

DeCoR: A Delayed Commit and Rollback Mechanism for Handling Inductive Noise in Processors

Meeta S. Gupta, Krishna K. Rangan, Michael D. Smith, Gu-Yeon Wei and David Brooks

School of Engineering and Applied Sciences, Harvard University, 33 Oxford St., Cambridge, MA 02138

{meeta,kkrangan,smith,guyeon,dbrooks}@eecs.harvard.edu

Abstract

Increases in peak current draw and reductions in the operating voltage of processors stress the importance of dealing with voltage fluctuations in processors. Noise-margin violations lead to undesired effects, like timing violations, which may result in incorrect execution of applications. Several recent architectural solutions for inductive noise have been proposed that, unfortunately, have a strong correlation to the underlying power-delivery package model and require a feedback loop that is largely constrained by the voltage/current sensor characteristics. The resulting solutions are not robust across a wide range of microprocessor designs and packaging technologies. This paper proposes a Delayed-Commit and Rollback scheme (DeCoR) that guarantees correctness, insensitive to the package model or the responsiveness of the voltage sensors. In particular, our approach recovers from, rather than attempting to avoid, voltage emergencies. This approach incurs a small performance penalty when compared to an ideal machine that does not have voltage emergencies. We show that explicit checkpoint-recovery schemes, intended to handle infrequent events, e.g., radiation-induced soft errors, suffer from large performance overheads for frequently-occurring voltage emergencies. DeCoR requires very few modifications to modern processor designs, as it leverages the existing store queue and reorder buffers. Unlike conventional designs that conservatively protect all components of the processor from inductive noise with overly-large timing margins, our approach only requires conservative protection of the architected register state and cache write paths.

1. Introduction

Inductive noise has been a long-standing problem in processor design. With greater interest in reduced-power microarchitectures, this problem has gained even more significance, especially as operating voltages decrease while peak currents increase in the presence of technology scaling [24]. Large current swings over small time scales cause large voltage swings in the power-delivery subsystem due to parasitic inductance. One can view such *voltage emergencies* as conditions when the supply voltage significantly deviates from the nominal operating voltage, resulting in transient timing faults and long-term reliability problems. The maximum allowable magnitude of this deviation is referred to as the *hard threshold*. Traditional designs require timing margins throughout the processor that are large to prevent timing faults. Supply-noise analysis of the POWER6 processor [14] shows voltage droop induced delay degradation as large as 17% (corresponding to a voltage droop of 200mV for a 1.1V supply) can occur. To avoid large timing margins required by supply noise, processor designers must find solutions to the inductive noise problem that are cost effective, guarantee correctness, and minimally impact system-level performance.

Several researchers and microprocessor vendors have investigated architecture-level solutions to deal with voltage emergencies [11, 15, 20, 21]. These prior works strive to detect and avoid impending voltage emergencies arising from inductive noise to prevent failures. These impending voltage emergencies are either detected using voltage sensors [15] or current sensors [20]. Joseph et al. propose a voltage sensor based approach, where a throttling mechanism is invoked when the sensed supply voltage crosses a specified level, called the *soft threshold* [15]. Such a throttling mechanism must react before the voltage deviation proceeds beyond the soft threshold to the hard threshold. The choice of the soft threshold level is largely governed by the voltage sensor response time and accuracy. Our experiments (detailed in Section 6.2) show that throttling mechanisms cannot guarantee robustness for some benchmarks even with an ideal sensor delay of zero cycles. Moreover, the high dependence of throttling mechanisms on underlying assumptions associated with the power-delivery subsystem and sensors means that these schemes cannot easily be re-targeted for different processor architectures or power-delivery subsystems. In addition, the conservative threshold settings used to offset the sensor delay and allow the response mechanism to engage in these schemes cause unnecessary throttling which degrades performance. These drawbacks motivate the need for a solution that can be applied robustly to a wide range of designs with minimal hardware cost and low performance impact.

In this paper, we propose a *delayed-commit* and *rollback* mechanism (DeCoR) to handle voltage emergencies. Rather than trying to prevent voltage emergencies, this mechanism allows noise margin violations to occur, but when they do, the architecture has the ability to *rollback* to a guaranteed *correct* processor state. This approach relaxes constraints on the power-delivery subsystem and sensor implementation. We divide our processors into two zones: *roll-back protected (RB-protected)* and *timing-margin protected (TM-protected)* zones. A *RB-protected* zone includes all structures where DeCoR permits recovery from voltage-induced timing violations. In particular, we RB-protect the vast majority of the processor core, along with the read path of the L1 data cache. A *TM-protected* zone encompasses all other structures that require timing margins large enough to prevent voltage fluctuations from corrupting execution. The write path of the L1 cache, the entire L2 cache, and the retirement register file (RRF) are TM-protected.

The delayed-commit mechanism speculatively buffers processor updates to the machine state (register file and memory) until it has verified that no noise-margin violations have occurred during a time period sufficient for the sensors to detect potential noise-margin violations. At the end of this sliding window of time, the state is said to be *noise-verified* and can be committed to its respective structure. In the event of noise margin failure, *noise-speculative* updates are discarded and execution is restarted from a prior noise-

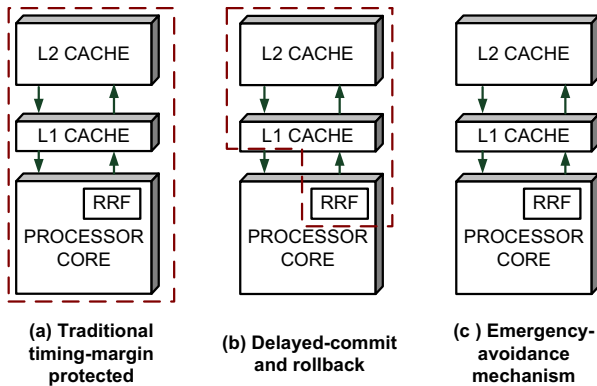


Figure 1: Three solutions to handle voltage emergencies with different tradeoffs between robustness and cost of design. (a) represents a robust, but costly solution. (c) provides a low-cost, but less robust solution.

verified state. Thus, the delayed-commit mechanism distinguishes the processor’s *noise-speculative* state from its *noise-verified* state. While the proposed delayed-commit and rollback mechanism may appear to resemble traditional checkpoint-recovery schemes, there are distinct differences in implementation requirements/challenges and resulting performance penalties (discussed in detail in Section 5.3). Moreover, DeCoR has a conservative overhead of 7% compared to the 39% overhead of explicit checkpointing schemes.

Figure 1 illustrates three schemes to handle inductive noise, discussed so far. In standard design flows (Figure 1(a)), the circuit designer is responsible for meeting timing margins under worst-case inductive noise conditions. This leads to a robust, but over-designed system for typical usage. In the emergency-avoidance throttling scheme (Figure 1(c)), architectural techniques are used to protect all state information while the timing margins are kept lean. Such throttling schemes yield a lower cost design, but fail to guarantee correct operation across different package solutions or designs with large sensor delays. We also show that aggressive throttling can suffer from its own timing problems or incur unacceptably large numbers of false alarms leading to performance degradation when compared to an ideal noiseless machine. In the proposed scheme (Figure 1(b)), we retain the robustness of conservative designs while gaining the benefits of throttling schemes. In other words, DeCoR strikes a balance between traditional designs that are completely TM-protected and previously proposed throttling solutions that attempt to entirely eliminate conservative timing margins.

The main contributions of this paper are:

1. We provide a detailed analysis of a wide range of power-delivery subsystems for modern processors and show how different characteristics affect the occurrence of voltage emergencies. This analysis highlights the need for a widely usable robust solution to deal with voltage emergencies.
2. We propose a delayed-commit and rollback mechanism to handle voltage emergencies and can accommodate various of voltage/current sensor and power-delivery subsystem options. This approach is robust—guaranteeing correct operation across a wide range of package designs and sensor delays.
3. We explore various design parameters and present an experimental evaluation of benefits and costs associated with our framework. DeCoR can be low cost by leveraging existing buffering and reusing the flush mechanisms found in typical microarchitectures, and performance impact is substantially lower than that of employing explicit checkpoint mechanisms.

The rest of the paper is organized as follows. Section 2 highlights two different types of transient errors and emphasizes the need for an explicit solution to deal with noise-margin violations. Section 3 provides a brief description of power-delivery subsystems for processors and shows the effects of differing characteristics on voltage emergencies. Then, the proposed delayed-commit and rollback mechanism is presented in detail in Section 4. The approach is evaluated in Section 5 and a comparison to explicit checkpoint schemes highlight the differences between the two schemes. Section 6 presents an overview of existing throttling mechanisms for emergency avoidance, and gives examples where throttling fails to guarantee correct application behavior and showcases the robustness of our proposed scheme for a wide range of package solutions. Section 7 reviews prior research related to inductive noise and checkpoint mechanisms. Section 8 concludes the paper.

2. Background on Transient Errors

With continued advances in technology, the issue of reliability for modern high-performance processors is gaining importance. Reliability can be affected by transient errors, which can either be radiation-induced soft errors or voltage-induced noise-margin violations. Soft errors and noise-margin violations are similar in that they can cause transient failures, but they differ greatly in their characteristics. The main differences are in 1) the physical phenomenon that causes them; 2) the frequency of error occurrence; 3) the structures sensitive to the errors; and 4) the relationship between application characteristics and error occurrence. In this section, we explore these main differences and highlight the differing requirements for detection and correction.

