details of circuit activity, the aforementioned phenomena place high demands on the soft-error analysis infrastructure.

3. MODELING AND ANALYSIS OF SOFT ERRORS

In this section, we describe our simulation infrastructure for modeling and analyzing transient faults. In subsection 3.1, we first give a detailed description of the simulation infrastructure and methodology used, and subsequently in

subsection 3.2, we give details about the transient faults model used.

3.1 A Gate-Level Soft-Error Simulation Infrastructure

Our simulation infrastructure for evaluating the impact of transient faults on a complex digital design is shown in Figure 1. It consists of an event-driven simulator which is comprised of three major modules:

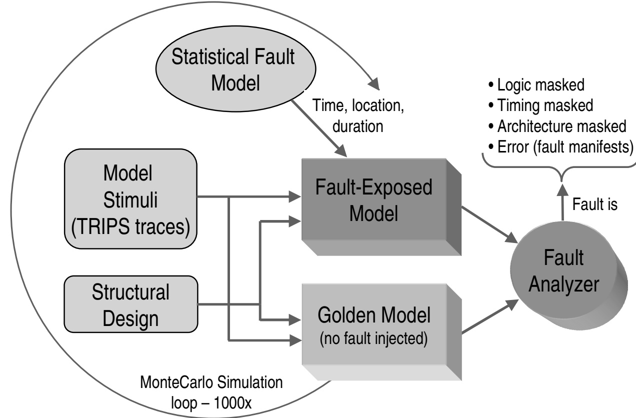

Figure 1: ***Simulation infrastructure.*** **The simulation infrastructure consists of an event-driven simulator, comprised of the design under test, a fault generator and a fault analyzer. The fault generation is based on a transient fault model and the model stimuli are based on realistic workloads.**

- **Design Under Test:** The design under test is the structural gate-level description of the design (netlist) obtained by synthesizing the Verilog description of the design using Synopsis CAD tools. In the simulation framework, we instantiate two copies of the design's structural gate-level description, one of which is subjected to fault injection (fault-exposed model) and the other is kept intact (golden model).
- **Fault Generator:** The fault generator is capable of injecting voltage pulses of various durations at any gate in the design and flipping the value of any individual flip-flop of the design. This is done by forcing values on the design nets and latches during the simulation. Faults are uniformly distributed at the time of occurrence, spatial location (which net they affect) and duration of the event.
- **Fault Analyzer:** At the end of each clock cycle, all outputs and sequential elements of the design under test are compared with the golden model. The fault analyzer distinguishes four cases: (1) if a mismatch is detected in the outputs of the design, then an error has occurred; (2) if a mismatch is found in the sequential elements, but not at the outputs, then the fault was microarchitecturally masked; we derive that all the other injected faults have been either (3) time masked or (4) logic masked. To discern between these two cases, we re-run the simulation by injecting the same faults synchronized with the design's clock cycle using a one cycle duration: the faults injected in this second simulation cannot be time-masked, hence, the difference between these two analyses gives the correct partition between time-and logic-masked faults.

Our simulation infrastructure does not model the electrical attenuation of the voltage pulses, caused by particle strikes, as they traverse through a chain of logic. Previous work [20], studied in detail the effects of electrical attenuation on transient voltage pulses and concluded that the primary effect of electrical masking is to screen out marginal pulses and the degradation effect on pulses with sufficient strength is minimal. In general, it is expected that the fraction of masked transient faults due to electrical masking is minimal and it would be decreased with device scaling.

The maximum propagation delay of an injected fault to propagate to the design's output in the simulated CMP switch design is 160 clock cycles. Therefore, we let the CMP switch to warm up for the first ten thousand cycles and then start injecting faults in the design with intervals of at least 200 clock cycles between each injected fault. For each injected fault we keep track of its effects on the design by monitoring the design's state and outputs and comparing them with those of golden model.

In order to gain statistical confidence of the results, we run the simulations many times in a Monte Carlo modeling framework. Each simulation setup was run 1000 times with different random number generation seeds, so that the distribution of injected faults in space, time and duration were different for each run.

3.2 Transient Faults Model

An important aspect of the simulation framework is the way it models transient faults. The system uses a pulse-based model for transient faults which are caused by energetic particle strikes. The transient voltage pulses are classified into five classes based on their duration. The model uses a sixth class of faults to model the flip of a flip-flop's value, when the flip-flop is hit directly by an energetic particle strike. The transient fault generation is modeled using a six-variable random process, where random variables model the uniformly distributed arrival rate of each class of faults. For both current and future technology processes the mean inter-arrival times for each class of faults was derived by [20] and by detailed SPICE simulations.

