# Using ECC Feedback to Guide Voltage Speculation in Low-Voltage Processors

Anys Bacha
*Computer Science and Engineering*
*The Ohio State University*
*bacha@cse.ohio-state.edu*

Radu Teodorescu
*Computer Science and Engineering*
*The Ohio State University*
*teodores@cse.ohio-state.edu*

***Abstract*—Low-voltage computing is emerging as a promising energy-efficient solution to power-constrained environments. Unfortunately, low-voltage operation presents significant reliability challenges, including increased sensitivity to static and dynamic variability. To prevent errors, safety guardbands can be added to the supply voltage. While these guardbands are feasible at higher supply voltages, they are prohibitively expensive at low voltages, to the point of negating most of the energy savings. Voltage speculation techniques have been proposed to dynamically reduce voltage margins. Most require additional hardware to be added to the chip to correct or prevent timing errors caused by excessively aggressive speculation.**

**This paper presents a mechanism for safely guiding voltage speculation using direct feedback from ECC-protected cache lines. We conduct extensive testing of an Intel Itanium processor running at low voltages. We find that as voltage margins are reduced, certain ECC-protected cache lines consistently exhibit correctable errors. We propose a hardware mechanism for continuously probing these cache lines to fine tune supply voltage at core granularity within a chip. Moreover, we demonstrate that this mechanism is sufficiently sensitive to detect and adapt to voltage noise caused by fluctuations in chip activity. We evaluate a proof-of-concept implementation of this mechanism in an Itanium-based server. We show that this solution lowers supply voltage by 18% on average, reducing power consumption by an average of 33% while running a mix of benchmark applications.**

## I. Introduction

Handheld computers (such as smartphones and tablets) represent the fastest growing segment of the computing industry. These systems are also increasingly power constrained by demands for high performance coupled with expectations of long battery life. In this context, low-voltage operation is emerging as a promising energy-efficient solution for the microprocessors powering these systems [6], [10], [21].

Unfortunately, chips operating at low voltages face a host of challenges, including decreased reliability and higher sensitivity to parameter variation (process, temperature, voltage noise, etc.). The most common approach for dealing with these issues at nominal voltages is to add conservative guardbands to the supply voltage ($V_{\text{dd}}$) of the chip. In other words, the chip will run at a higher voltage and/or lower frequency than necessary in order to prevent timing errors and other failures that only occur under worst-case operating conditions. While these guardbands are feasible (albeit inefficient) at nominal voltages, they are prohibitively expensive at low voltages. A typical guardband of 100mV (or 10% of the nominal $V_{\text{dd}}$) represents almost 20% of the $V_{\text{dd}}$ of a low-voltage chip running at 500mV. Employing such high guardbands can negate most of the energy benefits of low-voltage chips.

Previous work has proposed voltage speculation techniques that dynamically reduce voltage margins at runtime. The idea is to gradually lower supply voltage while keeping the processor frequency constant, saving power without impacting performance. These solutions either detect and recover from timing errors, as in Razor [12], or avoid errors altogether with the help of timing monitoring circuits as in work by Lefurgy et al. [20]. These approaches rely on dedicated hardware for error detection or avoidance.

In previous work [4], we presented a firmware-based voltage speculation solution that leverages feedback from on-chip error correcting code (ECC) hardware to safely adjust the supply voltage. When correctable errors are reported by the ECC logic, the voltage is raised to a safe level. The key observation made in the aforementioned work – based on experiments on real hardware – is that these benign ECC events are always triggered before actual errors occur. The system reduces $V_{\text{dd}}$ by 10%, on average, saving substantial amounts of power. However, the system relies on the actual workload to exercise sensitive cache lines that trigger correctable errors. As a result, the system is overly conservative, with most cores running at safe voltage levels determined during off-line calibration. In addition, because the system is based in firmware, it incurs a runtime overhead for each handled error. This leads to diminishing energy savings as the voltage is pushed lower and more correctable errors are triggered.

This paper presents a new ECC-based voltage speculation system that uses simple hardware support that directly targets sensitive cache lines to accurately and continuously monitor timing margins. The system is designed to take advantage of chip characteristics that are specific to low-$V_{\text{dd}}$

This work was supported in part by HP, the National Science Foundation under grants CCF-1117799 and CCF-1253933, and the Defense Advanced Research Projects Agency under the PERFECT (DARPA-BAA-12-24) program.

operation. We used an Intel Itanium processor (similar to the one examined in [4]) to characterize the voltage margins of the chip at low voltages (around 600mV). We compared the chip's characteristics at low voltage with those exhibited at the processor's nominal $V_{dd}$ of 1.1V.

