Voltage Noise in Production Processors

VOLTAGE VARIATIONS ARE A MAJOR CHALLENGE IN PROCESSOR DESIGN. HERE, RESEARCHERS CHARACTERIZE THE VOLTAGE NOISE CHARACTERISTICS OF PROGRAMS AS THEY RUN TO COMPLETION ON A PRODUCTION CORE 2 DUO PROCESSOR. FURTHERMORE, THEY CHARACTERIZE THE IMPLICATIONS OF RESILIENT ARCHITECTURE DESIGN FOR VOLTAGE VARIATION IN FUTURE SYSTEMS.

Vijay Janapa Reddi
Svilen Kanev
Wonyoung Kim
Simone Campanoni
Michael D. Smith
Gu-Yeon Wei
David Brooks
Harvard University

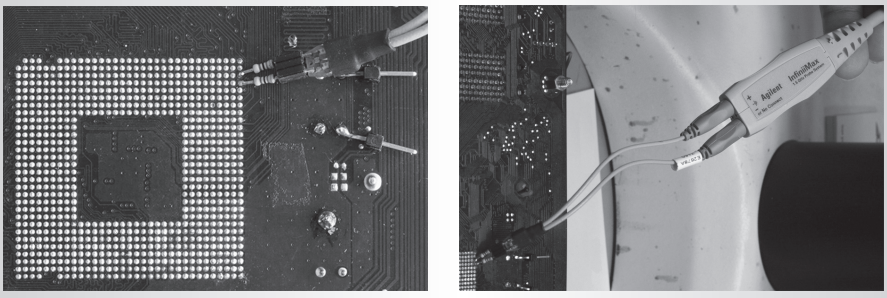
In the era of power-constrained processor design, supply voltage noise is becoming a dominant problem. Designers are aggressively using clock-gating techniques to reduce energy consumption. Nonzero impedance in the power delivery network, sudden fluctuations due to clock gating, and workload activity changes lead to large, unpredictable changes in supply voltage at runtime. Voltage fluctuations beyond the operating margin can lead to timing violations. If the processor must always avoid such voltage emergencies, its operating margin must be large enough to tolerate the absolute worst-case voltage swing. As such, the industry-wide standard is to allocate large operating voltage margins sufficient to guarantee robustness.

Today's production processors use operating voltage margins that are nearly 20 percent of nominal supply voltage.[1] Unfortunately, operating voltage margins decrease peak processor performance and lower power efficiency. To reduce the gap between nominal and worst-case operating voltages, researchers are trying to design the processor for the typical-case voltage swing. Instead of setting the operating voltage margin according to some extreme activity, such as a dI/dt (rate of current change) power virus, the solution is to relax the margin to a more typical voltage-swing level. In the rare event of a voltage emergency, fail-safe error-detection and error recovery circuits dynamically detect and correct timing violations. In this way, designers can use aggressive margins to maximize processor performance and power efficiency.

Much research in such resilient architecture design has been based on simulations or proof-of-concept chips. Next to circuit-level techniques that discuss error detection and recovery, there's no well-known body of work that demonstrates and investigates the potential for this line of work using production chip data. Although simulation efforts are valuable, they suffer from constraints such as program execution length or the extent to which the models are representative of production processors. Moreover, nearly all architectural-level noise mitigation efforts have been focused on single-core execution. In today's multicore era, it is important to characterize the effect of interactions across cores.

Therefore, we emphasize measuring voltage noise activity in an actual production chip. We found significant opportunity for typical-case design using resilient architectures, even in the presence of voltage noise

Published by the IEEE Computer Society



(a) (b)

Figure 1. Measurement setup: connecting to on-die voltage pins via low impedance paths (a); sensing voltage using a differential probe with ultra low loading (b).

and assuming various error recovery costs. This level of characterization leads us to propose a software mechanism to dampen voltage noise because it will be a dominant component of the PVT (process, voltage, and thermal) reliability problem in future systems.

Measuring voltage noise

Using only off-the-shelf components that include a differential probe and high bandwidth oscilloscope, we connect to on-die silicon voltage sensing pins of an Intel Core 2 Duo processor via its external package pins (see Figure 1). We studied a desktop Core 2 Duo Processor (E6300) on a Gigabyte GA-945GM-S2 motherboard. While we constrain our analysis to this processor and motherboard setup, the general methodology can extend to other platforms as well. We successfully validated this measurement setup by reconstructing the platform's impedance profile, which has a resonance peak between the typical 100-MHz to 300-MHz range.