4. CMP SWITCH ARCHITECTURE

The design that we have chosen to experiment with is an essential part of a Chip Multiprocessor (CMP): an interconnection network switch. We selected this design because it is much less complex than a microprocessor yet it is not too simplistic in that it contains many representative components of larger designs, including finite state machines, buffers, control logic, and buses.

The interconnection network switch chosen is a wormhole switch that collectively implements the routing and flow-control functions required to buffer and forward 32-bit packet flits in a mesh and 2D torus interconnect network topologies. The design is derived from the design described in [15, 2]. Our design is pipelined at the flit level, where a

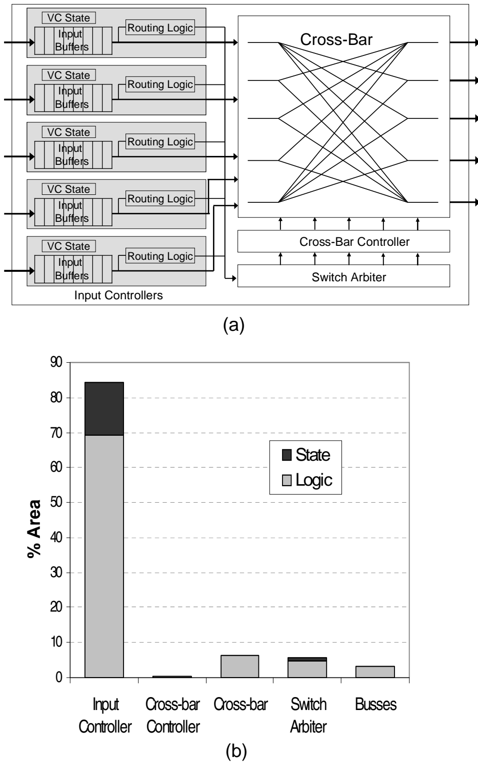

Figure 2: ***Baseline CMP switch design.*** **In (a) a high-level block diagram for a wormhole interconnection switch is presented. It consists of five input controllers, a cross-bar, a switch arbiter, and a cross-bar controller. In (b) an area breakdown of the four switch modules and the input/output buses is given. The area is broken down into state and logic.**

head flit proceeds through the routing and output virtual-channel allocation stages and all packet flits pass through the switch allocation and switch traversal pipeline stages. A high-level block diagram of the interconnection switch is depicted in Figure 2(a).

The implemented interconnection switch is composed of four major functional modules. The first being the input controller module which is responsible for determining the virtual output channel to which the packet flits must be routed. It uses a dimensional order routing algorithm (routing logic). In addition, the input controller keeps track of the state of the virtual channel by determining whether it is idle, receiving a route request by a head flit, requesting a virtual channel allocation from the switch arbiter or transmitting a packet (VC State). The input controller is also augmented with buffers used for buffering flits when it is unable to allocate and route the incoming flits (input buffers). For the specific design, each of the five input controllers is augmented with eight 32-bit buffers.

The second major functional module, the switch arbiter, allocates the output virtual channels to the input controllers, depending on their requests. To avoid starvation, the switch arbiter uses a priority matrix, in a way that the input controller that least recently requested an output virtual channel has the highest priority during the output virtual channels allocation. Furthermore, the switch arbiter uses a credit-based flow control by keeping track of the number of buffers available on the next hop. After the switch arbiter allocates the output virtual channels to the input controllers, the third functional module, the cross-bar controller, decodes the allocate signals. The final and fourth functional module is the cross-bar, which provides common paths between the switch's input and output ports.

The design is specified in Verilog and synthesized using the Synopsys tools to create a gate-level netlist. To provide a better understanding of the design, Figure 2(b) provides an area breakdown of the four modules along with the input/output buses. The entire switch is composed of approximately 10k gates. The height of each bar represents the percentage of the total area for each module, and the bars are broken down into two pieces that indicate, for each module, the fraction of area devoted to logic/state. In this design, the input controller is obviously dominant in area. There are five input controllers, thus the fraction of area is magnified. The design is also heavily dominated by logic as compared to state: 84% logic verses 16% state.

5. EXPERIMENTAL METHODOLOGY AND RESULTS

5.1 Experimental Methodology

The design under test is described using the Verilog HDL and synthesized to a gate-level representation using the Synopsys Design Compiler [22]. The simulator used is the Synopsys VCS 7.1.1. To evaluate the design's exposure to transient faults, we used realistic workloads from communication traces derived from the TRIPS architecture [19]. We used traffic traces for 13 benchmarks from the SPEC2000 benchmark suite, six benchmarks from the MediaBench suite, and one synthetic high-utilization traffic trace (hi_util), as shown in Table 1.