We find that instruction and data caches are the most sensitive structures at low voltages. These structures always trigger correctable errors first as the supply voltage is lowered while keeping the frequency constant. Moreover, these correctable errors are encountered consistently in the same cache lines; although the addresses of such lines vary from core to core. In addition, we find that the spread between the $V_{dd}$ at which a sensitive line reports an error and the voltage at which the system crashes is almost 4× larger at low $V_{dd}$ compared to that at the nominal $V_{dd}$. This gives every single core in the system we tested a wide spread of safe operating voltages below the $V_{dd}$ that triggers the first correctable error. It allows the system much more aggressive speculation than is possible in the nominal $V_{dd}$ region. Overall, we find that correctable errors are more reliable and more consistent predictors for timing margins at low $V_{dd}$ compared to the high $V_{dd}$ region.

We also find significant variability in the minimum $V_{dd}$ that can be reached by individual cores, likely due to the impact of manufacturing process variation on circuit delay. This variability is about 4× higher than at nominal $V_{dd}$, making core-level voltage tuning solutions more attractive at low-$V_{dd}$.

We evaluate our voltage speculation solution on a real hardware platform that uses Intel Itanium 9560 processors. We simulate some of the hardware-based components in software running on a dedicated thread. We conduct dozens of hours of testing of multiple chips and cores and found our speculation system to operate reliably and without data corruption. Moreover, we demonstrate that this mechanism is sufficiently sensitive to detect and adapt to voltage noise caused by fluctuations in chip activity. We find that our solution lowers $V_{dd}$ by 18% on average while running applications from CoreMark, SPECjbb2005, and SPEC CPU2000 benchmark sets. This reduces power consumption by an average of 33% with no performance impact.

Overall, this paper makes the following contributions:

- Characterizes the low-voltage behavior of a production microprocessor and demonstrates the amplified process variation effects on memory devices.
- Presents a new, more reliable, precise, and aggressive ECC-based voltage speculation solution specifically designed to take advantage of low-voltage characteristics.
- Shows that the technique is sufficiently sensitive to detect and adapt to voltage noise caused by processor activity changes.
- Evaluates the proposed solution on a real hardware platform based on Intel's Itanium 9560 processors.

The rest of this paper is organized as follows: Section II analyzes the voltage speculation potential at low voltages. Section III details the architecture of the proposed ECC-based voltage speculation system. Sections IV and V present the methodology and experimental evaluation. Section VI details related work; and Section VII concludes.

## II. Voltage Speculation Potential at Low-Vdd

Caches are generally the most vulnerable structures to low-$V_{dd}$ operation [1], [5], [26], [27], [37]. They are optimized for density and therefore use the smallest transistors available in a given technology node. These transistors are the most affected by random variations such as dopant density fluctuations, leading to imbalance between the SRAM cell inverters. As the voltage is lowered, these cells may fail to reliably store data. Low-voltage operation coupled with variation can also slow down access transistors in the SRAM arrays. As a result, data reads may not complete in the expected timeframe, leading to timing and other errors.

While many improvements and optimizations have made SRAM cells more robust to low-voltage operation, caches generally determine the supply voltage floor at which chips can operate reliably [2], [8], [11], [34], [35] (also known as $V_{ccmin}$). Our study adds empirical evidence from experiments on production processors to support this conclusion.

To help motivate this work, we explore the limits of speculation in low-$V_{dd}$ processors, as well as the potential for using correctable errors to dynamically choose safe voltage levels. We begin by examining the voltage margins available for speculation when running a production microprocessor at low $V_{dd}$.

### A. Voltage Margins

For this study, we use a system with an Intel Itanium II 9560 8-core processor [29]. More details about the experimental setup are presented in Section IV. We conduct two sets of experiments. In the first, we set the frequency and $V_{dd}$ at the nominal level of 2.53GHz. In the second, we set the processor frequency to 340MHz, the lowest supported, in order to test the limits of this system. A production low-voltage system would likely run at higher frequencies (500MHz-1GHz) in order to keep performance at reasonable levels. In both experiments, we gradually lower supply voltage while keeping the frequency fixed and the system under load. We run a stress test application consisting of CPU-intensive kernels, as well as cache and memory-intensive kernels. For each core we record the lowest $V_{dd}$ at which it functions correctly with no crashes or data corruption.

Figure 1 shows the minimum safe voltage of each core for both 2.53GHz and 340MHz relative to their respective nominal $V_{dd}$s. At high frequency, the average minimum safe voltage is more than 10% below the chip's high-$V_{dd}$ nominal of 1.1V. This is a typical guardband in CPUs today. At 340MHz, the lowest safe $V_{dd}$ ranges from 600 to 660mV

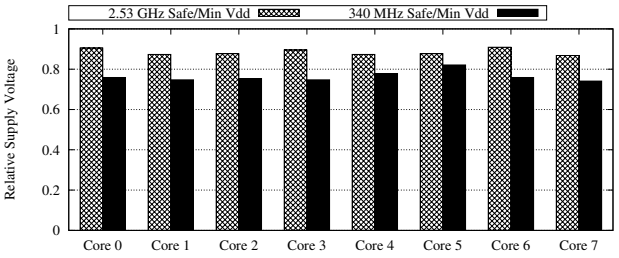



Figure 1. Lowest safe $V_{dd}$ for each core of an Itanium CMP at both high and low frequencies.