By synchronizing measurement collection with program execution, we can perform full-length program analysis, including introspecting and characterizing voltage noise of a processor under fully operational settings. Our setup lets us run through entire suites of real programs to completion rather than relying on simulation to observe activity over just a few million instructions. This experimental setup lets us characterize single-core and multicore noise activity of the Core 2 Duo chip under various micro-benchmarks, and single-threaded, multi-threaded, and multiprogram executions.

Findings from a real chip

The ability to unobtrusively monitor chip activity not only lets us make interesting observations but also validate the potential for resilient architecture designs based on prior work. The discussion covers the noise characteristics of real-world programs as they run to completion: 29 single-threaded SPEC CPU2006 workloads, 11 Parsec programs, and 29 $\times$ 29 multiprogram workload combinations from CPU2006.

Worst-case design is overly conservative

The worst-case operating voltage margin typically used in the industry is overly conservative. Figure 2 shows a cumulative histogram of voltage samples for the processor. The deviation of voltage samples is shown relative to the nominal supply voltage. Each line in the graph corresponds to a benchmark execution. Runtime voltage droops for the set of programs we benchmarked can be as large as 9.6 percent (see the minimum droop marker in Figure 2). The processor can tolerate a worst-case droop of approximately 14 percent below the nominal voltage before running into correctness errors. In order to determine this value, we progressively undervolt the processor while maintaining its clock frequency. This ultimately forces the processor into a functional error, which we detect when the processor fails

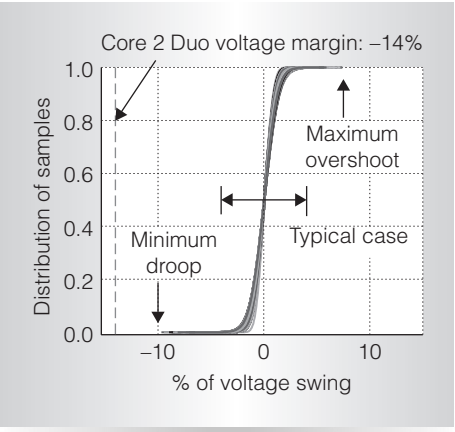


Figure 2. Typical-case voltage swing versus the absolute worst-case margin. The Core 2 Duo margin is overly conservative compared to typical-case behavior.

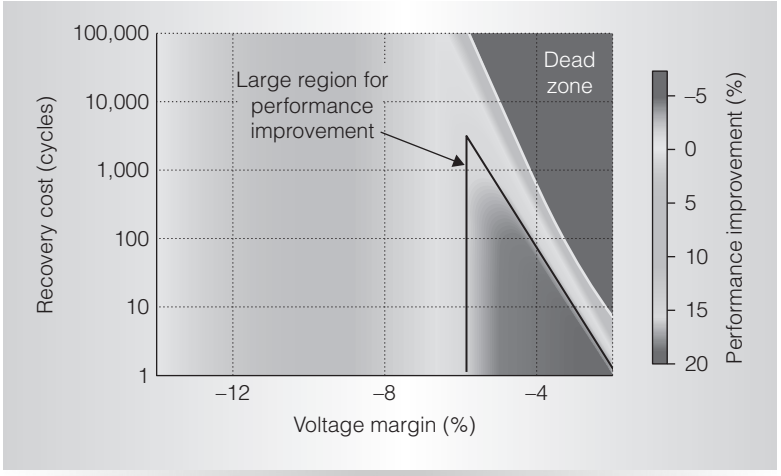


Figure 3. Performance gains from enabling aggressive voltage margins, assuming some fail-safe recovery. Typical-case margins can enable significant performance gains.

stress-testing under multiple copies of the power virus. Real program voltage swings at about 10 percent are dangerously close to this failure limit. Therefore, large worst-case margins are necessary to guarantee robustness. However, such large droops occur infrequently. Most samples were within 4 percent of the nominal voltage. The typical-case marker in Figure 2 identifies this range. Only a small fraction of samples (0.06 percent) lie beyond this typical-case region.