Each traffic trace consists of 32-bit packet communication transactions, where each communication transaction is specified by the incoming input channel, the header of the flit with the destination node (needed for the routing of the packet), the data of the packet and the clock cycle that the packet is injected into the switch. The mean switch utilization for each traffic trace is specified by the ratio between the number of communication transactions and the number of clock cycles needed to complete all the communication transactions.

As shown in Table 1, the time needed for running a benchmark is relatively small (the simulations were ran on a 1GHz SunBlade 1500 with 1GB RAM). This enables us to run the simulations many times in a Monte Carlo modeling framework, in order to gain statistical confidence of the results.

5.2 Experimental Results

In Figure 3, we classify the injected faults into four categories: (1) the faults that caused an error, (2) faults that were microarchitecturaly masked, (3) faults that were timing masked, and (4) faults that were logically masked. This classification is presented per transient fault type. The first type (column) are state-bit flipping transient faults, while

Benchmark	# of Comm. Transactions	Clock Cycles to Completion	Mean Util. (%)	Sim. Time (s)
ammp	64664	356452	18.14	758.53
art	44050	549648	8.01	531.64
bzip2	69014	686084	10.06	823.22
compress	60306	1304340	4.62	715.28
equake	57608	641020	8.99	664.00
gzip	76538	572473	13.37	909.39
m88ksim	58335	814768	7.16	689.42
mcf	76866	704404	10.91	920.78
mgrid	71786	258160	27.81	822.63
parser	80338	1429220	5.62	951.34
swim	47105	236272	19.94	540.58
twolf	59529	1009868	5.89	731.58
vortex	70848	439440	16.12	842.99
adpcm	79257	1454536	5.45	969.68
dct	70786	175120	40.42	836.07
hydro2d	62026	302728	20.49	728.78
mpeg2encode	58368	533684	10.94	680.13
tomcatv	51796	208952	24.79	602.62
turb3d	38695	416412	9.29	448.12
hi_util	40005	57150	70.00	505.05

Table 1: *Simulated Benchmarks.* The benchmark pool consists of 13 benchmarks from the SPEC2000 suite, six from the MediaBench suite and one synthetic benchmark. For each benchmark, we list the number of communication transactions, the clock cycles needed to complete these transactions, the switch's mean utilization and the simulation time in seconds.

the next five are transient glitches with a pulse duration of 100%, 80%, 60%, 40% and 20% of the design's clock period, respectively. The last column is the classification for all types of transient faults combined. The presented data are averages over all the SPEC2000 and MediaBench benchmarks. The bit-flipping faults have no logic-related or time-related masking, since they are injected directly into the design's state.

We notice that in the design there is high microarchitectural masking that reaches 95% of the injected faults. This suggests that even if the logic-related and time-related masking of transient glitches in combinational logic blocks were more infrequent and glitches altered latches' values more frequently, the derating factors of transient faults would still be high due to microarchitectural masking effects. From the graph, we can also notice that as the pulse duration of the faulty glitches gets smaller, the logic related masking fraction gets larger. We attribute this to fact that when the glitch pulse is smaller the possibility of being logically masked by a subsequent gate is higher. The reason for this higher probability is that the other inputs of the blocking gate, which logically masks the fault, need to stay at this blocking state for a lesser period of time. As the pulse duration of the faulty glitches gets smaller the time related masking fraction is increasing as well. This is because glitches with smaller pulse durations have less of a chance to reach

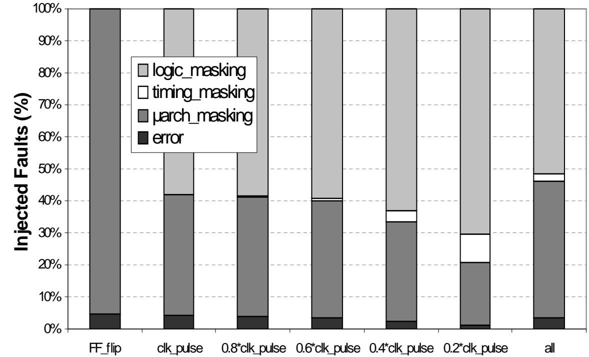

Figure 3: *Classification of transient fault effects.* The graph classifies the effect of transient faults on: logically-masked, time-masked, microarchitecturally-masked, and those faults that caused errors. The classification is made for six different categories of transient faults with different characteristics, and for the combination of all the different fault categories.

a latch's input during the latching window.

From Figure 3, we observe that the degree and type of masking of transient faults depends on the characteristics of each fault. For example, the percentage of transient glitches with small pulse duration that cause an error in the application's execution (1.2%) is four times smaller than the equivalent percentage for state-bit flipping transient faults (4.8%). Furthermore, the logic masking of transient faulty glitches with small pulse duration is 21% more than the logic masking of transient glitches with pulse duration equal to the design's clock period. Although timing masking is negligible (less than 1%) for glitches with pulse duration larger than half of the design's clock period, it becomes important for small glitches and it accounts for up to 9% of the masked faults.