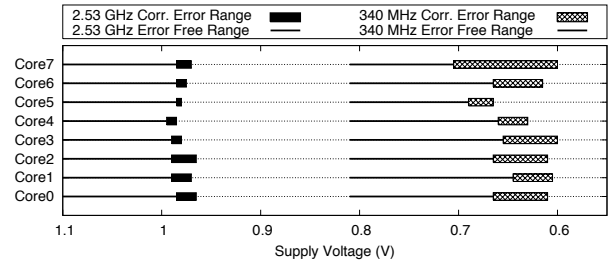



Figure 2. Voltage speculation range for each core at high and low frequencies.

with an average of 618 mV. This is 23% lower than the low-$V_{dd}$ nominal of 810mV. This indicates that voltage speculation at low-$V_{dd}$ has the potential to double the energy savings obtained at high-$V_{dd}$.

The data also shows core-to-core variation in the minimum safe voltage increases at low-$V_{dd}$, exceeding 10%. This is due to process variation and suggests that core-level voltage speculation is potentially beneficial at low $V_{dd}$.

### *B. Correctable Error Range*

We also find that, as $V_{dd}$ approaches the lowest safe level, the hardware reports correctable error events that occur in the chip's caches. Figure 2 illustrates the voltage speculation ranges for both the high and low $V_{dd}$ cases. The solid lines represent voltage ranges over which the cores exhibit no correctable errors. The bars to the right of the solid lines mark the voltage ranges over which correctable errors occur. The bars stop at the lowest safe $V_{dd}$.

The figure shows that in addition to the voltage speculation margin being much larger at low-$V_{dd}$, the range of voltages over which correctable errors occur is 4× larger at low-$V_{dd}$ compared to high-$V_{dd}$. This has important implications for ECC-driven voltage speculation. At nominal $V_{dd}$, the smaller error range limits the aggressiveness of the voltage speculation. This is because correctable errors are only raised close to the minimum safe voltage. For this reason, many of the cores examined in [4] were constrained to run at voltages that were higher than necessary. At low $V_{dd}$, the voltage speculation system receives earlier feedback about approaching timing margins. This feedback spans a wider

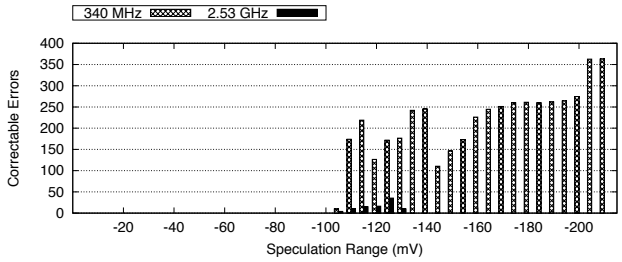



Figure 3. Average correctable errors across all cores vs. voltage speculation range at high and low frequencies.

voltage range, allowing speculation to be more aggressive and bring $V_{dd}$ substantially lower. This means that each core should be able to routinely run in an environment in which correctable errors occur regularly (region marked by shaded bars in Figure 2), without affecting the correctness of the execution.

We also found that the number of correctable errors raised at low-$V_{dd}$ is higher than at high-$V_{dd}$. Figure 3 shows the average correctable error rate as a function of $V_{dd}$ for both experiments. The X-axis in the figure represents the voltage distance from the nominal levels of each experiment. The origin on the X-axis represents the nominal $V_{dd}$ for both the high frequency and the low frequency cases. We can see that for both experiments there is a voltage range that exceeds 100mV in which no correctable errors are triggered. If voltage is lowered more than 110mV below nominal, correctable errors are triggered. As the voltage is lowered further, some cores reach their minimum safe voltage. At each voltage level we report the average error rate only across the cores that are still active at that voltage.

For the high-$V_{dd}$ case, the error rate peaks at approximately 35 errors over a 5 minute interval before the last core reaches its minimum safe voltage. The low-$V_{dd}$ case generates many more errors, reaching an average of more than 350 errors over the same time interval. The average error rate generally increases as the $V_{dd}$ is lowered. There is some noise in the data caused by the inclusion of a decreasing number of cores in the average as the $V_{dd}$ is lowered and cores reach their minimum $V_{dd}$.

Although this may appear counterintuitive, the higher correctable error rate is helpful to the hardware-based ECC-guided voltage speculation. Raising correctable errors more frequently and consistently helps provide constant feedback to the speculation system. This gives the system more precise guidance about approaching timing margins and makes it easier to accurately target a certain correctable error rate.