Soft errors are generally characterized as single-event upsets or bit-flips, caused by energetic particles from cosmic rays or alpha particles. The occurrence of soft errors is quite rare and these errors primarily affect data storage nodes. The probability of single-event upsets affecting the correctness of computation depends on the architectural vulnerability factor of the logic [18], which determines whether a fault in that logic would actually affect the outcome of the application. A common strategy is to employ a *reactive mechanism* where appropriate recovery actions are taken once an error has been detected. One of the main challenges to dealing with soft errors is the implementation of a robust error-detection mechanism, often seen in the form of parity bits and/or error correcting codes (ECC). For example, a parity-bit propagation technique to detect soft errors was implemented by Fujitsu in their SPARC processors, providing coverage for almost 80% of latch banks and array structures [2]. Other detection approaches utilize redundant (or checker) processors and threads, which re-execute some or all of the instructions to verify correctness [4, 22, 28]. The infrequent occurrence of soft errors allows these reactive mechanisms to have large penalties associated with recovery. For example, the Fujitsu processor employs a checkpoint hardware mechanism with a quiescent and preparation period for restart of around $1\mu\text{s}$. This is because microsecond scale penalties are acceptable for soft errors that occur at the timescale of days.

Noise-margin violations have very different characteristics from those observed for soft errors. Noise-margin violations are sensitive to the characteristics of the underlying power-delivery subsystem and the application. Inductive noise results from parasitics present in the system, which can cause the voltage to swing significantly in response to current fluctuations. If the voltage swings are significant, they can induce timing-margin violations. Noise-margin violations primarily affect logic delay paths and are tightly cou-

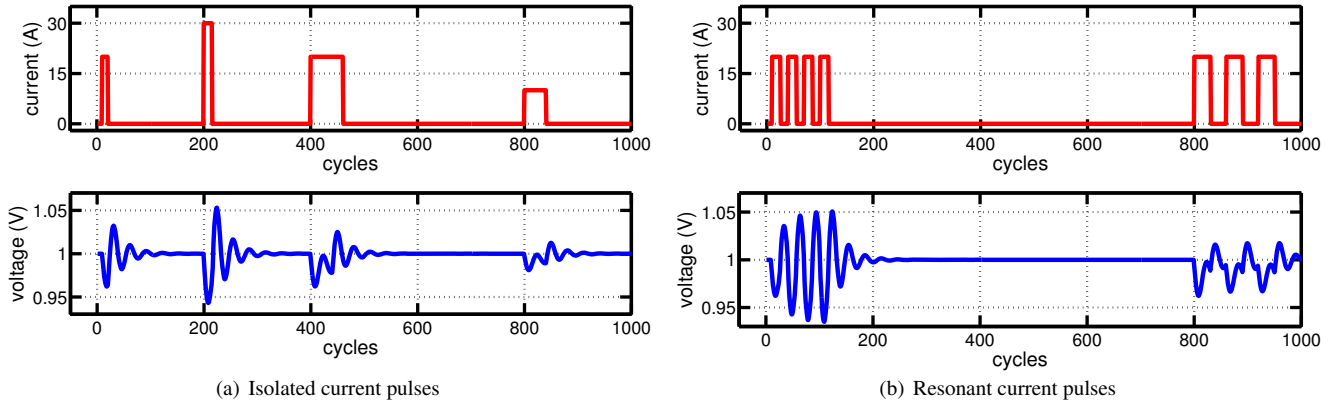


Figure 2: Isolated current pulses and resonating pulses with a period of 100MHz.

pled to application characteristics. For example, the presence of repeated execution patterns in applications can increase the susceptibility to timing violations due to resonance in the power-delivery network [12]. Unlike soft errors, noise-margin violations are easier to detect, e.g., using hardware sensors, but require careful balance between latency and resolution. Several researchers have proposed to address noise-margin violations through feedback-driven *avoidance mechanisms* [11, 15, 20, 21], where these sensors predict impending violations and take preemptive actions, such as throttling. Traditional designs require overly conservative timing margins, preventing the occurrence of such violations. In contrast, designs can more aggressively reduce timing margins if the aforementioned avoidance mechanisms are in place. Consequently, frequent and repetitive occurrence of voltage noise requires fast detection and response to ensure correctness and limit performance loss.

In contrast to the above avoidance techniques, this paper presents a simple yet robust reactive approach to deal with noise-margin violations. The approach is based on a delayed-commit mechanism that requires very little modification to conventional processor designs. The high occurrence (due to repeated execution patterns) of voltage-induced errors means that the implementation and performance issues related to delayed-commit and rollback are paramount. Reactive mechanisms that have been proposed to handle soft errors cannot be used to address noise-margin violations due to the large disparity in the temporal characteristics of the two types of transient errors. Radiation-induced soft errors are infrequent and, hence, can use recovery schemes [2, 29] that implement checkpointing with coarse granularity on order of hundreds to thousands of cycles. While sufficient for soft errors, this granularity is too coarse for much more frequent voltage emergencies, incurring overly high overheads and leading to unacceptably large performance penalties. DeCoR provides a fine-grained reactive approach for this very different problem domain—addressing voltage noise-margin violations—with minimal impact on performance.

3. The Power-Delivery Subsystem and Voltage Emergencies

Noise-margin violations, or voltage emergencies, are closely linked to the detailed characteristics of the power-delivery subsystem (PDS). In this section, we investigate the interaction between current consumption profiles and the PDS that leads to voltage emergencies. We consider voltage swings greater than $\pm 5\%$ of the nominal voltage to be voltage emergencies. The $\pm 5\%$ margins, assumed in previous studies, allow designers to choose relatively aggressive

margins as opposed to 15-20% [14]. We also show how different package parameters can affect voltage variations for a given processor. It is important to understand this interaction to design robust architectural solutions that can handle noise-margin violations.

3.1 Characteristics of current pulses

While the PDS of a given microprocessor is a complex system consisting of several different components (e.g., voltage regulator module, package, on-die capacitors, etc.) [5, 8], a simplified second-order lumped model [13, 26] can adequately capture its resonance characteristics with impedance peaking in the mid-frequency range of 50MHz-200MHz. Ideally, the supply voltage across a processor should be constant. However, due to dynamic current fluctuations and the non-zero impedance of the PDS, large voltage fluctuations can occur. One way to characterize voltage variations is by convolving the instantaneous current profile of the microprocessor with the impulse response of the PDS (Equation 1).

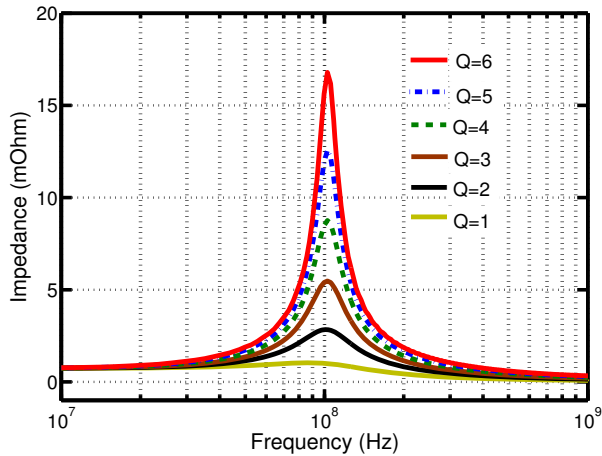
$$v(t) = i(t) * h(t) \quad (1)$$

Sudden short spikes in current can cause voltage variations, but the magnitude of the variation largely depends on the amount of charge build up over a specific time interval. Figure 2(a) shows the voltage transients for current pulses of varying amplitudes and durations. The first and second pulses have the same width, but the second pulse has a higher amplitude. A sufficiently high amplitude can induce violations ($\pm 5\%$). The last two pulses shown, though they have large integrated charge, do not cause significant variations in voltage. This shows that isolated pulses with a certain amplitude/width combination can lead to voltage emergencies.

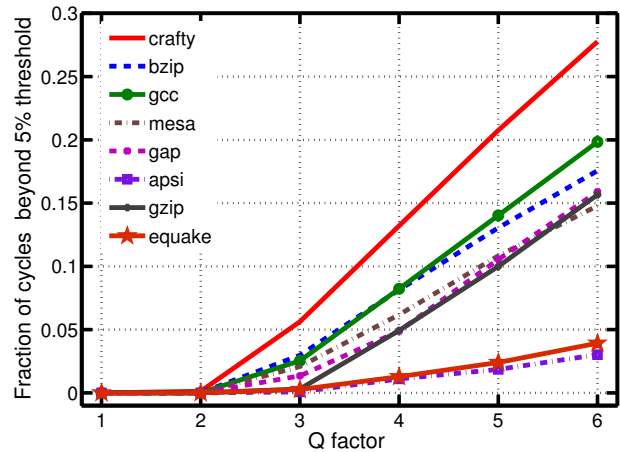
Even if the voltage fluctuation caused by a current pulse in isolation does not exceed noise margins, a series of such pulses at the resonance frequency of the PDS may lead to a voltage emergency. Figure 2(b) shows the voltage response for a series of current pulses. The first sequence of current pulses has a period of 30 cycles, which corresponds to a frequency of 100 MHz for a 3GHz processor. If the resonance of the PDS also occurs at 100 MHz, voltage swings gradually build up and exceed emergency thresholds. Thus, it is important to consider both isolated pulses and resonating pulses when designing an architectural solution to handle voltage emergencies.