When all transient fault types are combined, we observe that 51.7% of them are logic masked, 2.2% are timing masked, and 42.9% are microarchitectural masked. The remaining 3.2% of the injected transient faults propagate the fault at the output of the design and consequently cause an error in the application's execution. The 24% of the total error rate results from the state-bit flipping transient faults, while the rest 76% results from transient faults injected into combinational logic.

The 3.2% of the injected faults that cause an error in the application's execution can also be represented as a derating factor (the term vulnerability factor is also used in related works [14]) of 31.25, which means that one of every 31.25 energetic particle strikes that hit the design will cause an error in the application's execution. The derating factor for each benchmark is shown in Figure 4. We notice that the derating factors for the SPEC2000 and MediaBench benchmarks fluctuate between 22 and 45, which correspond to error rates of 4.5% and 2.2% respectively. On average, both SPEC2000 and MediaBench benchmarks exhibit derating factors of 31. The synthetic high-utilization benchmark exhibits a much lower derating factor of 12 due to less microarchitectural masking of transient faults. Even

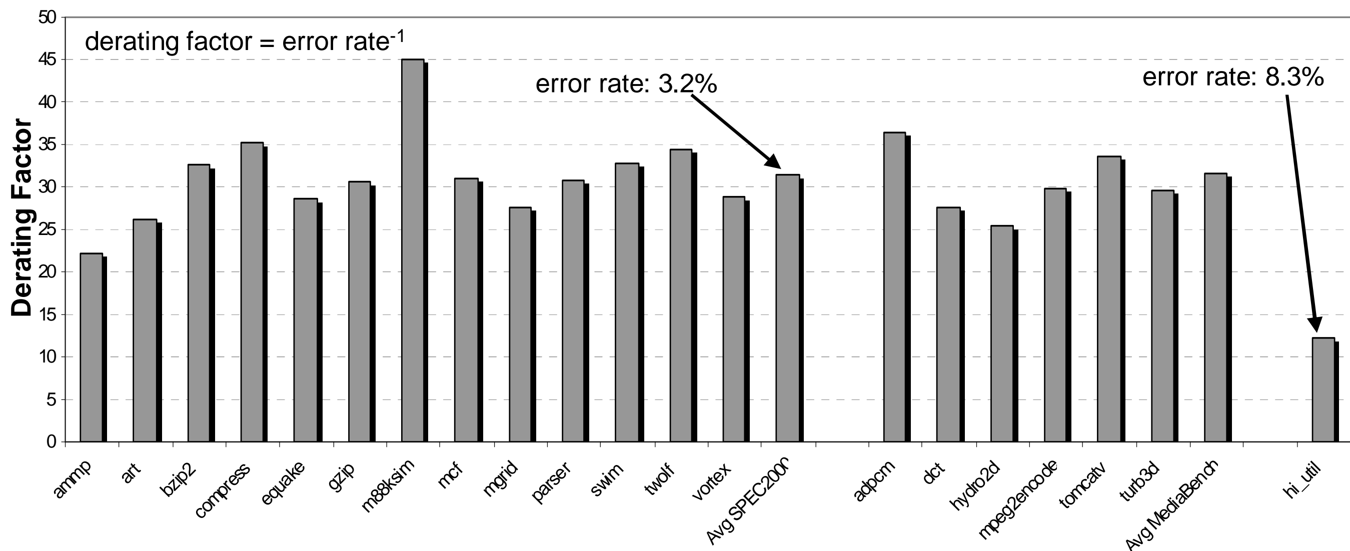

Figure 4: ***Derating Factor per benchmark.*** **The graph presents the derating factors (the inverse of the error percentage) for each individual simulated benchmark, along with the averages for the SPEC2000 and MediaBench benchmark suites.**

though this artificially created benchmark keeps the switch's buffers full and a number of packets get routed by the switch every cycle, still the transient faults that cause an error to the application's execution is relatively low at 8.3%. Although the error rate for the high-utilization benchmark is low, it is still 2.6 times larger than the average error rate over the realistic workloads. This brings to light the importance of evaluating a design's tolerance to transient faults using realistic workloads.

It 's interesting to notice from Figure 3 that when logic and time related masking is not taken into account, 95% of the injected transient faults are microarchitecturally masked leading to a derating factor of 20. However, when logic and time related masking is taken into account the derating factor is increased by 56% to 31.25. This shows the importance of logic/timing masking analysis in order to get more accurate estimates of soft-error derating factors.