Resilient architectures enable large improvements

If we optimize the architecture for typical-case voltage swings and rely on error recovery hardware to correct infrequent emergencies, we can achieve large performance improvements. We consider performance, although tighter voltage margins also enable power savings. Bowman et al. show that removing a 10 percent operating voltage margin leads to a 15 percent improvement in clock frequency.[2] Using an analytical performance model that combines measurement data with hypothetical operating voltage margins and the cost of rolling back execution, we quantify the gains of a resilient architecture design. The heatmap in Figure 3 shows sweeps of performance improvement over worst-case design at any given combination of recovery cost and margin. The map's intensity corresponds to average gains for all 881 executions. We see significant room for improvement between the margins of negative 6 percent to negative 2 percent.

Voltage droops correspond to program activity

Voltage noise activity varies according to program characteristics. Assuming a 2.3 percent voltage margin, only for characterization purposes, we observe droops per 1,000 clock cycles across each program. Figure 4 shows that droop counts vary noticeably, which indicates a heterogeneous mix of noise characteristics in CPU2006. More interestingly, droops correlate to program execution stalls. We combine various hardware performance counters for stalls, such as branch prediction and cache miss events, into a single metric called *stall ratio*. The linear correlation coefficient between stalls and droops is 0.97. This is an important finding that validates previous research. Assuming that resilient architectures are promising, researchers have been attempting to understand what factors influence voltage noise activity as a means of mitigating error recovery frequency. Simulation-based studies have shown that microarchitectural events that cause sudden processor stalls result in voltage fluctuations. We omit citations here for brevity; for details, please refer to our paper for the 43rd Annual IEEE/ACM International Symposium on Microarchitecture.[3] Our chip measurements

imply that we can indeed predict and potentially react to droop activity using microarchitectural events.

Voltage noise in multicore systems

Very little is known about voltage fluctuations in multicore systems. In such systems, microarchitectural event activity across cores causes interference that leads to chip-wide transient voltage swings that are much larger than those in their single-core counterparts. This is a problem in resilient architecture designs because a transient voltage droop anywhere on the shared power grid can inadvertently affect all cores.

We characterize noise in multicores initially using microbenchmarks that stimulate the processor with specific hardware events such as TLB (translation look-aside buffer) misses only, L1 (level one) or L2 cache miss events only, and so forth. But here we demonstrate that such behavior is visible even at a macroscopic, full-program-execution level. Depending on the set of programs running simultaneously on separate cores that are tied to the same power supply source, voltage droop activity can vary considerably. Figure 5 shows aggregate droop activity for our dual-core chip in which both cores share a common power source and are actively running instances of benchmark 473.astar, albeit with different starting offsets. The x-axis of the graph refers to this offset as *scheduling time offset*. The graph is a convolution of two execution windows. The figure shows an example of destructive interference in which the noise when two cores are simultaneously active is smaller than the noise during single-core execution. It also shows an example of constructive interference, which is just the opposite.

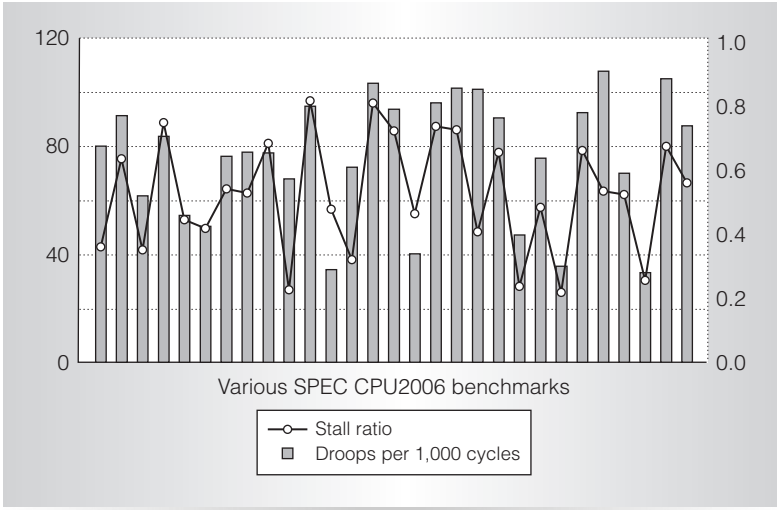


Figure 4. Heterogeneous mix of voltage noise activity and the relationship to program activity stalls. There is a general relationship trend between noise behavior and program activity.