3.2 Analysis of sensitivity to power-delivery subsystem

Given that a power-delivery subsystem can be modeled as a second-order linear system, the response of the package model to current variations is largely governed by three factors: Q (quality factor),

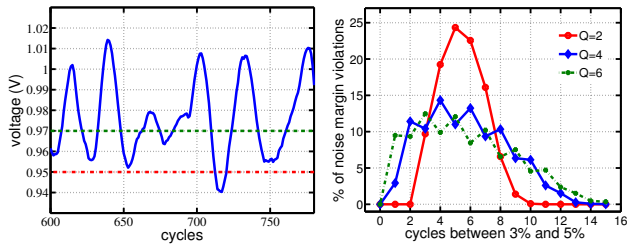


(a) Package impedance versus Q



(b) Fraction of cycles crossing 5% threshold versus Q factor

Figure 3: Sensitivities to Q show packages with higher Q incur more noise-margin violations.



(a) Snapshot of *crafty* (Q = 2)

(b) Distribution of noise-margin violations

Figure 4: Snapshot and distribution of voltage emergencies in *crafty*. (a) Example of voltage transitions between 3% and 5% thresholds. (b) Distribution of transition times (cycles) for all all noise-margin violations.

C (resonance cycles) and Z (peak impedance). These factors affect the robustness and correctness of any solution for handling noise-margin violations. In this section, we analyze the effects of these three factors on voltage emergencies.

Quality factor (Q): The quality factor of a system is the ratio of the resonant frequency to the rate at which it dissipates its energy. This factor determines the width of the resonance, or the *resonance band* of the system. A higher Q leads to a greater build up of voltage for currents oscillating within the resonance band. Q depends on the effective inductance (L) and resistive losses (R) at the resonant frequency ($f = \frac{1}{2 * \pi * \sqrt{LC}}$) as shown in the following equation:

$$Q = \frac{2 * \pi * f * L}{R} \quad (2)$$

A good package will have lower parasitic inductance and hence lower Q than a poor package. Figure 3(a) shows different packages with different Q, where it can be seen that higher-Q packages have a narrower resonance band and higher impedance at the resonant frequency. Higher impedance means that applications with current variations within the resonance band experience larger voltage fluctuations. Figure 3(b) illustrates how different packages with different Q factors affect voltage emergencies on a subset of the SPEC CPU2000 benchmarks. As Q increases, the fraction of cycles where the voltage extends beyond $\pm 5\%$ thresholds increases for all benchmarks. However, the slope for each benchmark differs, with *crafty* experiencing the steepest increase in noise-margin violations. This can be attributed to the differing current profiles of the benchmarks.

The PDS Q factor defines an important constraint on any technique designed to handle voltage emergencies. Specifically, the rate of change of voltage will depend on Q. For example, a snapshot of *crafty*'s voltage trace is depicted in Figure 4(a). This example assumes two thresholds, one at $\pm 3\%$ (soft) and the other at $\pm 5\%$ (hard). The short time interval over which the voltage crosses both soft and hard thresholds determines the maximum delay any soft-threshold based avoidance mechanism can tolerate. Figure 4(b) plots the percentage of voltage emergencies occurring for different delays between threshold crossings over a range of Qs across the entire voltage trace of *crafty*. This plot shows that even with moderately low Qs voltage fluctuations can be very fast.

Resonance Cycles (C): This factor represents the number of processor cycles corresponding to one period of the PDS resonant frequency. As processor frequency increases while the PDS resonant frequency remains fixed, C also increases. For example, a resonant frequency of 100MHz for a 10GHz processor would result in a C of 100 [20], whereas C would be 30 for a 3GHz processor.

Voltage emergencies strongly depend on this resonance cycles metric. Figure 5(a) plots the resulting voltage fluctuations for three settings of C and shows that the minimum width of an emergency-inducing isolated current pulse differs for different resonance cycles. In fact, this width depends on the resonant frequency of the PDS such that a larger C tends to require wider current pulse widths, in terms of the number of processor cycles. Figure 5(b) shows how the fraction of cycles with noise-margin violations varies with processor frequency for a given package. The package considered here has a resonant frequency of 100MHz. We can see that the peaks for the different benchmark cycles shown here are at lower values of the resonance cycles metric (20-30 cycles). This is because, in these benchmarks, current pulses tend to have smaller widths—both for resonating and isolated pulses.

Peak Impedance(Z): This factor represents the peak impedance of the power-delivery subsystem at its resonant frequency. Ideally, this peak (or target) impedance should be as low as possible to avoid voltage emergencies. However, efforts to reduce this peak impedance can increase system cost. Therefore, circuit and architecture designs must cope with higher than desired impedance to avoid voltage emergencies. Figure 5(b) shows that as the peak impedance of the package increases, noise-margin violations also increase across all benchmarks and resonance cycles.

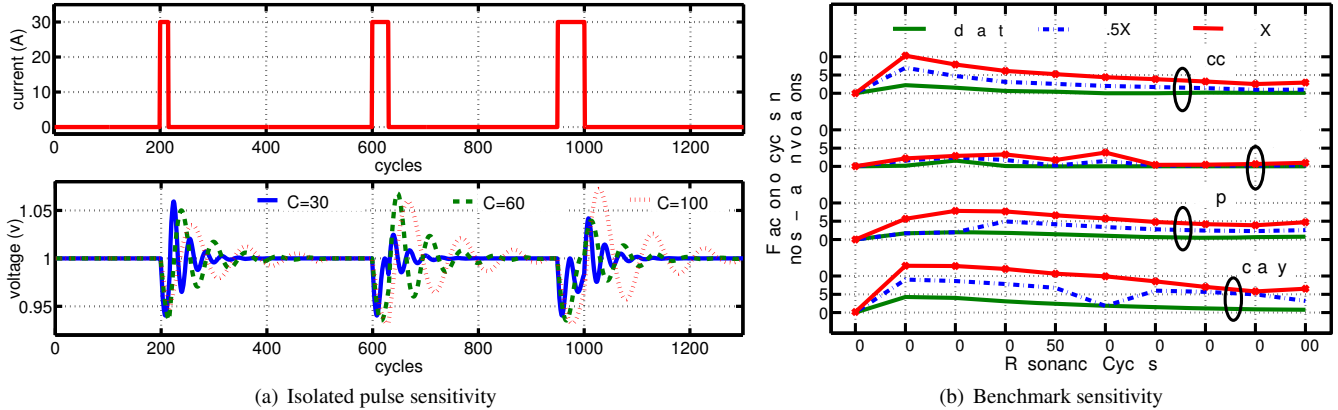


Figure 5: Sensitivity to resonance cycles (C) and peak impedance (Z).

The frequency of noise-margin violations varies across different power-delivery subsystem designs, and is also closely related to the current consumption profiles. It is important to guarantee that the mechanism used for handling voltage emergencies is robust across a wide range of package and processor characteristics.

4. Delayed-Commit and Rollback

The previous section has shown that a solution for handling noise-margin violations needs to be robust for different packages and sensors characteristics. Along these lines, we propose a technique to deal with inductive noise that does not require fast sensors and actuation mechanisms, which are difficult to implement in practice. Moreover, if sensor thresholds can be set close to or at the hard threshold, we can reduce the false alarm rate (explained in Section 6) to zero and avoid unnecessary performance loss. This section describes an approach with exactly these characteristics. The proposed solution provides the robustness of existing traditional designs, while avoiding the need for large timing margins in the core blocks of the processor. This approach requires very little new hardware and, as we will demonstrate in Section 5, incurs small performance penalties over ideal designs that do not suffer from voltage emergencies. We approach the problem by dividing the processor into two zones as described in Section 4.1. Section 4.2 presents details of the new technique that uses a delayed-commit mechanism. This mechanism guarantees that only states verified to be correct are committed, even in the presence of voltage fluctuations. Section 4.3 then describes the proposed recovery mechanism that ensures a correct execution whenever a noise-margin violation occurs.

4.1 RB-protected and TM-protected zones

The proposed delayed-commit and rollback (DeCoR) mechanism does not attempt to avoid voltage emergencies. Instead, it lets the processor core run freely with aggressively set timing margins, and provides safeguards to detect and recover from potential noise-margin violations if and when voltage emergencies do occur. We divide the overall machine architecture into two zones: a *timing-margin protected (TM-protected)* zone, as is traditionally done, and a *rollback protected (RB-protected)* zone.

The TM-protected zone relies on standard circuit-based techniques to guarantee all timing margins are met. Although circuitry in this zone requires more conservative designs, we limit blocks that reside here to the retirement register file, the PC chain, the L1 write port, and the L2 cache. Fortunately, these structures tend to be less timing critical for several reasons. First, processor per-

formance is relatively insensitive to L2 cache latency. For example, many designs will construct L2 caches using low-leakage, high-Vt transistors that trade access latency for reduced power consumption [10]. For the L1 cache, the L1 *read* path is known to set the access time, while the write path is less critical [31]. This is because the read ports are driven by small SRAM cells that cannot easily be sized up, while the write ports are driven by external peripheral circuits that can be appropriately sized to increase speed. We assume idle memory cells are resilient to common-mode voltage fluctuations, which affect both sides of a differential SRAM cell equally. Recent cache designs from Intel [23] demonstrate that memory cells retain state in lower-voltage drowsy modes, where the voltages are much lower than the low-end voltage-emergency level we assume. Furthermore, we assume idle memory cells can have additional protection by using standard ECC measures common in today’s microprocessors. Finally, the retirement register file and PC chain are relatively small structures, are less likely to be timing critical, and can be sized up with small power penalties, if needed.