The aspects of a design that are important in its tolerance to transient faults are the transistor density (number of transistors per area unit), the raw soft-error rate of a single device in the design, the design's size, and the design's clock frequency. All of these design parameters change with process technology scaling, except for the design size of microprocessor chips which, based on ITRS [6] projections, will stay constant for future generation microprocessor designs. In order to derive a design's failure rate, we need to know all of the above along with the design's SER derating factor. Based on the design's masking derated SER, derived from simulations using our simulation infrastructure, and projections for the characteristics of future designs from ITRS [6], we estimated the failure rates of the CMP switch design for current and future process technologies. We note that the accuracy of these failure rate estimations are subject to the degree that the fault model used for raw SER (described in Section 3.2) accurately estimates raw SER of single devices for current and future process technologies and to the accuracy of ITRS's projections of future design characteristics.

Figure 5, presents the estimated failure rates of six different process technologies for varied workloads. Across the different process technologies the architecture of the CMP switch design is kept the same, and the presented failure rates are for an area equal to a chip die. The Y axis represents the design's failure rate in FITs (Failures In Time), which is the number of failures in a billion hours of operation, and is plotted in logarithmic scale. The top line is the raw SER, where each energetic particle strike that hits the design is assumed to cause an error at the application's execution. The other lines project the estimated failure rates for two different designs for varied workloads considering fault masking. Since the design's clock frequency is one of the major design aspects that affect its tolerance to transient faults (and typical interconnection networks are clocked with much slower frequencies than microprocessors), we estimate the failure rates for two designs: one with projected clock frequencies for interconnection networks and a second with projected clock frequencies for microprocessors (though the architecture of the design is the same). As we can see, from the graph, the failure rates for the higher-clock frequency design are an order of magnitude larger than that of the lower-clock frequency design. The reason for this is that while the pulse duration of a given transient fault is equal with the clock cycle of the high frequency design, it is only a fraction of the clock cycle of the low frequency design. As shown in Figure 3, transient faults whose pulse duration is only a small fraction of the clock cycle exhibit significantly lower error rates than the ones with pulse durations equal to the clock cycle. This indicates the importance of the frequency at which the design is clocked in relation to its ability to tolerate transient faults.

In Figure 5, for each design we present the failure rates for different workloads: the synthetic high-utilization benchmark, the average failure rate over the SPEC2000 benchmarks, and the average failure rate over the MediaBench

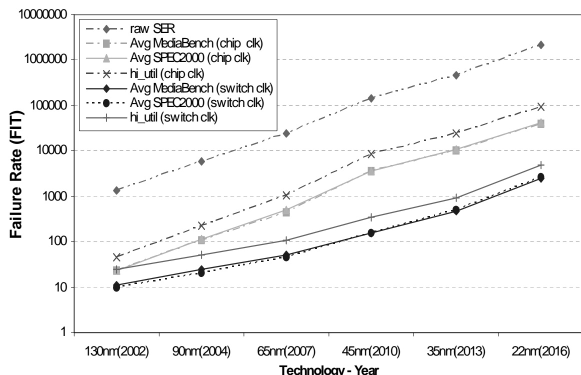

Figure 5: ***Projection of failure rates due to transient faults.*** **The estimated failure rates due to transient faults are projected for six different process technologies, for designs with different clock frequencies and for various workloads. The raw SER is projected as well.**

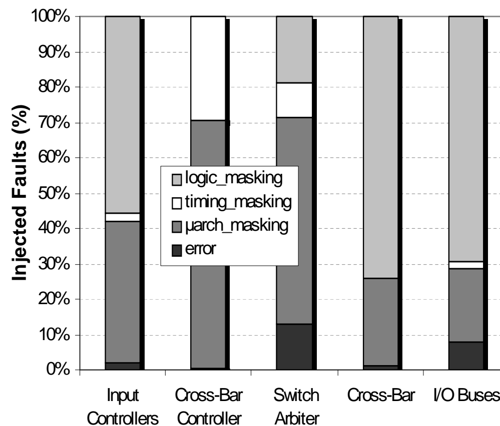

Figure 6: ***Transient fault distribution per switch component.*** **The distribution of injected transient faults is shown for the five different components of the CMP switch. Each component, based on its characteristics, exhibits different error rates and fault-masking distributions.**

benchmarks. The failure rates for the SPEC2000 and MediaBench benchmarks overlap, which indicates that realistic workloads exhibit similar masking effects to transient faults. The failure rate for the high-utilization benchmark is higher than that of the realistic benchmarks, which correlates with the lower derating factor showed in Figure 4.