### *C. Correctable Error Types*

We find that the types of errors exhibited at low $V_{dd}$ differ from those at nominal $V_{dd}$. At high $V_{dd}$, a mix of cache and register file correctable errors are triggered, as reported in [4]. At low $V_{dd}$, we only encounter errors in the instruction

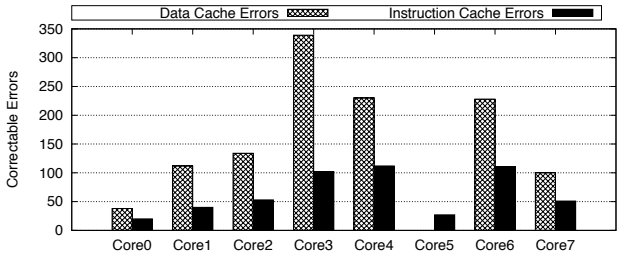



Figure 4. Number and type of correctable errors for each core for a 5 minute run under load.

and data L2 caches. We believe this is due to the different sizing of the SRAM cells used in the register files vs. caches. Caches are designed using the smallest cells to increase density, which makes them relatively more vulnerable to low-voltage operation. The fact that we never see L1 cache errors likely indicates that these caches are built using larger, more robust SRAM cells, or perhaps a different cell design.

Figure 4 shows the breakdown of the number of errors raised by each core while running the same workload mix – consisting of both memory and compute intensive benchmarks – for 5 minutes. The voltage of each core is set at its lowest safe level. We can see that all the cores exhibit both instruction and data cache correctable errors (with the exception of core 5 which only triggers instruction cache errors). There is also significant core-to-core variability in the number of errors triggered. This can be explained primarily by the fact that each cache has sensitive lines in different locations. Since the test workload will likely exercise some cache lines more than others, the number of errors triggered by each core differs substantially.

There is also variability in error counts between instruction and data caches of each core. This is due to the smaller miss rate in the instruction L1, resulting in fewer accesses – and therefore fewer errors – in the instruction L2 cache.

### D. Deterministic Error Distribution

An important observation we make while conducting these experiments is that the correctable errors raised by the system are deterministic. In other words, at the same $V_{dd}$ levels, cores exhibit roughly the same number of errors in multiple runs of the same workload. Moreover, we find that in each core errors are raised consistently by the same cache lines. These lines likely contain cells that are more vulnerable to low voltage than others due to process variation. Starting from this observation, we propose a new approach to guiding voltage speculation that directly targets these weak lines with the help of simple hardware. Our system is targeted and precise, enabling safer and more aggressive voltage speculation.

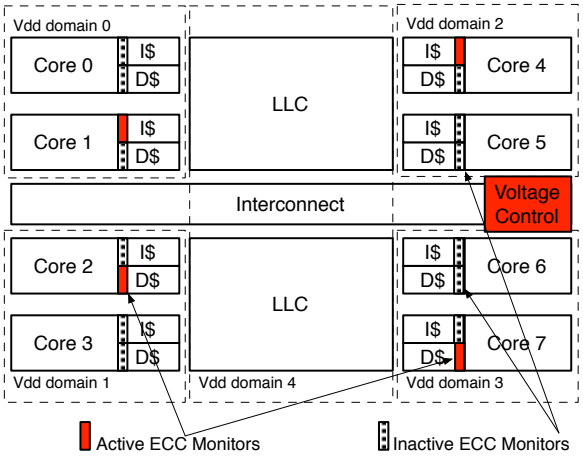



Figure 5. Overview of the voltage speculation system integrated in a chip multiprocessor with multiple $V_{dd}$ domains.

## III. Voltage Speculation Guided by ECC

We developed a voltage speculation mechanism specifically designed to take advantage of chip properties that are specific to low-voltage operation. The proposed system takes advantage of the observations that correctable errors are deterministic; and that at low voltages, the distance between the first reported correctable error and the failure $V_{dd}$ increases substantially. The voltage speculation system consists of two main components: a lightweight hardware ECC monitor that continuously probes known vulnerable cache lines and a voltage control system that uses feedback from the ECC monitor to guide $V_{dd}$ adjustments. Figure 5 shows an overview of how the voltage speculation system would be integrated into a chip multiprocessor.

### A. Hardware ECC Monitors

The ECC monitor is a hardware unit designed to continuously probe the most vulnerable cache lines in the system. The monitor consists of simple logic that generates test bit patterns and writes them into the designated cache line. A read request is issued after each write to that line. If the ECC hardware already built into the system detects a single bit error, it will correct the error and report the event to the ECC monitor. The monitor maintains two counters: an access counter and an error counter. The access counter is incremented for every read request issued by the monitor to the cache line under test. The error counter is incremented every time a correctable error event is triggered by the cache line under test. The counters are periodically reset. The ratio between the two counter values represents the correctable error rate for the line under test. This value will be used to guide voltage adjustment decisions.