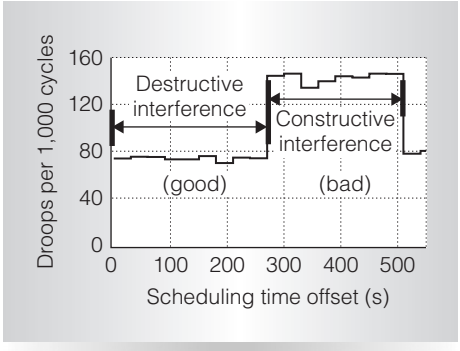


Figure 5. Noise interference in multicore systems, as two instances of benchmark 473.astar are running simultaneously on Core 0 and Core 1 of the processor, offset by different amounts of time.

Long-term implications

We examine how typical-case design gains scale in future systems. To model voltage noise scaling, we remove decoupling package capacitance from current chips and use the new chips as a heuristic into future technology nodes. We show a significant decrease in performance gain opportunities due to increasingly frequent recovery. To address this issue in multiprocessors, we propose a voltage-noise-aware thread scheduler that decreases the total number of voltage emergencies.

Studying future systems by decap removal

Aiming to model future systems in terms of voltage noise, we physically modified processors that were identical to our production chips. The end result was similar chips, but with less decoupling package capacitance, which we can easily remove from a chip's landside. Figure 6 shows a subset of the resulting processors. Decreasing capacitance on the package increases its impedance and,