Since the die area of a CMP is not dominated by the interconnection network, in order to estimate the failure rate of a CMP interconnection network the failures rates presented in Figure 5 must be multiplied by the area percentage of the interconnection network over the total die area. For example, in a typical scenario of a chip multiprocessor, where the 20% of the chip area is allocated to the interconnection network then the failure rates corresponding to the interconnection network are five times less than the ones presented in Figure 5. Of course, in order to estimate the total failure rate for the chip multiprocessor, the estimated failure rate for the remaining 80% of the chip area has to be added to the interconnection network's failure rate.

Figure 6, shows the distribution of the injected transient faults for the five different components of the CMP switch. The data referenced by the graph suggests that transient faults can have different effects on components with different characteristics. The most vulnerable component of the switch design is the switch arbiter with an error rate of 12.8%. The high vulnerability of the switch arbiter is inherent to its functionality, to arbitrate the output channels of the switch to the input channels. Hence, an error in the switch arbiter would lead to misrouting a flit. The microarchitectural-masking fraction for this component accounts for the cases where an output channel was arbitrated incorrectly to an input channel, but during the particular clock cycle no useful flit was routed through the output channel and the corresponding input channel. Thus, the erroneous arbitration is transparent to the application's correct execution.

The next most vulnerable component in the design is the I/O busses with error rates of 7.9%. We attribute this to the fact that, in a case where a transient fault strikes an output bus transferring a useful flit, the fault is propagated directly to the design's output causing an error. Otherwise, if the bus does not transfer a useful flit the fault is microarchitecturally masked. Furthermore, time-related, logic-related and microarchitectural-related masking can manifest in cases where faults occur at input busses. The input controllers exhibit moderate error rates at 2.1% and the cross-bar controller and cross-bar components exhibit negligible error rates. As shown in Figure 2(b) the switch's area is dominated by the five input controllers, hence, the pattern of transient fault distribution for the hole design, shown in Figure 3, matches with the input controller's distribution.

An interesting experiment in studying the effects of transient faults is to observe the effects of transient faults on the design when a single energetic particle strike that hits the design affects more than one neighboring devices. Using our simulation infrastructure, we modeled this phenomenon by injecting multiple glitches at neighboring gates in combinational logic blocks and by flipping multiple neighboring state bits at the time that a strike hits the design. Since no complete place and route was performed on the design, the way we modeled multi-fault strikes was by injecting faults in multiple gates in the same functional module in the case of strikes on combinational logic and by flipping contiguous bits in the case of strikes on sequential logic. In [11], the authors measure the probability that a single energetic particle strike will affect multiple SRAM cells for the 130nm and 90nm process technologies, and conclude that there is a 2% probability of affecting two cells and a 0.1% probability of affecting three cells. These rates are expected to grow with process-technology scaling, although multi-fault strikes in combinational logic will be less common than in SRAM due to the larger area and wider transistors employed in combinational logic. Due to the lack of a model for multi-fault strikes for latches and logic gates in combinational logic blocks, for future process technologies, we are over-pessimistic and each strike results to multiple faults in our simulations.

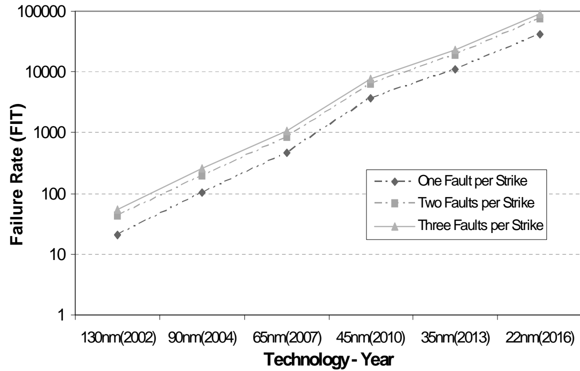

Figure 7: *The failure rates of multiple faults per energetic particle strike.* The projected failure rates for different number of faults per strike are shown for six different processes technologies.

Figure 7 shows the estimated failure rates for current and future process technologies for one, two and three faults per strike. The corresponding derating factors are shown in Figure 8(a). From the results, we observe that the derating factor is decreased, when a single strike causes two and three faults, by 75% and 112% respectively. The concluding observation is presented in Figure 8(b), where we show that when multiple faults are injected into the design in a single cycle, more faults manifest as errors in the design's operation. Specifically, by injecting more faults per strike, more faults pass the first level of masking, which is logic masking, and then get masked by timing masking. Thus, logic masking percentages decrease as more faults manifest per strike and, respectively, timing masking percentages increase. Furthermore, more faults successfully alter the value of a latch but subsequently get masked by microarchitectural masking and, as expected, more faults eventually cause an error at the application's execution, thereby causing both microarchitectural masking and error percentages to increase as more faults manifest per strike.