The rest of the processor pipeline resides in the RB-protected zone, which includes the instruction fetch unit, instruction cache (there are no writes to the I-cache from the processor and the static memory cells are robust as explained above), the issue logic, the execution units, and the commit logic that consists of the reorder buffer and store queues. These structures can assume more aggressive timing margins to avoid unnecessary performance loss, since they rely on an architectural mechanism for protection. We note that updates to the branch predictor in the noise speculative state may corrupt the predictor state if a rollback is initiated. However, the frequency of these corruptions would be small and would not impact correctness or performance. Slicing the processor into TM-protected and RB-protected zones is also straightforward, and can simply be applied at the architectural block level (in RTL). The paths from these blocks can be flagged to have extra timing margins. This approach is more efficient than arbitrarily adding margins to all paths, and is no more complex than identifying critical paths in schemes like Razor [9].

To deal with potential timing violations in the RB-protected zone, we propose a delayed-commit and rollback mechanism that guarantees correctness in the presence of voltage emergencies. Our approach enables the processor to recover from voltage emergencies by rolling back the system state if voltage emergencies are detected by voltage sensors. The delayed-commit mechanism ensures that the architected state and the values in the L1 data cache are not corrupted by timing violations. Unlike emergency avoidance,

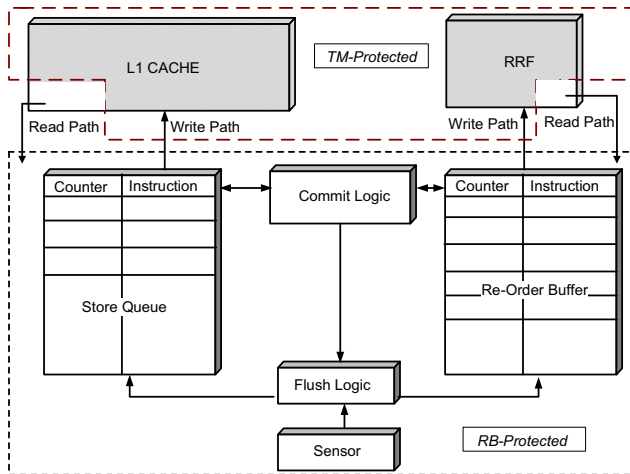


Figure 6: Block diagram of delayed-commit architecture.

this mechanism does not seek to detect and mitigate emergencies, but reacts to them after they occur. Hence, the response mechanism is engaged only for true voltage emergencies (i.e., violations of the $\pm 5\%$ hard threshold) and does not need to detect emergencies using soft thresholds. More importantly, our approach *guarantees* correctness of the system across wide ranges of power-delivery network and processor characteristics and sensor-delay latencies.

4.2 Delayed commit

Figure 6 presents a functional diagram of the delayed-commit architecture. To guarantee system correctness, this architecture distinguishes between the *noise-verified* state and *noise-speculative* state. A noise-verified state is the state of the machine that is *known* to be correct. The program can be rolled back to the noise-verified state following a voltage emergency, which will be signaled by a voltage-sensor reading. In the noise-speculative state, the processor continues executing down the regular execution path and results are held in existing buffering mechanisms (ROB and STQ) until the outcome of the sensor reading is known. This buffering breaks the traditional feedback loop necessary in sense and throttle schemes (Section 6), accommodating any length of sensor delays.

During program execution, the processor buffers values corresponding to the noise-speculative state in the reorder buffer (ROB) and store queue (STQ). To know when this state can become noise-verified, each entry in the ROB/STQ has a counter associated with it. Completed results need to be buffered in the ROB/STQ until they are verified to be correct with respect to noise violations. The time the instructions need to be buffered is directly proportional to the sensor delay in the system. If we assume a sensor delay of D cycles, all completing instructions will set their counters for this delay. When the instruction reaches the head of the queue and is ready to retire, the commit logic verifies that the counter has expired and then declares the state to be *noise verified*. At this point, it is safe to commit the state to the appropriate TM-protected structure, i.e., RRF or L1 data cache.

The correctness of our scheme centers on proper transitions from the noise-speculative state to the noise-verified state. For our current implementation, this transition takes place when committing state from the STQ to the L1 data cache and from the ROB to the RRF. We must guarantee the robustness of writes to the RRF and data caches at all times, because in a worst-case scenario, a voltage emergency could occur when moving state into these noise-verified locations. For this reason, the write paths of these structures

must have sufficient timing margins to tolerate voltage noise and hence lie in the TM-protected zone. In contrast, reads from the data cache and the retirement register file can experience voltage emergencies, because these emergencies would be detected and handled by the delayed-commit mechanism. In other words, we do not need to take any special measures when data transitions into the RB-protected zone. As discussed earlier, we assume that idle memory cells in the ROB/STQ will not be corrupted during the few cycles between a state becoming noise-verified and committed.

4.3 Rollback

When a voltage emergency occurs, the sensors notify the machine that all noise-speculative states should be flushed, and that it should initiate a rollback to the last noise-verified correct state. Flushing is straightforward, as the noise-speculative state is already located in structures (the ROB and STQ) that are capable of flushing speculative states and rolling back program execution. Thus, a flush in our rollback mechanism is similar to a flush after branch mispredicts, and the machine can be restarted the next cycle. A key attribute of our scheme is that rollbacks occur only when noise margins are actually violated; false alarms never occur.

To ensure that rolling back the processor does not cause new emergencies, we start the processor at a reduced frequency for some number of cycles, called the throttling period. This guarantees forward progress at the cost of a small loss in performance. Our experiments show that a throttling period of 10 cycles at a 50% frequency following the rollback action ensures forward progress. This half-rate throttling can be achieved without PLL involvement, that is, by simply gating the clock every other cycle.

5. Performance Analysis

This section investigates the performance impact of different parameters used in the proposed DeCoR mechanism and seeks to understand how changes in sensor delay, buffer size, throttling period, and throttling factor affect performance. We refer to sensor delay as the total time comprised of making measurements on one or multiple distributed sensors, communicating the results to the actuator, and taking appropriate actions. Section 5.3 provides a performance comparison of our proposed schemes with previous explicit checkpoint schemes, and shows the high restore penalties associated with previous schemes.

5.1 Power-delivery and simulation frameworks

Though we do provide detailed results considering four different package configurations in Section 6.2, the following analysis will focus mainly on a single package model based on the characteristics of the Pentium IV package [5]. In this model, the resonant frequency of the PDS occurs at 100MHz with a peak impedance of $5m\Omega$. This corresponds to a resonance cycle (C) of 30 cycles for a 3GHz machine. This model's impedance response is illustrated in Figure 3(a) ($Q=3$). Finally, we assume peak current swings of 16-50A, and noise-margin violations occur at $\pm 5\%$ of a 1V supply. Earlier works [11, 15, 20] used parameters based on the Alpha 21264/21364 package, which we also evaluate in Section 6.2.

Our architectural simulations are based on a version of SimpleScalar for the *x86* architecture. Table 1 tabulates the parameters used to configure an 8-way superscalar, out-of-order processor. To get a detailed cycle-accurate current profile, we incorporate a modified version of Wattch [6] into our SimpleScalar simulator. Voltage variations are calculated by convolving the simulated current profiles with an impulse response of the PDS, as detailed previously in

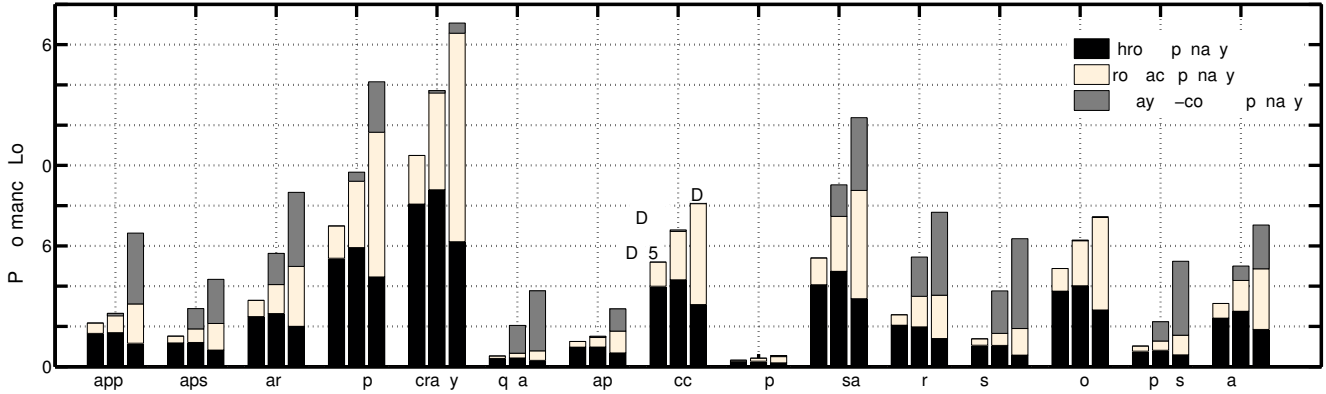


Figure 7: Performance penalty of delay-commit and rollback relative to an ideal machine for three sensor delays (D).