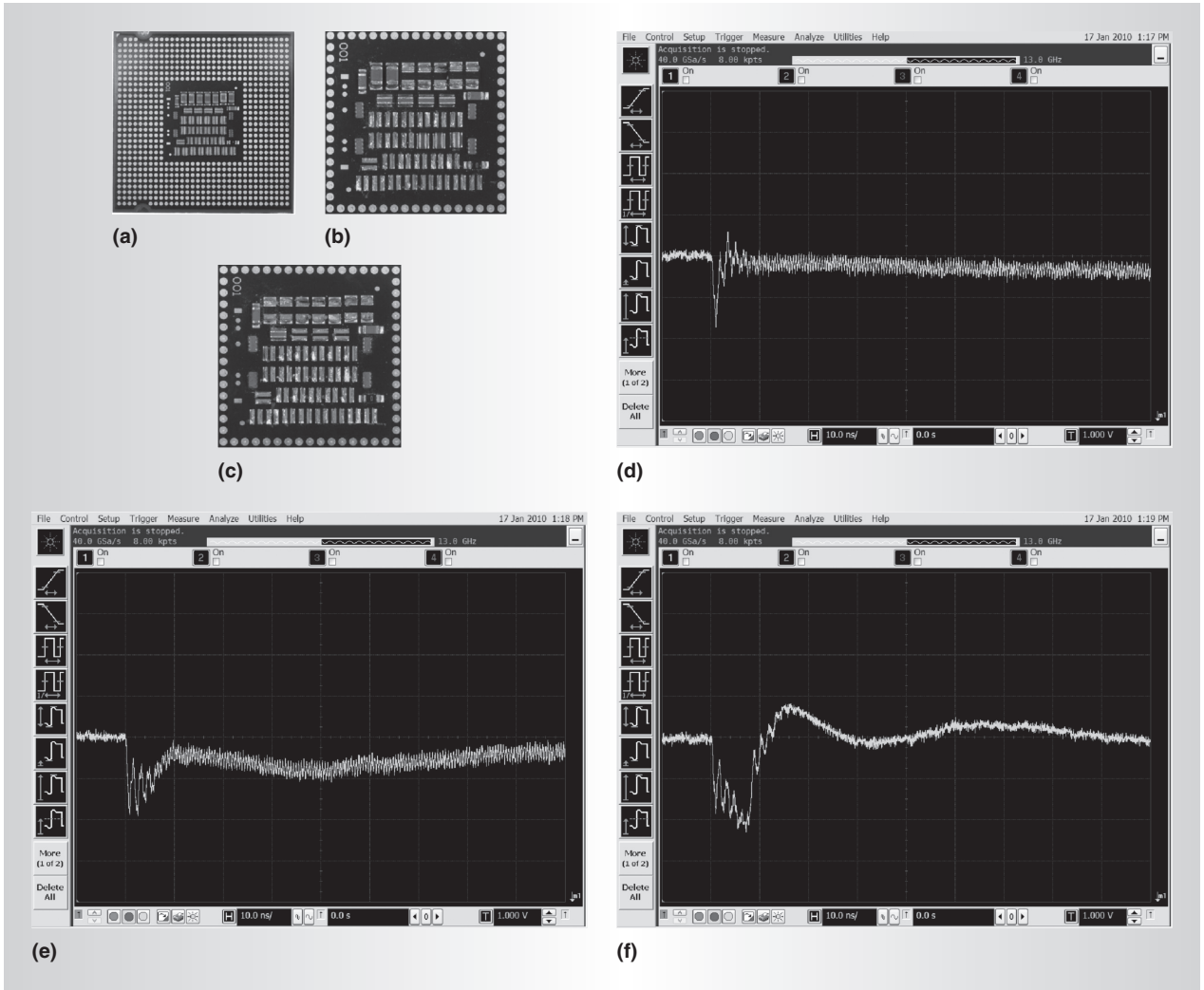


Figure 6. Package-side chip photos show decreasing amounts of package capacitance for $Proc_{100}$, $Proc_{25}$, and $Proc_3$ (a-c) and, as a result, the increasing magnitude of voltage swing when the processor is subject to a hard reset (d-f).

consecutively, the magnitude of the voltage swings. Thus, it serves as a heuristic that lets us project into future technology nodes to understand the impact of amplifying voltage swings on resilient architecture designs. The technique does not translate absolutely to voltage noise in future nodes—it is indeed a crude heuristic that ignores any nonlinear effects that might arise. Nevertheless, it is sufficient for studying the scaling of voltage noise effects in full-program execution.

Here, we consider two new modified processors—$Proc_{25}$ and $Proc_3$ (subscript values correspond to remaining package capacitance)—that let us approximate two future processor generations. More precisely, we chose these particular chips because their maximal voltage swing roughly corresponds to simulation data for our test processor, scaled down to 32 nm and 22 nm, respectively.

Diminishing benefits from average-case design

As we extrapolate the benefits of resilient microarchitecture designs into future nodes using $Proc_{25}$ and $Proc_3$, we anticipate an alarming decrease in the corresponding performance gains from aggressive margins. Figure 7 illustrates the performance we can expect from future nodes under the same experimental conditions as shown in Figure 3. We see diminishing gains due to worsening

voltage swings, because processors in the future will experience more frequent voltage emergencies. Thus, more recoveries are necessary that penalize performance. The region for performance improvement we see in Figure 3 (margins between negative 6 percent and negative 2 percent) diminishes as we go into future nodes in Figures 7a and 7b.

This implies that, to retain the same level of performance improvement as in today's $Proc_{100}$, future processors will need to use more fine-grained recovery mechanisms. For instance, in Figure 3, designers could use a 1,000-cycle recovery mechanism with $Proc_{100}$ to reap a 15 percent performance improvement. But with $Proc_{25}$, they would have to achieve a tenfold reduction in recovery cost implementation to just 100 cycles. $Proc_3$ requires even further reductions to about 10 cycles per recovery to maintain the 15 percent improvement.

The problem with implementing fine-grained recovery is that it is severely intrusive. Such schemes require invasive changes to traditional microarchitectural structures. They add area and cost overheads that make design and validation even more complicated than they already are. Such techniques might apply to a niche high-performance computing market in which design and validation costs are tolerable due to high reliability, availability, and service demands.

An alternative is to constrain ourselves to relying on coarser-grained mechanisms, at least in the commodity processor market segment. Coarse-grained recovery mechanisms require less invasive changes, which makes them more cost-effective to implement. This matters for commodity processors because in such a market a chip's price to performance ratio matters.[4] Typically, coarser-grained mechanisms implement rollback via some flavor of checkpoint recovery. A major benefit of coarser recovery mechanisms is that some form of checkpoint recovery is already shipping in today's systems for soft-error tolerance.[5,6] Voltage emergencies are an emerging form of more deterministic transient errors. Moreover, newer applications leverage and reuse this general-purpose hardware for tasks such as debugging, testing, and so forth.[7,8]