ECC monitors are built into all the data and instruction cache controllers on the chip, as shown in Figure 5. However, at runtime, only a fraction of these monitors will be

activated. Since multiple cores and caches often share a voltage domain, only the most vulnerable line in that domain needs to be targeted by direct testing. Therefore, only the ECC monitor corresponding to that line's cache needs to be active; the rest can be shut down. In the case of the system in Figure 5, four ECC monitors are activated, one for each $V_{\text{dd}}$ domain that contains cores. Since there is no way of knowing at design time where the most vulnerable line will be, we need to provision all cache controllers with ECC monitors.

### B. Voltage Control System

A centralized voltage control system (Figure 5) runs on the service microcontroller available in many processors today [15], [29]. The control system periodically reads the error counters for all active ECC monitors. A voltage adjustment decision is then made based on the correctable error rate. For instance, the control system can be set to maintain the error rate somewhere between a *floor* and a *ceiling* value. When the error rate exceeds the ceiling, the voltage is raised by some small increment (e.g. 5mV). If the error rate falls below the floor, the voltage is lowered by the same increment. The floor and ceiling for the speculation algorithm can be customized to the sensitivity of the voltage domain, to account for process variation or other factors. In our implementation, we set the floor and ceiling for all voltage domains at 10% and 50% respectively.

An emergency mechanism is also in place in each hardware ECC monitor. When the error rate exceeds an *emergency ceiling* (for example 80%), an interrupt signal is sent to the voltage control system which raises the voltage for the domain by a larger increment to bring the system back into the targeted error range.

### C. System Calibration

A calibration step is necessary to configure the voltage speculation system. The voltage speculation system is designed to monitor the weakest cache line in each voltage domain. This is the cache line that triggers correctable errors at the highest $V_{\text{dd}}$. This line is identified during a simple calibration step that can be performed periodically at system boot time. Calibration involves progressively lowering the $V_{\text{dd}}$ and performing a cache sweep at each voltage level.

The cache sweep test involves both the data and instruction caches. As a mechanism to stress the data cache during this phase, a set of loads and stores are performed in cache line sized increments. In the case of the instruction cache, the stress test is built dynamically. The process is illustrated in Figure 6. A template of straight line instructions is flashed in the System Firmware ROM. The template is sized to match the L1 cache line. During boot, the template is copied from the ROM and is sequentially replicated throughout the allocated physical memory. Each template ends with a conditional branch that determines if execution

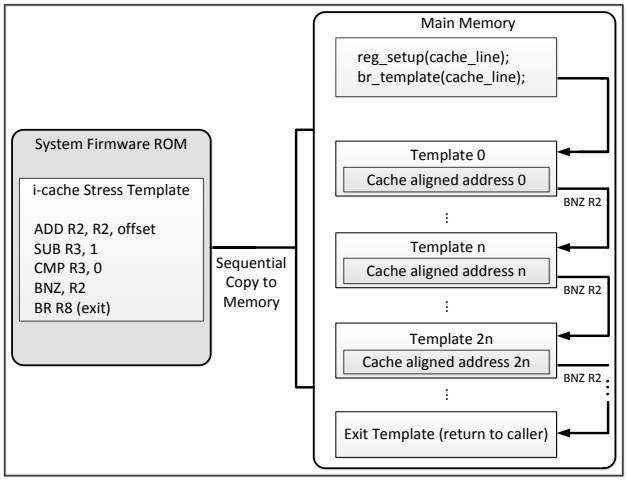



Figure 6. Illustration of the instruction cache sweep process.

must return to the caller or proceed to the next requested offset. During the instruction cache sweep, the execution branches to the immediately adjacent template until the entire cache, including all the ways, have been exercised.

The cache sweep stops when a correctable error is encountered. The set and way of associativity of the cache line that triggered the error is recorded. The corresponding ECC monitor is activated and programmed to target the newly designated line. The line is de-configured from the cache to ensure no data will be stored there. The selected line will only be used for speculation monitoring and will not store any actual data. The voltage control system is also programmed to interrogate the active ECC monitor for that voltage domain.

### D. Managing Aging and Temperature Variation

The voltage speculation system can be recalibrated periodically to determine if the error distribution has changed and a new cache line needs to be designated for monitoring. If the weakest line has changed due to aging, the ECC monitor is reprogrammed to target the newly discovered weak line. This ensures that the system can adapt to aging effects.

To verify if temperature variation can affect the correctable error distribution we conducted experiments under different temperatures by slowing system enclosure fan speeds. For variations of up to $20\,^{\circ}\text{C}$ we did not observe a measurable effect on the rate or distribution of errors.