Clock Rate	3.0 GHz	RAS	64 Entries
Inst. Window	128-ROB, 64-LSQ	Branch Penalty	10 cycles
Functional Units	8 Int ALU, 4 FP ALU, 2 Int Mul/Div, 2 FP Mul/Div	Branch Predictor	64-KB bimodal gshare/chooser
Fetch Width	8 Instructions	BTB	1K Entries
L1 D-Cache	64 KB 2-way	Decode Width	8 Instructions
L2 I/D-Cache	2MB 4-way, 16 cycle latency	L1 I-Cache	64 KB 2-way
		Main Memory	300 cycle latency

Table 1: Processor parameters for SimpleScalar.

Equation 1. This second-order model is used in several other studies [15, 20].

We ran a subset of SPEC2000 benchmarks, as certain benchmarks failed to complete successfully on our simulator. Each benchmark was run for 100 million instructions of a representative phase given by Simpoint. Large current variations can lead to significant voltage drops, and these variations are generally governed by the activity of the benchmark application just before the voltage drop. Our experiments show some correlation between the distribution of voltage and the IPC¹ of the application. Applications with high IPC (i.e., *crafty* and *mesa*) exhibit high levels of processor activity and, hence, have more current variations around the resonant frequency of the system. Benchmarks with lower IPC (e.g., *equake* and *apsi*) have longer stall/idle periods (due to high cache miss rates) than the previously mentioned benchmarks and, thus, have a smaller fraction of cycles within the $\pm 5\%$ threshold.

5.2 Performance Analysis of DeCoR

There are two primary sources of performance loss associated with the delayed-commit scheme: 1) Delaying the commit of instructions in the reorder buffer and store queue may lead to buffer pressure and potential stalls and 2) Each rollback and subsequent recovery incurs a runtime penalty. The performance impact of delayed commits is, therefore, a function of the sensor delay, the characteristics of the power-delivery system, and size of the store queue and reorder buffer. Rollback overhead can be further divided into cycles spent in *replay/rollback* and cycles spent during *throttle* (the slow startup mode engaged during recovery). Equation 3 provides a breakdown of the penalty associated with rollback:

$$\text{ExtraWork}_{DCR} = RA * (RC + (TF - 1) * TP) \quad (3)$$

¹As we are using an x86 microprocessor model, IPC refers to the number of micro-operations executed per cycle (μIPC).

where *RA* represents the number of times the rollback happens (or *rollback alarms*), *RC* represents the number of rollback cycles, *TP* represents the throttle period, and *TF* represents the throttling factor (fraction of the clock frequency applied during throttle).

Figure 7 presents the total performance penalty due to the DeCoR mechanism for different sensor delays across several benchmarks (*TF* = 2 and *TP* = 10 cycles). For sensor delays as low as 5 cycles, the performance impact ranges from 0.02% to 7% across the benchmark suite. This includes the total performance penalty due to both STQ and ROB pressure during delayed commit and rollback and the slow startup of the processor during rollbacks.

A breakdown of the contribution of each factor (buffer pressure, rollback, and throttle) is also shown in Figure 7 for different sensor delays. We observe that the performance loss in benchmarks with high IPC (e.g., *bzip* and *crafty*) is largely dominated by rollback and throttle penalties. There are two reasons for this. First, benchmarks with higher IPCs generally have more voltage emergencies. Second, high IPC benchmarks tend to have fewer memory stalls, resulting in lower occupancy rates for the ROB and STQ. Consequently, the penalty from buffer pressure is small. The performance loss of benchmarks with low IPC is dominated by buffer pressure, but the performance loss in these benchmarks is also generally small. For larger sensor delays, the buffer pressure penalty and rollback penalty tend to dominate, and the throttle penalty decreases. Another way to understand how the delayed-commit scheme applies pressure to the store queue and ROB is to look at occupancy. Our simulations show that benchmarks with low IPC, like *equake* and *apsi*, tend to have near full ROB/STQ structures and, hence, experience higher performance penalty due to buffer pressure. The occupancy of these structures increases by only a few entries for almost all of the benchmarks, indicating that a slightly larger STQ/ROB can alleviate performance loss.

5.3 Comparison with existing explicit checkpoint-recovery schemes

The proposed delayed-commit and rollback mechanism differs greatly from traditional checkpoint-recovery mechanisms in terms of the mechanisms and overheads involved. Previous checkpoint-recovery schemes have an explicit checkpoint mechanism, whereas our approach relies on a light-weight implicit checkpointing scheme. In this section, we highlight the key differences between the two approaches and compare the performance penalties of not using a delayed-commit mechanism for handling inductive noise.

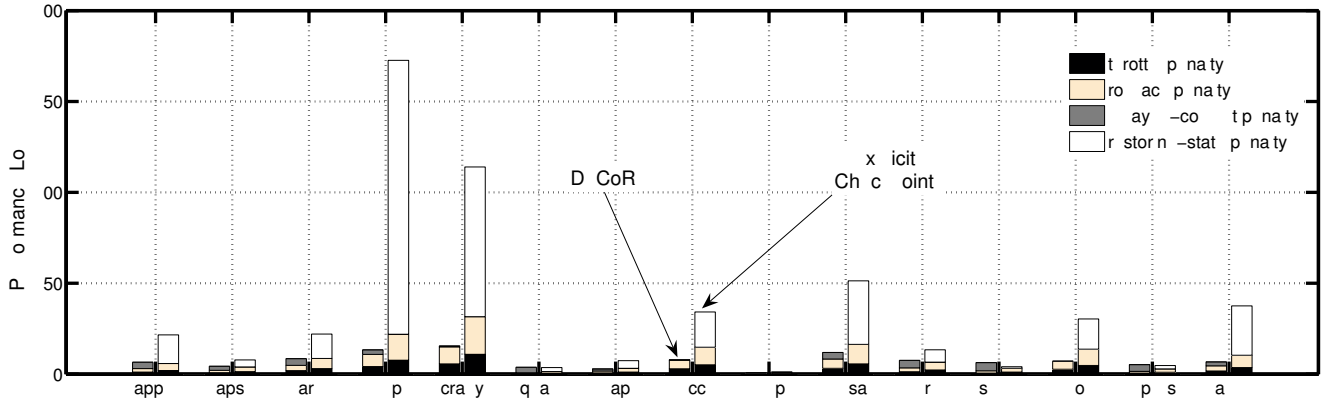


Figure 9: Side by side comparison of penalty breakdowns versus benchmarks for the delayed-commit and rollback scheme (left bar) and the explicit checkpoint scheme (right bar). This comparison assumes 20 cycle sensor delays and a 21 cycle checkpoint intervals for the explicit checkpoint scheme.

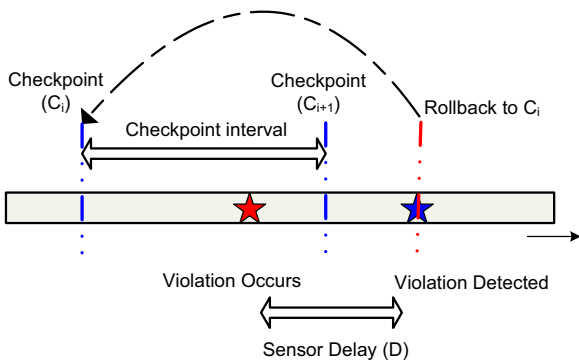


Figure 8: Example illustrating the two checkpoints required in explicit checkpoint schemes to maintain correctness given noise-margin violations.

Explicit checkpoint-recovery mechanisms have been employed to tackle problems in a variety of domains. There can be several mechanisms for explicit checkpoints that vary in their degree of implementation complexity and overheads involved. Earlier checkpointing schemes were predominantly aimed at providing fault tolerance in large systems. Recently, however, checkpointing schemes have been proposed for several other domains: soft error detection [29], boosting processor performance [16, 17, 27], fault detection [25] and debugging [19].

The general aim of recently proposed hardware checkpoint-restart mechanisms has been to either provide recovery from soft errors or to provide improved performance due to larger speculation opportunities. All these schemes aim to take checkpoints at very coarse granularities, ranging from 100 to 1000 cycles. The high frequency of noise-margin violations (as compared to soft errors or rollbacks due to exceptions in highly speculative architectures) necessitates the need to have a low-overhead and very fine-grained mechanism. Moreover, a robust scheme to handle noise-margin violations needs to be invariant to sensor delays. Figure 8 shows an example of a scenario where the violation is detected after checkpoint C_{i+1} has been taken. To have correct semantics, the system should rollback to checkpoint C_i , incurring a higher rollback cost. In general, even if the checkpoint interval is adjusted to match the sensor delay in the previously proposed mechanisms, the recovery costs of discrete, explicit mechanisms are higher than the implicit checkpointing mechanism in DeCoR.