6. RELATED WORK

Several works studied with varied degrees of simulation fidelity the soft-error's impact on a system's reliability. The impact of soft errors on embedded microprocessors was studied in [18]. The simulation infrastructure used injected transient faults in a gate-level model of a DLX-like microprocessor. However, the focus of that work was to compare the soft-error vulnerability of control versus speculative logic, and combinational versus sequential logic. The main purpose was not to quantify the transient fault masking effects yet the concluding outcome of this work concurs with our observations that there are inherently high soft-error derating factors in complex designs.

In [23], the authors focus on the effects of transient faults and their manifestation within a typical modern microprocessor. The authors present a fine-grained latch-level Verilog processor model and they randomly flip single-bit state elements in order to accurately characterize microarchitectural fault masking and thereby to identify which portions of the microarchitecture are most susceptible to error when a transient fault hits the processor. The main difference between

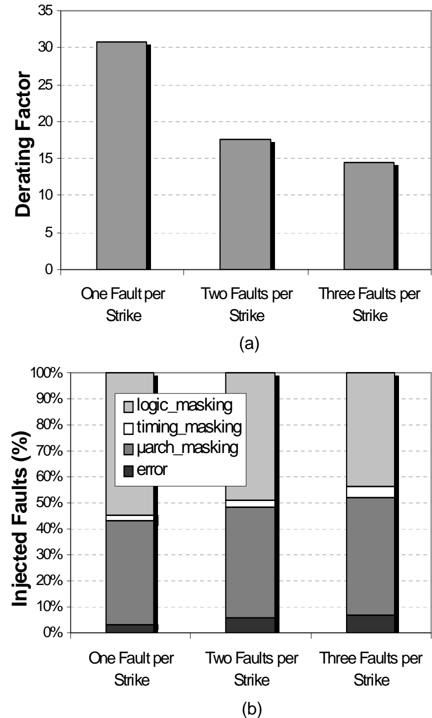

Figure 8: *Modeling multiple faults per energetic particle strike.* (a) shows the corresponding derating factors and (b) the distribution of injected transient faults.

this work and ours is that our simulation infrastructure models the design under test at a gate-level model enabling accurate consideration of timing-related and logic-related masking effects in addition to microarchitectural masking. Our technique provides a higher degree of simulation fidelity and accuracy. A similar simulation approach was used in [9] to evaluate the transient-fault vulnerability of a PicoJava II microprocessor.

In [26], an analytical soft-error rate analysis methodology (SERA) is introduced, based on a modeling and analysis-based approach employing probability theory, circuit simulation, graph theory and fault simulation. The major drawback of this methodology is that its applicability is limited to relatively simple designs (in the hundreds of gates). Furthermore, the methodology concentrates only on logic-related masking of transient faults in blocks of combinational logic.

Another work studying the effects of soft errors on processor reliability is SoftArch [10]. This work quantifies the impact of technology scaling on the processor soft-error rate, taking the architecture level derating effects and workload characteristics into consideration for different structures in a modern superscalar microprocessor. However, this work studied soft-error derating at an architecture-level model, disregarding soft-error derating effects such as logic-related and time-related masking.

7. CONCLUSIONS AND FUTURE WORK

With process technology scaling, there is a growing con-

cern that soft-error rate will grow and will constitute a major challenge for designing reliable systems. In this work, we use a high-fidelity, high-performance simulation infrastructure to study the derating effects on soft-error rates by considering microarchitectural, timing and logic related fault masking. Our experimental results show that for complex designs there is significant fault masking, with derating factors as high as 30. We also show that soft-error derating effects highly depend on the design's characteristics and its utilization. Our observations suggest that the soft-error rate threat to future systems' reliability might have been overstated and that system designers need to evaluate their design's soft-error tolerance with high-fidelity simulation infrastructures by considering low-level derating effects, and basing their design-protection decisions on more accurate soft-error rates.

Our future directions include studying the soft-error derating effects for several designs with different characteristics, as well as higher-complexity designs, such as high-performance microprocessor cores. Due to the size of the higher-complexity designs, studying the soft-error derating effects for such designs with the existing simulation framework might be cumbersome and lead to long simulation runs. We are planning to overcome this limitation by employing a hierarchical simulation framework. In this hierarchical simulation framework, a detailed gate-level simulation is actively employed for limited number of clock cycles and to a subset of the design. In this manner, windows of accuracy are achieved when faults occur, but the majority of the simulation can run fast.

Acknowledgments: This work is supported by grants from NSF and Gigascale Systems Research Center. We would also like to acknowledge Li-Shiuan Peh for providing us access to CMP Switch models, and Doug Burger for providing CMP network reference traces.