## IV. Evaluation Methodology

Evaluation of our system was performed on a hardware platform, the BL860c-i4 Integrity Server from HP, equipped with two Intel Itanium 9560 processors, each possessing eight cores with hyperthreading. The system ran the HP-UX Operating System. Table I lists additional detailed information about the evaluation system.

| Processor | Itanium II 9560 |
|---|---|
| Cores | 8, in-order |
| Frequency | 2.53GHz (high), 340MHz (low) |
| Nominal $V_{dd}$ | 1.1V (high), 810mV (low) |
| Register file size | 1.38KB int, 1.25KB fp |
| L1 data cache | 4-way 16KB, 1-cycle |
| L1 instruction cache | 4-way 16KB, 1-cycle |
| L2 data cache | 8-way 256KB, 9-cycle |
| L2 instruction cache | 8-way 512KB, 9-cycle |
| L3 unified | 32-way 32MB, 50-cycles |
| QPI Speed | 6.4 GT/s |
| Max TDP | 170 W |
| Technology | 32nm |
| Voltage domains | 6 |
| System | HP BL860c-i4 blade |
| Memory | DDR3 32GB |
| Operating System | HP-UX 11i v3 |

Table I
ARCHITECTURAL AND SYSTEM DETAILS OF THE BL860-I4 INTEGRITY SERVER AND ITANIUM 9560 PROCESSOR [16], [17].

The low frequency is set to the lowest supported by the system, 340MHz. Since there is no published "nominal" $V_{dd}$ for this frequency, we assumed the same absolute guardband would be used at both high and low $V_{dd}$. We measured the guardband as the difference between the nominal $Vdd$ at 2.53GHz and the voltage at which the first correctable error is encountered at the same frequency. This was determined to be 100mV. We added this guardband to the $V_{dd}$ at which the first correctable error is encountered at 340MHz. This gave us a nominal $V_{dd}$ of 810mV for the low-voltage environment.

### A. Experimental Platform

We use a firmware-based framework for modeling our system on real hardware. A runtime system is implemented to model both the ECC monitor and the voltage speculation control. The functionality of the ECC monitor is implemented with the help of cache self-tests that perform targeted reads and writes to designated lines. In our system, the most vulnerable lines reside in the L2 instruction and data caches. The challenge of performing this test in firmware is that direct access to specific cache ways in the L2 is not possible. Therefore, we developed a testing routine that bypasses the L1 to effectively exercise the designated cache line within the L2.

*1) Targeted Cache Line Testing:* Figure 7 illustrates the steps involved in the targeted testing of a specific cache line. In the first step, a total of eight lines are fetched to populate each way in the L2 cache, which is 8-way set associative. To get around the L1 cache preventing accesses from reaching the L2, we fetch four other cache lines (step 2). These map to the previously used set in the L1 (the L1 is 4-way set associative), but map to a different set in the L2. This is possible since the size of the L2 cache is a multiple of the L1 cache. Once we clear the entries in the L1 cache, we access the original eight cache lines that are still resident in the L2 cache entry targeted by the self-test (step 3).

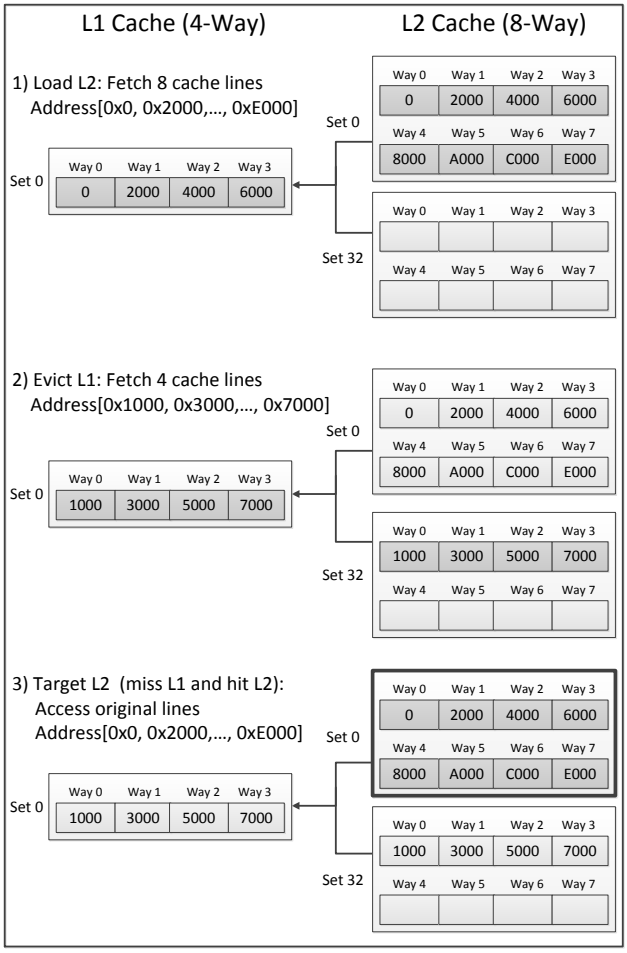



Figure 7. Execution steps for performing a targeted cache line test.