A common trait to all checkpoint-recovery schemes is the explicit saving and restoring of required architectural state. The states that must be saved for correct execution at recovery is mainly the architected state, which consists of the registers and the updated memory-state. For example, [16] and [17] assume a buffered memory update, using volatile bits for updated lines between checkpoints. When a rollback occurs, the lines marked volatile are flushed from the cache. However, this results in additional cache misses after rollback. The overhead of implementations proposed in [1] and [17] includes a register restore latency of 8 cycles (for 32 registers with 4 write ports); the infrequency of the rollbacks in other implementations shadows the rollback cost. However, such schemes will require frequent rollbacks and incur large performance penalties when applied to noise-margin violations.

Figure 9 shows a breakdown of the performance impact of rollback, throttle, and restoring of state for explicit checkpointing compared to the performance impact of rollback, throttle, and delayed-commit for DeCoR with a sensor delay of 20 cycles. Due to the rollback to the previous checkpoint, C_i , instead of current checkpoint, C_{i+1} , for every rollback, the rollback penalties (including the throttle penalties) are higher for the explicit checkpoint schemes. We can also see that the performance impact of restoring state in explicit checkpoint mechanisms is significantly higher, around 39% on average. This includes both the register restore penalty as well as the impact of flushing the volatile lines. For example, a huge performance loss of 170% is seen for *bzip*, because frequent flushing of volatile lines significantly increases cache miss rate. The majority of the benchmarks clearly favor DeCoR by a wide performance margin, with *swim*, a benchmark with very few emergencies, being a notable exception. In this case, the benchmark incurs a high delayed-commit penalty (also seen in Figure 7), but the number of rollbacks is small resulting in better performance for the explicit checkpointing scheme.

In summary, explicit checkpoint mechanisms incur unacceptable performance overheads when applied to highly-frequent transient errors, characteristic of inductive noise. In contrast, DeCoR provides reasonably low performance overheads across a wide range of sensor delays. We now turn our attention to robustness comparisons between DeCoR and existing throttling-based noise-control mechanisms.

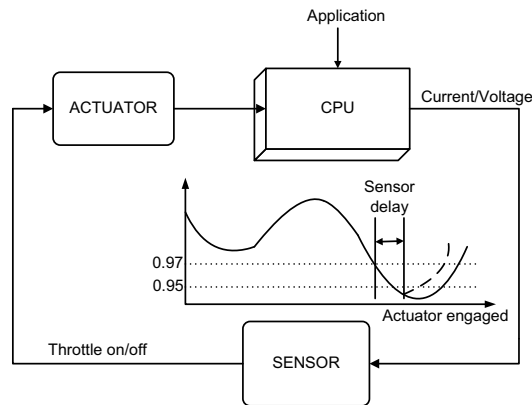


Figure 10: Feedback loop in emergency-avoidance mechanisms.

6. Robustness Comparison to HW Throttling

Solutions to deal with inductive noise must *correctly* avoid or recover from voltage emergencies, preferably with small performance loss. This section evaluates the robustness of previously proposed schemes [11, 15, 20, 21] across a wide range of PDS and sensor delay assumptions. A common attribute of these schemes is to use techniques that either spread out emergency-causing current variations in time or reduce their amplitudes. We refer to all of these techniques as *throttling*. The problem arises when one tries to use throttling alone in an *emergency-avoidance* manner to stop the processor from ever crossing a hard threshold. For the following discussion, we again assume a hard threshold of $\pm 5\%$ of the nominal voltage. Section 6.1 illustrates the feedback associated with these emergency-avoidance approaches and discusses the impact of voltage sensor delay characteristics on the correctness of these schemes. Section 6.2 evaluates the effect of PDS characteristics on the lack of correctness guarantees for *emergency-avoidance* schemes in comparison to DeCoR. We also quantify DeCoR’s performance overheads across a similar range of power delivery subsystems (Section 6.3).

6.1 Sensor characteristics

There are several different ways to reduce current variations via throttling mechanisms such as frequency throttling, pipeline freezing, pipeline firing, issue ramping, or changing the number of the available memory ports [11, 15, 20, 21]. All of these mechanisms rely on voltage or current sensors to detect threshold crossings indicative of noise-margin violations having occurred or about to occur. As shown in Figure 10, the sensor controls the throttling mechanism (i.e., the actuator). Assuming that the control logic and actuation mechanism can react quickly, the main bottleneck in throttling for emergency-avoidance is the speed of the sensors. This latency sets the speed of the overall feedback loop. There are many ways to build sensors that can trade-off delay and precision. Hence, it is important to understand the impact of the inherent delay and inaccuracy associated with the sensors on the different emergency-avoidance schemes.

Recall Figure 4(a), which shows the number of cycles from the crossing of the $\pm 3\%$ soft threshold to the crossing of the $\pm 5\%$ hard threshold for the benchmark *crafty*. It took only 3 cycles. In general for the avoidance schemes, the maximum allowable sensor delay is largely determined by the minimum number of cycles for voltage to transition between the two thresholds. This suggests that for the emergency-avoidance throttling mechanism to work correctly for *crafty* with our package model, the sensors need to detect the soft

threshold crossing within two cycles, leaving only one cycle for the actuator mechanism. Since noise violations are rare events, one might argue that the fraction of those occurring with such a steep slope might be extremely rare. Unfortunately, we need only a single noise-margin violation to disrupt the reliability of our processor circuits and cause incorrect program execution. Consequently, all such situations must be avoided.

To guarantee correct operation, emergency-avoidance throttling mechanisms must throttle before a hard emergency actually occurs. In order to accommodate potentially long sensor delays, one can consider increasing the separation between the hard and soft thresholds. However, conservatively setting the soft threshold increases the number of false alarms, where voltage variations are unnecessarily flagged as requiring throttling. Since the processor incurs a performance penalty every time throttling is engaged (i.e., the program slows down), we would like to engage the throttling mechanism only when we are sure that the hard threshold would otherwise be crossed.

Figure 11 illustrates the distribution of false alarms across our benchmarks for several different soft threshold levels. A soft threshold of $\pm 3\%$ leads to an 80% false alarm rate, averaged across all the benchmarks. Increasing thresholds closer to the noise margins leads to fewer false alarms, but the percentage is still relatively high. In comparison, DeCoR does not incur any false alarms since only the occurrence of voltage emergencies (hard threshold crossings) is detected.

6.2 Robustness across package choices

As discussed in Section 3, a processor’s susceptibility to voltage emergencies is tightly coupled to the underlying power-delivery subsystem. Hence, correctness of any proposed solution depends on assumptions made about the package and/or processor models. In this section we show that the *correctness* of previously proposed throttling schemes vary with respect to current swings (dependent on processor architecture), resonant frequencies (i.e., packaging assumptions), and sensor delays. In contrast, DeCoR’s performance is affected by these parameters, but correctness is not. We chose to sweep two of the three parameters governing power-delivery subsystem characteristics—Q factor and resonance cycles—as changing either one affects the peak impedance of the system. We also evaluate and compare the performance impact of different package characteristics on the proposed DeCoR scheme.

The techniques discussed in Section 6 aim to avoid emergencies by reducing current fluctuations via system throttling in response to detection of voltage droop [15] or repetitive current patterns [20]. Figure 12 presents contour plots of the number of noise-margin violations for *bzip* across different package characteristics, while employing an aggressive, $0.5\times$ frequency-throttling mechanism that responds to voltage swings. With a soft threshold of $\pm 3\%$, we consider two sensor delay scenarios – a sensor delay of 0 (Figure 12(a)) and a sensor delay of 5 (Figure 12(b)). Even with an optimistic sensor delay of 0, this throttling scheme fails to prevent noise-margin violations for packages with Q greater than 2, leading to correctness violations. For more realistic sensor delays, the number of violations increases by two orders of magnitude, and even packages with relatively low Q are sometimes unable to avoid noise-margin violations. The package characteristics assumed in [15] lie in the small region (upper left corner) where the throttling mechanism is effective. These results show that a throttling-based emergency-avoidance scheme cannot alone guarantee correctness for a wide range of designs.

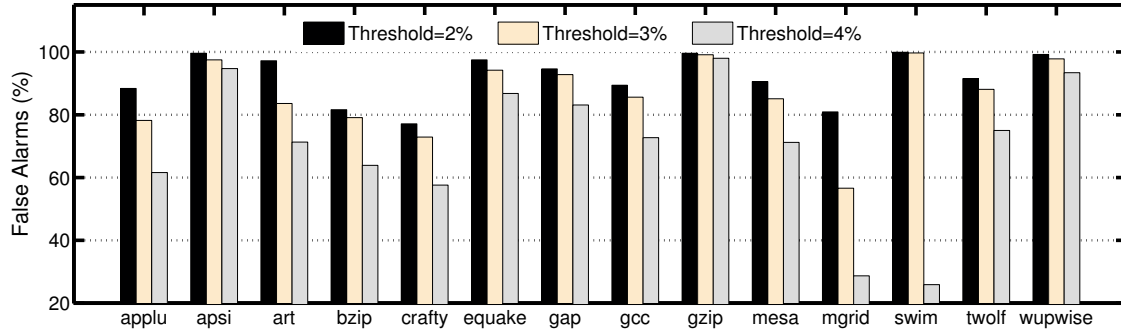


Figure 11: Distribution of false alarms generated by emergency-avoidance mechanisms for different soft threshold levels.