8. REFERENCES

[1] T. M. Austin, DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design, In *Proceedings of the 32nd Annual International Symposium on Microarchitecture (MICRO)*, November 1999.

[2] W. J. Dally and B. Towles, Principles and Practices of Interconnection Networks, *Morgan Kaufmann*, 2004.

[3] M. Gomaa and T. N. Vijaykumar, Opportunistic Transient-Fault Detection, In *Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA)*, June, 2005.

[4] S.Hareland, J. Maiz, M.Alavi, K.Mistry, S.Walstra, and C.Dai, Impact of CMOS Scaling and SOI on soft error rates of logic processes, *VLSI Technology Digest of Technical Papers*, 2001.

[5] P. Hazucha, et al., Measurements and analysis of SER-tolerant latch in a 90-nm dual-V/sub T/CMOS process, In *IEEE Journal of Solid-State Circuits*,39(9), 2004.

[6] International Technology Roadmap for Semiconductors (ITRS) 2004 update, Document available at *http://public.itrs.net/*, 2004.

[7] T. Juhnke, H. Klar, Calculation of the soft error rate of submicron CMOS logic circuits, In *IEEE Journal of Solid-State Circuits*, 30, 1995

[8] T.Karnik, B.Bloechel, K.Soumyanath, V.De, and S.Borkar, Scaling trends of Cosmic Rays induced Soft Errors in static latches beyond 0.18um, *VLSI Technology Digest of Technical Papers*, 2001.

[9] S. Kim, A. K. Somani, Soft error sensitivity characterization for microprocessor dependability enhancement strategy, In *Proceedings of the Internanional Conference on Dependable Systems and Networks (DSN)*, 2002.

[10] X. Li, S. V. Adve, P. Bose, and J. A. Rivers, SoftArch: An Architecture Level Tool for Modeling and Analyzing Soft Errors, In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, June, 2005.

[11] J. Maiz, et al., Characterization of Multi-Bit Soft Error Events in Advanced SRAMs, In *Digest of International Electron Devices Meeting (DIEDM)*, 2003.

[12] T. May and M. Woods, Alpha-particle-induced soft errors in dynamic memories, *IEEE Transactions on Electronic Devices*, vol. 26, no. 2, 1979.

[13] A. Messer, et al,. Susceptibility of Commodity Systems and Software to Memory Soft Errors *IEEE Transactions on Computers*,53(12), 2004.

[14] S. S. Mukherjee, C. T. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor, In *Proceedings of the 36th Annual International Symposium on Microarchitecture (MICRO)*, December 2003.

[15] Li-Shiuan Peh, Flow Control and Micro-Architectural Mechanisms for Extending the Performance of Interconnection Networks. Ph.D. Thesis, Stanford University, August 2001.

[16] S. K. Reinhardt and S. S. Mukherjee, Transient Fault Detection via Simultaneous Multithreading, In *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA)*, June, 2000.

[17] E. Rotenberg, AR-SMT: A microarchitectural approach to fault tolerance in microprocessors, In *Proceedings of Fault-Tolerant Computing Systems*, 1999.

[18] G. P. Saggese, A. Vetteth, Z. Kalbarczyk, R. K. Iyer, Microprocessor Sensitivity to Failures: Control vs Execution and Combinational vs Sequential Logic, In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, 2005.

[19] K. Sankaralingam, et al,. Exploiting ILP, TLP and DLP with the Polymorphous TRIPS Architecture, In *Proceedings of the 36th Annual International Symposium on Microarchitecture (ISCA)*, December, 2003.

[20] P. Shivakumar, et al., Modeling the effect of technology trends on the soft error rate of combinatorial logic, In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, 2002.

[21] J. Smolens, J. Kim, J. C. Hoe, and B. Falsafi, Efficient Resource Sharing in Concurrent Error Detecting Superscalar Microarchitecture, In *Proceedings of the 37th International Symposium on Microarchitecture (MICRO)*, December, 2004.

[22] Synopsys Design Compiler Document available at *http://www.synopsys.com/products/logic/design_compiler.html*

[23] N.J. Wang, et al., Characterizing the effects of transient faults on a high-performance pipeline, In *Proceedings of the Internanional Conference on Dependable Systems and Networks (DSN)*, 2004.

[24] J. F. Ziegler, Terrestrial cosmic rays, *IBM Journal of Research and Development*, 40(1), 1996.

[25] J. Z. et al,. IBM experiments in soft fails in computer electronics, *IBM Journal of Research and Development*, vol. 40, pp. 318, Jan. 1996.

[26] M. Zhang, N. R. Shanbhag, A Soft Error Rate Analysis (SERA) Methodology, In *Proceedings of the International Conference on Computer Aided Design*, November, 2004.