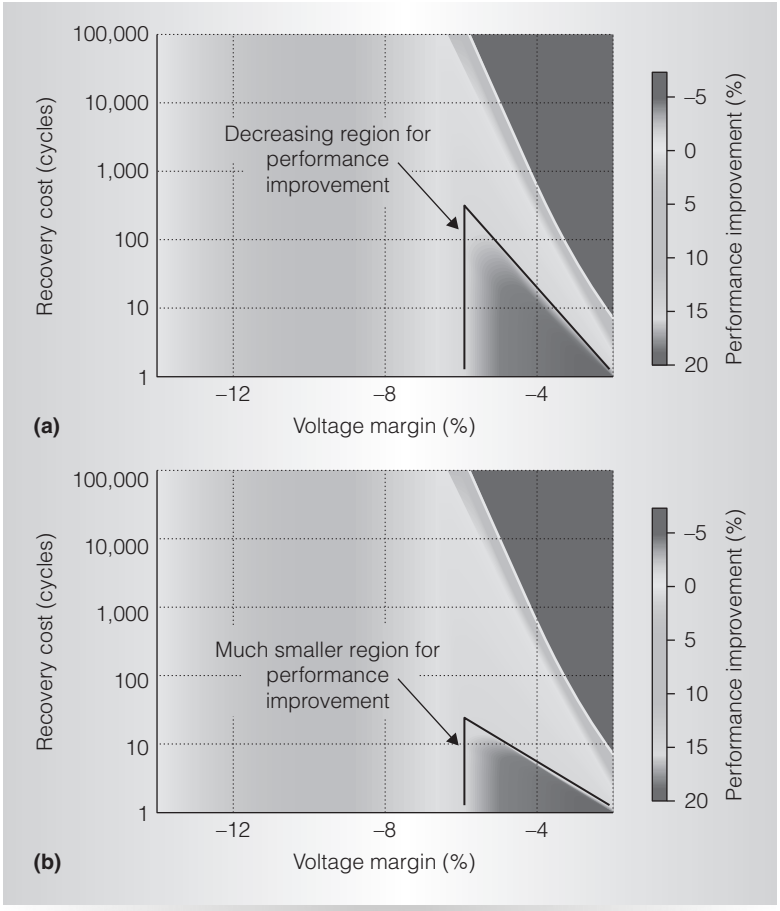


Figure 7. Decreasing room for improvement via resilient architectures: $Proc_{25}$ (a); $Proc_3$ (b). In the future, finer-grained recovery mechanisms will be necessary to sustain performance.

Software-aided coarse-grained recovery

Based on our projections, we might need to develop new techniques that allow more coarse-grained resilient architectures to avoid losing potential gains. Part of our contribution is voltage-noise-aware thread scheduling. Our technique is hardware guaranteed but software assisted. Hardware provides a fail-safe guarantee to recover from errors, whereas software reduces the frequency of this fail-safe invocation, thereby improving performance by reducing recovery penalties. Such a software solution complements hardware rather than substituting hardware.

We propose a thread scheduling scheme that tracks emergency activity and decides which threads to run together partially on the basis of current noise levels. We do not

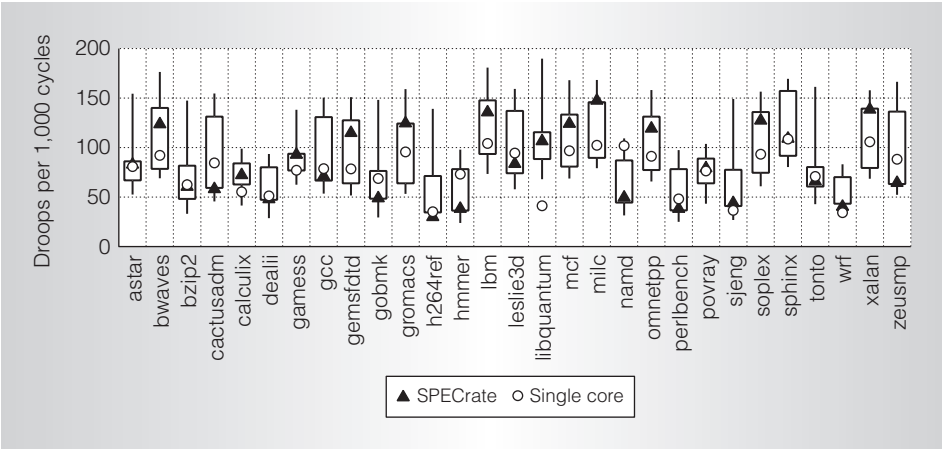


Figure 8. Droop variance across single-core and dual-core runs. In dual-core experiments, droops per 1,000 cycles frequently exceed noise activity compared to single-core runs.