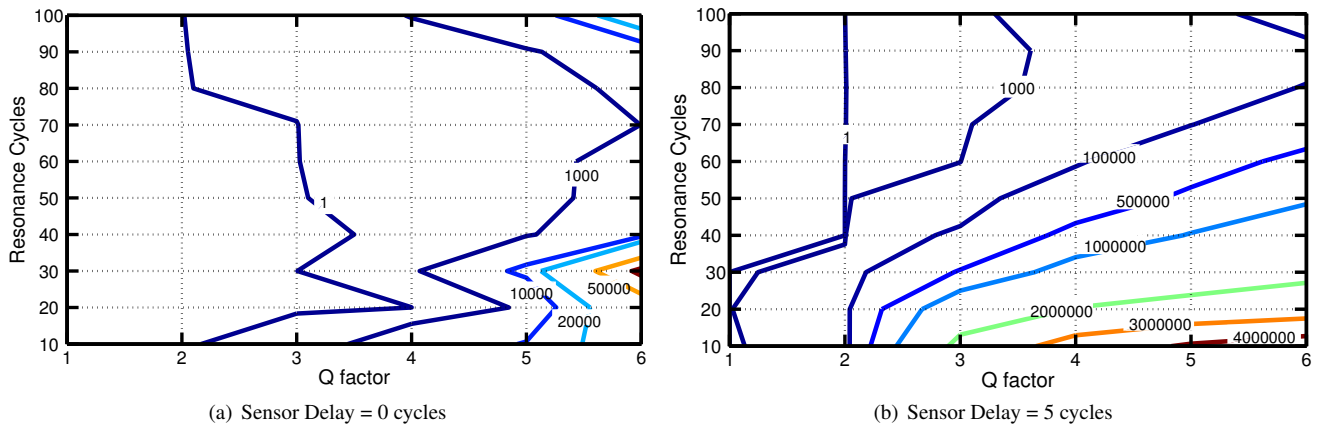


Figure 12: Evaluation of robustness for the emergency-avoidance scheme in [15], showing the number of noise-margin violations for SPEC benchmark *bzip* across different package characteristics and sensor delays.

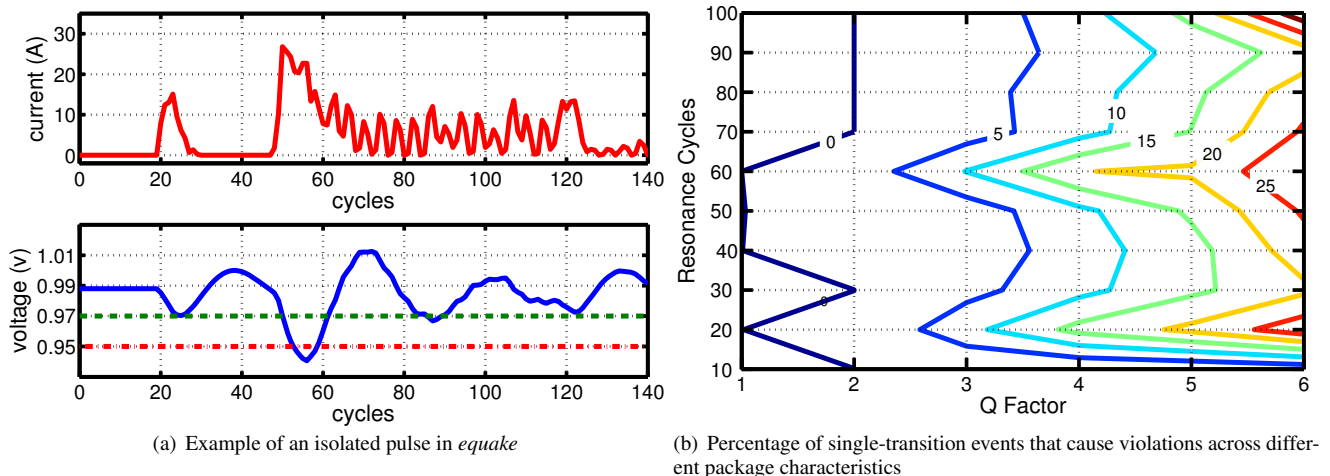


Figure 13: Evaluation of resonance tuning scheme [20] for *equake*.

Resonance tuning is another throttling mechanism, which assumes voltage emergencies are caused by repeating high-low/low-high current transitions occurring within the resonance band [20]. We consider the mechanism proposed in [20] and compute the corresponding parameters for each package with an *initial response threshold* of 2. In addition to resonant pulses, several single-transition events (single current pulses within the resonance band) occur for different packages, and a significant fraction of these events cause noise-margin violations. For example, Figure 13(a)

illustrates an example of a such an isolated current pulse found in SPEC benchmark *equake* that causes voltage to swing below the hard threshold. Figure 13(b) shows a contour plot delineating the fraction of single-transition events seen in *equake* that cause noise-margin violations across different package characteristics. These results show that the scheme proposed in [20] would not be able to detect such pulse emergencies for packages with $Q > 2$. On the other hand, resonance tuning techniques could be used to enhance DeCoR by filtering out many potential rollbacks while

Package	Z (mOhm)	Q	Resonance Cycles	Comment
pkg1	2.252	1.879	60	Used in [15] 30-70A
pkg2	2.755	2.83	100	Used in [20] 35-105A
pkg3	4.763	2.657	30	Pentium IV [5] 16-50A
pkg4	16.79	6	30	Worst package 16-50A

Table 2: Four packages with different characteristics.

relying on DeCoR to provide correctness guarantees for single-transition events. DeCoR’s correctness guarantee would also ease the implementation of the resonance tuning approach.

6.3 DeCoR’s performance across package assumptions

In contrast to the two schemes evaluated above, the DeCoR scheme proposed in this paper provides correctness guarantees across all of the different package assumptions for the underlying power-delivery network. However, there is a rollback penalty whenever a noise-margin violation occurs, and performance degrades for lower-cost package solutions. Table 2 provides a description of different packages, sorted by impedance. Pkg3 represents the power-delivery subsystem that we use in our model, based on the Pentium IV. Pkg1 and Pkg2 represent packages used in related work [15, 20] based on the Alpha 21264 and 21364. Pkg1 represents a reasonably good package as it has a low target impedance and low Q. Similarly Pkg2 is also a reasonably good package, with low impedance, but slightly higher Q and much higher resonance cycles. Table 2 also notes the current range (max current to min current) of the processors assumed in the related work. For comparison, we also include a fourth package, Pkg4, that represents a low-cost package with very high impedance and high Q. Such poor characteristics can be a result of high package-to-chip inductance, due to fewer or low-cost bump connections, or less decoupling capacitance being available on the chip.

Figure 14 presents surface plots of how performance of DeCoR varies for *crafty* across a wide range of package characteristics and two sensor delays.² Package models with a Q of 3 or lower result in less than 10% performance loss, but performance degrades rapidly as Q increases. Packages with higher resonance cycles tend to have less impact on performance, which suggests the current profiles in this benchmark have more activity around smaller resonance cycles.

In summary, investigation of various avoidance and recovery schemes applied across a wide range of systems with different packaging characteristics shows only the proposed DeCoR scheme can be a generally applicable solution to handle voltage emergencies. As Q increases, DeCoR remains robust albeit with increasing performance penalties, while the emergency-avoidance throttling schemes fail to guarantee correctness.

7. Related Work

Prior works that handle noise-margin violations use throttling techniques to either spread out current variations in time or reduce their amplitude. Sections 6 and 6.2 provide detailed discussions and comparisons between DeCoR and the previously proposed approaches [11, 15, 20, 21].

² We chose *crafty* for this analysis, because this was the worst SPEC benchmark, representing the worst-case results for our scheme.

As mentioned in Section 2 and 5.3, reactive schemes have been proposed to deal with soft errors, but cannot be applied to noise-margin violations. For example, Wang and Patel present a coarse-grained checkpoint-restart mechanism to recover from soft errors, relying on a separate checkpoint hardware structure [29]. Due to the rare occurrence of radiation strikes, they assume checkpoint recovery mechanisms have zero performance cost and primary attention is given to soft error detection and coverage.

In another related work, Razor [9], the authors propose a circuit-level mechanism to dynamically detect and correct timing failures by augmenting critical flip-flops in the microprocessor pipeline with shadow latches. These shadow latches rely on a delayed clock to provide additional timing margins and enable detection of speed-path failures. However, Razor may be costly to implement in a high-performance out-of-order core with several large array structures and tight timing paths. A recent study by Annavaram et al. [3] shows the distribution of timing margins for different paths across functional blocks in the Intel Core Duo microprocessor have hundreds of paths within 10% timing margins. This suggests that voltage-induced violations are likely to affect many paths.

DIVA also provides a method to dynamically detect and recover from transient errors [4, 7, 30]. This scheme relies on a checker processor that runs in parallel with the main out-of-order core, checking results prior to committing the instructions. None of the existing DIVA papers specifically address nor provide detailed analysis of how well it can cope with inductive noise. DIVA assumes that all reads and writes to all registers and memories will complete without error, requiring extensive TM-protected zones within the processor core given a noisy voltage environment. In addition, DIVA requires duplicate TM-protected functional units (e.g., INT/FPU/SSE units) that consume additional power and area resources. In contrast, DeCoR requires little, if any, additional hardware, and only the D-cache and register file write ports must be TM-protected. If the DIVA detection mechanism is already in place within a processor, it can replace explicit voltage sensors to detect failures due to voltage noise within the DeCoR framework.