*2) Implementation of ECC Monitor:* To approximate the behavior of the hardware ECC monitor on a real platform, we dedicate one of the two hardware threads within each core for initiating and handling self-test operations that drive voltage speculation. This required disabling multi-threading at the OS level for the purpose of this study. To achieve this, System Firmware claimed ownership of each disabled thread (Thread 1) within a core, while the OS continued to use the primary thread (Thread 0) for application scheduling. This is shown in Figure 8. In most of the experiments we conducted, the benchmark thread ran on the primary hardware thread while System Firmware simultaneously ran the self-test and monitored ECC events on the secondary thread.

*3) Service Processor:* For the purpose of logging and reporting experimental data, an entire core was reserved for System Firmware use. Dedicating a core to handling such measurements greatly simplified the data collection process. However, in order to facilitate such retention of hardware

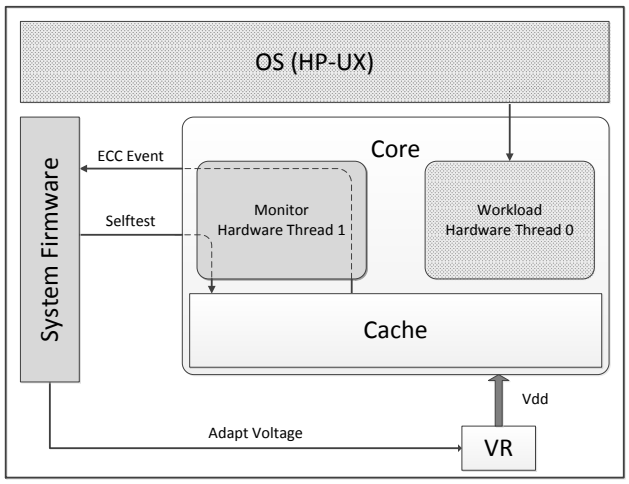



Figure 8. Overview of the ECC Monitor simulation framework.

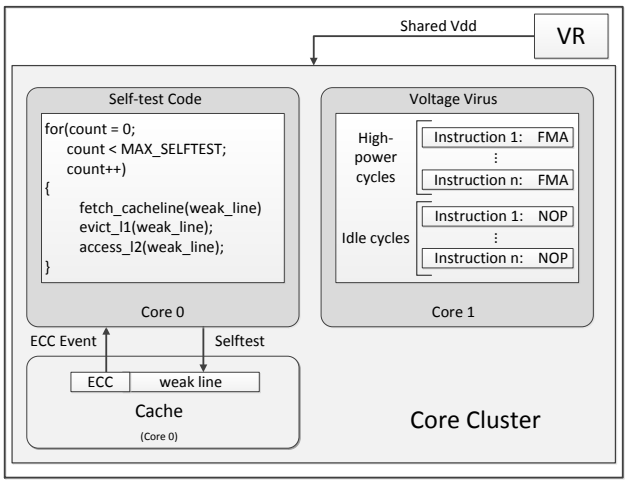



Figure 9. Overview of the noise experiment setup with the voltage virus running on the auxiliary core.

resources from the OS, additional firmware layers had to be modified. These layers are: the Advanced Configuration and Power Interface (ACPI) and the Unified Extensible Firmware Interface (UEFI). Modifying these layers enabled the live data collection we needed while the OS was active. This data included average power, voltage settings, error rate information, and coordination of voltage speculation experiments.

*4) Data Logging and Collection:* Power consumption information was collected by sampling a set of processor registers. We collected the power information for each core pair in addition to the uncore component. We also logged the temperature information for each core. To keep the logging overhead manageable for long runs, the aforementioned data was sampled every 1ms.

Special hooks were developed to record logs of the set and way of correctable cache errors reported by the hardware. These were used to characterize the correctable error profile of each core at multiple voltage levels. Error logs were also kept while running the voltage speculation algorithm. These were used to construct time based voltage and error rate traces.

The processors in this system have multiple power delivery lines – one for each pair of cores and a separate one for the "uncore" components, such as the L3 cache and memory controllers [29]. The supply voltage of each of these power lines can be independently modulated. Experiments that examined the sensitivity of each core in response to low voltage were conducted by exercising a single core at a time. The auxiliary core that shares a supply line with the one under evaluation was left idle in a tight spin-loop within System Firmware. This prevented the OS from reclaiming the core for background tasks which could skew our results. This allowed data collection at core granularity even with core pairs sharing voltage rails.

### B. Inducing Voltage Noise

An important part of the evaluation was to test the resilience of the proposed voltage speculation system under voltage noise conditions. To artificially generate noise in the supply voltage, we exploited the fact that two cores share a single supply. We use one of the cores to induce noise through the execution of a carefully calibrated "voltage virus" in an approach similar to that used by Kim et al. in [19]. This setup is illustrated in Figure 9.