intend for this scheduling policy to replace performance-centric scheduling, but to be an addition that exposes the implications of increasingly frequent checkpoint recovery on performance. In fact, a large body of coscheduling research optimizes resource access to shared L2 and L3 caches, which are performance critical. Similarly, our multicore characterization results suggest that the processor supply voltage is another such shared resource that has potential for interference (see Figure 5). Intuitively, due to the common power supply plane, activity on one core can trigger recovery on other cores.

To evaluate the potential of scheduling, we examined the noise behavior of coscheduled pairs of SPEC2006 programs on the $Proc_3$ chip. Figure 8 shows the range of voltage droops for all possible schedules. The circular markers represent single-core activity, whereas the triangular ones correspond to two instances of the same benchmark running together (also known as SPECrate). Most importantly, destructive interference is certainly present, with some box-plot data even falling below single-core noise activity. Overall, we observe both destructive and constructive interference across the entire suite. If we relax the definition of destructive interference from single core to multicore, then room for coscheduling improvement expands—there is opportunity to do better than SPECrate in more than half of the coschedules. This suggests that a scheduling solution can reduce the number of recovery mechanism invocations by scheduling threads that exhibit destructive interference behavior. Implementing such a scheme on the software level is appropriate because voltage emergencies exhibit strong phase behavior even at relatively large time scales. Data that's not shown here for brevity, but which is available in our Micro 43 paper,[3] confirms that even on a 60-second granularity we observe voltage noise phases, thus allowing the operating system scheduler enough time to make decisions on the basis of noise behavior.

Due to the lack of existing resilient architectures, we investigated the usefulness of a software-aided solution via analytical modeling and oracle-based analysis. In reality, we would implement the feedback path that tracks emergency activity either as a dedicated voltage-noise hardware performance counter or via an analytic model similar to the stall ratio metric in Figure 4.

In our simulation analysis we find that an explicit noise-only thread scheduler can reduce recovery overheads. In comparison, a performance-only scheduler increases machine throughput but is agnostic to noise-related effects. However, a synergic solution is possible: a scheduler that optimizes for the $IPC/Droops^n$ metric effectively co-optimizes both performance and noise behavior. It resembles optimizing for energy-delay product or energy-delay-squared

product. We envision this as a configurable scheme—a high value of the parameter n is suitable for machines with coarse-grained recovery mechanisms when each emergency has a high performance cost, whereas lower n implies fine-grained recovery. Designers can use such a general solution when implementing different grades of recovery schemes according to processor class. Server-class or high-performance systems typically use fine-grained recovery schemes despite implementation overheads. Therefore, they'll have a smaller recovery penalty. Cheaper, more cost-sensitive commodity systems, such as workstations and desktop processors, will likely rely on more coarse-grained mechanisms. Then, a synergic scheduler can adapt dynamically to track platform-specific recovery costs.

The need for error resilient architectures is becoming paramount in the presence of variations. As technology scaling further diminishes transistor size, the likelihood of errors will increase. This article aims to understand the ramifications, specifically in the context of voltage noise. Our measurements on a Core 2 Duo processor show that voltage noise will be a dominant issue in the future. Designing processors for worst-case conditions will increasingly compromise performance and power efficiency. A suitable alternative is to build error recovery hardware that tolerates errors infrequently while maximizing performance during typical-case operation. Unfortunately, such a solution is only effective in the near term. As feature size diminishes, susceptibility to voltage noise increases and, as a result, voltage emergencies will become more frequent. Then, error recovery overheads in resilient architectures will begin to dominate and diminish the opportunities for potential runtime performance gains, especially in multicore systems in which cross-core interference is likely. Thus, long-term solutions will require abstracting circuit-level challenges to the higher layers. We advocate a software-layer thread scheduling solution to lighten the burden on recovery mechanisms. Scheduling collaborative threads that produce destructive interference across cores is one way to smooth voltage noise. We envision future work in the development of thread scheduling policies that strike a balance between power and performance, as well as recovery overheads. MICRO