8. Conclusion

This paper presents a delayed-commit and rollback mechanism to handle inductive noise in microprocessors. We divide the processor into two zones: a timing-margin protected zone encompassing selected structures implemented with conservative timing margins to handle the worst-case *di/dt* drops, and a rollback protected zone encompassing all of the processor structures protected by DeCoR from voltage-induced timing violations. Our scheme delays commits to the processor state by speculatively holding processor updates until it can verify that no noise-margin violations have occurred. In the event of a noise margin violation, the rollback returns the processor to a guaranteed correct state.

DeCoR is insensitive to the performance of voltage or current sensors and the characteristics of the power delivery subsystem. We demonstrate that our approach is robust across a wide range of package assumptions and sensor delays. Moreover, this approach simply relies on existing buffering and flush mechanisms present in modern microarchitectures and, thus, can be implemented with low cost. Experimental evaluation of overheads associated with our framework show acceptable performance loss relative to an ideal machine with no voltage emergencies. For sensor delays between 5 and 10 cycles, the observed performance loss is between 3% and 5%. Moreover, the performance impact of DeCoR is much lower than using explicit checkpoint-rollback schemes to tackle voltage emergencies (7% versus 39% for a sensor delay of 20 cycles).

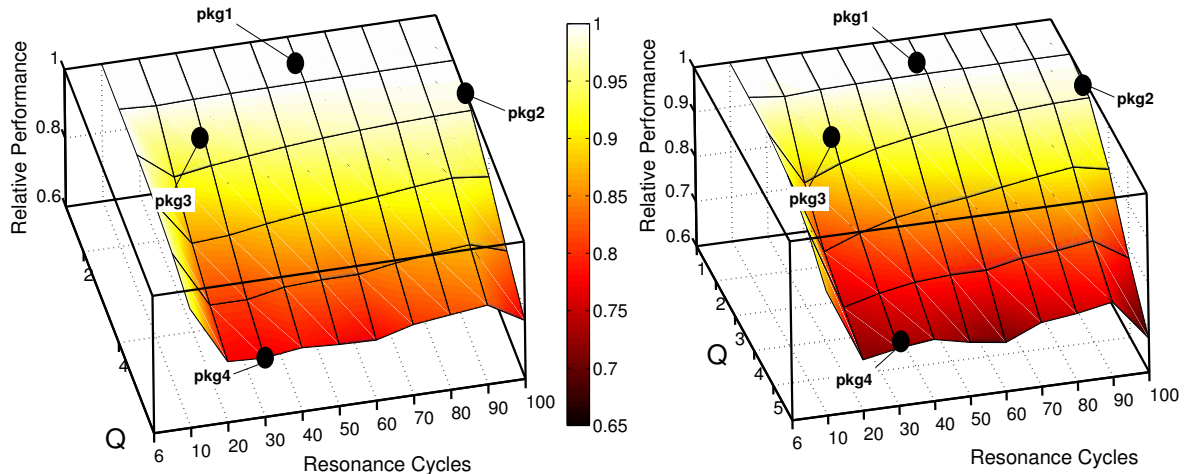


Figure 14: Performance impact of delayed-commit and rollback on *crafty* for different packages and sensor delays of 0 (left plot) and 5 cycles (right plot).

Acknowledgments

This work is supported by National Science Foundation grants CCF-0429782 and CSR-0720566. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

- [1] H. Akkary, R. Rajwar, and S. T. Srinivasan. Checkpoint Processing and Recovery: Towards Scalable Large Instruction Window Processors. In *MICRO 36: 36th annual IEEE/ACM International Symposium on Microarchitecture*, 2003.
- [2] H. Ando and et al. A 1.3GHz Fifth Generation SPARC64 Microprocessor. *IEEE Journal of Solid-State Circuits*, 38(11), Nov. 2003.
- [3] M. Annawaram, E. Grochowski, and P. Reed. Implications of Device Timing Variability on Full Chip Timing. In *HPCA '07: High-Performance Computer Architecture*, 2007.
- [4] T. M. Austin. DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design. In *MICRO 32: 32nd annual ACM/IEEE International Symposium on Microarchitecture*, 1999.
- [5] K. Aygun, M. J. Hill, K. Eilert, R. Radhakrishnan, and A. Levin. Power Delivery for High-Performance Microprocessors. *Intel Technology Journal*, 9(4), Nov. 2005.
- [6] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a Framework for Architectural-level Power Analysis and Optimizations. In *ISCA*, 2000.
- [7] S. Chatterjee, C. Weaver, and T. Austin. Efficient Checker Processor Design. In *MICRO 33: 33rd annual ACM/IEEE international symposium on Microarchitecture*, 2000.
- [8] W. El-Essawy and D. Albonesi. Mitigating Inductive Noise in SMT Processors. In *ISLPED*, 2004.
- [9] D. Ernst, N. Kim, S. Das, S. Pant, R. Rao, T. Pham, K. F. C. Ziesler D. Blaauw, T. Austin, and T. Mudge. Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation. In *MICRO'03*, 2003.
- [10] S. M. et al. 1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold Voltage CMOS. *IEEE JSSC'95*, 30(8), 1995.
- [11] E. Grochowski, D. Ayers, and V. Tiwari. Microarchitectural Simulation and Control of di/dt-induced Power Supply Voltage Variation. In *HPCA'02*, 2002.
- [12] K. Hazelwood and D. Brooks. Eliminating Voltage Emergencies via Microarchitectural Voltage Control Feedback and Dynamic Optimization. In *ISLPED*, 2004.
- [13] D. Herrell and B. Becker. Modelling of Power Distribution Systems for High-Performance Microprocessors. In *IEEE Transactions on Advanced Packaging*, volume 22, October 1999.
- [14] N. James and et al. Comparison of Split-Versus Connected-Core Supplies in the POWER6 Microprocessor. In *ISSCC 2007*, Feb 2007.
- [15] R. Joseph, D. Brooks, and M. Martonosi. Control Techniques to Eliminate Voltage Emergencies in High Performance Processors. In *Int'l Symposium on High-Performance Computer Architecture*, 2003.
- [16] N. Kirman, M. Kirman, M. Chaudhuri, and J. Martinez. Checkpointed Early Load Retirement. In *HPCA '05: 11th International Symposium on High-Performance Computer Architecture*, 2005.
- [17] J. F. Martinez, J. Renau, M. C. Huang, M. Prvulovic, and J. Torrellas. Cherry: Checkpointed Early Resource Recycling in Out-of-order Microprocessors. In *MICRO*, 2002.
- [18] S. S. Mukherjee, C. T. Weaver, J. Emer, S. K. Reinhardt, and T. Austin. Measuring Architectural Vulnerability Factors. *IEEE MICRO'03*, 23(6), November 2003.
- [19] S. Narayanasamy, G. Pokam, and B. Calder. BugNet: Continuously Recording Program Execution for Deterministic Replay Debugging. In *ISCA '05*, 2005.
- [20] M. Powell and T. N. Vijaykumar. Exploiting Resonant Behavior to Reduce Inductive Noise. In *Int'l Symp. on Computer Architecture*, Jun 2004.
- [21] M. D. Powell and T. N. Vijaykumar. Pipeline muffling and a priori current ramping: architectural techniques to reduce high-frequency inductive noise. In *Int'l Symposium on Low Power Electronics and Design*, 2003.
- [22] E. Rotenberg. AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors. In *FTCS '99: Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*, 1999.
- [23] S. Rusu and et al. A Dual-Core Multi-Threaded Xeon Processor with 16MB L3 Cache. *IEEE Journal of Solid-State Circuits*, 42(1), 2006.
- [24] Semiconductor Industry Association. International Technology Roadmap for Semiconductors, 2005.
- [25] S. Shyam, K. Constantinides, S. Phadke, V. Bertacco, and T. Austin. Ultra Low-Cost Defect Protection for Microprocessor Pipelines. *ASPLOS-XII*, 2006.
- [26] L. Smith, R. E. Anderson, and T. Roy. Chip-package Resonance in core Power Supply Structures for a High Power Microprocessor. In *Pacific Rim/ASME International Electronic Packaging Technical Conference and Exhibition*, July 2001.
- [27] D. J. Sorin, M. M. K. Martin, M. D. Hill, and D. A. Wood. Fast Checkpoint/Recovery to Support Kilo-instruction Speculation and Hardware Fault Tolerance. Computing Science Technical Report CS-TR-2000-1420, University of Wisconsin-Madison, october 2000.
- [28] L. Spainhower and T. Gregg. G4: A fault-tolerant CMOS mainframe. In *FTCS '98: Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*, 1998.
- [29] N. J. Wang and S. J. Patel. ReStore: Symptom-Based Soft Error Reduction in Microprocessors. *IEEE Trans. Dependable Secur. Comput.*, 3(3), 2006.
- [30] C. Weaver and T. M. Austin. A Fault Tolerant Approach to Microprocessor Design. In *DSN '01: 2001 International Conference on Dependable Systems and Networks (formerly: FTCS)*, 2001.
- [31] S. Wijeratne and et al. A 9GHz 65nm Intel Pentium 4 Processor Integer Execution Core. In *ISSCC'2006*, Feb 2006.