References

1. N. James et al., ''Comparison of Split-Versus Connected-Core Supplies in the POWER6 Microprocessor,'' *Proc. 2007 IEEE Int'l Solid-State Circuits Conf.,* IEEE Press, 2007, doi:10.1109/ISSCC.2007.373412.
2. K.A. Bowman et al., ''Energy-Efficient and Metastability Immune Timing-Error Detection and Instruction Replay-Based Recovery Circuits for Dynamic Variation Tolerance,'' *Proc. 2008 IEEE Int'l Solid-State Circuits Conf.,* IEEE Press, 2008, doi:10.1109/ISSCC.2008.4523227.
3. V.J. Reddi et al., ''Voltage Smoothing: Characterizing and Mitigating Voltage Noise in Production Processors using Software-Guided Thread Scheduling,'' *Proc. 43rd Ann. IEEE/ACM Int'l Symp. Microarchitecture,* ACM Press, 2010.
4. L.A. Barroso, ''The Price of Performance: An Economic Case for Chip Multiprocessing,'' *Queue: Multiprocessors,* vol. 3, no. 7, 2005, pp. 48-53.
5. T.J. Slegel et al., ''IBM's S/390 G5 Microprocessor Design,'' *IEEE Micro,* vol. 19, no. 2, 1999, pp. 12-23.
6. A. Hisashige et al., ''A 1.3 GHz Fifth-Generation SPARC64 Microprocessor,'' *Proc. 40th Ann. Design Automation Conf.,* ACM Press, 2003, pp. 702-705.
7. N.J. Wang and S.J. Patel, ''ReStore: Symptom-Based Soft Error Detection in Microprocessors,'' *IEEE Trans. Dependable and Secure Computing,* vol. 3, no. 3, 2006, pp. 188-201.
8. S. Narayanasamy, G. Pokam, and B. Calder, ''BugNet: Continuously Recording Program Execution for Deterministic Replay Debugging,'' *Proc. 32nd Ann. Int'l Symp. Computer Architecture,* IEEE CS Press, 2005, pp. 284-295.

Vijay Janapa Reddi is a researcher in the Research and Advanced Development Laboratories at Advanced Micro Devices (AMD). His research interests are in the area

of computer systems, specifically in applying virtual machines, runtime compiler systems, and architecture to address processor power and reliability challenges via hardware-software codesign. He has a PhD in computer science from Harvard University.

Svilen Kanev is a bachelor's student of computer science at Harvard University. His research interests include power-performance modeling and architecture of small cores.

Wonyoung Kim is a PhD candidate in engineering sciences at Harvard University. His research interests are in power management of multicore processors, with specific interest in on-chip DC-DC converter design for fine-grain dynamic voltage and frequency scaling. He has a BS in electrical engineering from KAIST (Korea Advanced Institute of Science and Technology).

Simone Campanoni is a postdoc in computer science at Harvard University. His work focuses on the boundary between hardware and software, relying on dynamic compilation, runtime optimizations, and virtual execution environments for investigating opportunities on auto-parallelization. He has a PhD in computer science from Politecnico di Milano University.

Michael D. Smith is a John H. Finley, Jr. Professor of Engineering and Applied Sciences, as well as the Dean of the Faculty of Arts and Sciences, at Harvard University. His research interests include dynamic optimization, machine-specific and profile-driven compilation, high-performance computer architecture, and practical applications of security. He has a PhD in electrical engineering from Stanford University.

Gu-Yeon Wei is a Gordon McKay Professor of Electrical Engineering in the School of Engineering and Applied Sciences at Harvard University. His research interests include high-speed, low-power link design; mixed-signal circuits for communications; and power regulation circuitry and management. He has a PhD in electrical engineering from Stanford University.

David Brooks is a Gordon McKay Professor of Computer Science at Harvard University. His research interests include architectural and software approaches to address power, thermal, and reliability issues for embedded and high-performance computing systems. He has a PhD in electrical engineering from Princeton University.

Direct questions and comments to Vijay Janapa Reddi, 90 Central St., AMD, Boxborough, MA; vijay.reddi@amd.com.