The "voltage virus" consisted of a loop containing high-power instructions such as Floating-point Multiply Add (FMA) interleaved with NOPs at a 50% duty cycle. The goal was to induce the type of regular activity fluctuation pattern that has been previously reported to excite the chip's resonant frequency and cause large droops in $V_{dd}$ [14], [19], [28]. We generated multiple variants of this workload by varying the number of NOP instructions. This allowed us to sweep through multiple workload oscillation frequencies to try to match the chip's resonance frequency.

The main core of the cluster was used to monitor ECC events and detect noisy conditions through abrupt increases in the number of correctable errors.

### C. Benchmarks

Multiple benchmark suites were used in the evaluation: CoreMark, SPECjbb2005, and SPEC CPU2000. CoreMark, which consists of kernels tailored for mobile processors was configured to run a full instance of the suite on each core. SPECjbb2005 was configured in a similar fashion where a total of 8 warehouses were launched on each core under test. For SPEC CPU2000, all benchmarks were individually run on the respective cores within the CMP, with the exception of *wupwise* and *apsi*, which we could not successfully run on this system. In addition to the aforementioned industry

| Suite | Benchmark |
|---|---|
| CoreMark | list processing, matrix manipulation, state machine, CRC. |
| SPECjbb2005 | 8 warehouses |
| SPECint | gzip, vpr, gcc, mcf,crafty, parser, eon, perbmk, gap, vortex, bzip2, twolf |
| SPECfp | twolf, swim, mgrid, applu, mesa, galgel, art, equake, facerec, ammp, art, lucas, fma3d, sixtrack |
| Stress test | CPU-intensive (FP and INT) kernels. Cache and memory-intensive kernels. Designed to stress test HP servers. |

Table II
APPLICATIONS AND BENCHMARKS USED IN THE EVALUATION.

standard benchmarks, a stress test application consisting of CPU-intensive kernels, as well as cache and memory-intensive kernels, was used to characterize the processor's voltage margins. Benchmarks were run back-to-back to ensure context switches are handled correctly by the voltage speculation algorithm. Table II shows a summary of the different benchmarks used in the evaluation.

## V. EVALUATION

In this section we evaluate the benefits of aggressively lowering the supply voltage while maintaining safe operation. We show a significant reduction in voltage that leads to substantial power savings. We examine the robustness of the system in adapting to changes in workload intensity, including those sufficiently severe to lead to voltage noise. Cache line error rate sensitivity to voltage and graceful degradation is also shown. We compare the energy savings to a software-only voltage speculation solution similar to that in [4].

### A. Voltage Reduction and Power Savings

Figure 10 shows the average voltage of each core of one processor for each of the four benchmark suites we ran. The baseline reference is the low-voltage nominal $V_{dd}$ of 810mV, illustrated on the figure as the dotted red line. Our system lowers $V_{dd}$ by an average of 18% relative to the baseline. We observe large core-to-core variability with the $V_{dd}$ reduction ranging from 13% to 23% across all the cores. This is evidence of process variation effects which are more pronounced at low voltages [11], [23].

There is little variability in the voltage reduction across the four benchmark sets under evaluation. This is because our algorithm does not rely on the workload to exercise sensitive cache lines as in prior work [4]. It instead relies on targeting the weakest cache lines, making the system more precise. Significant variability in $V_{dd}$ does exist over shorter time intervals and between individual applications as the workload intensity changes.

The large reduction in supply voltage translates into substantial power savings. Figure 11 shows an average power savings of 33% across all benchmarks, again with little variability between the benchmark suites.

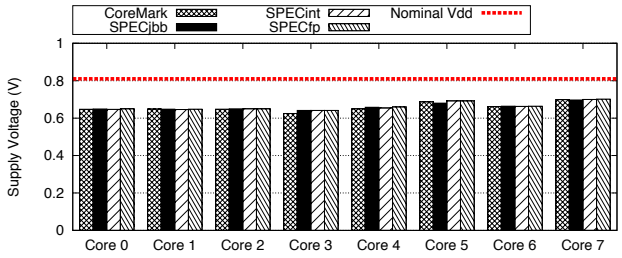



Figure 10. Average core voltages achieved through voltage speculation for each benchmark suite.

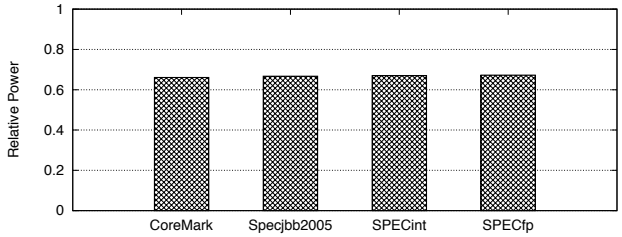



Figure 11. Total power relative to the reference voltage for each benchmark suite.

### B. Dynamic Adaptation to Workload

The voltage speculation system continuously adjusts the supply voltage to ensure reliable operation. All cores start running at their nominal voltage. Voltage is then continuously reduced or increased in steps of 5mV until the self-test reports an error rate between a floor of 10% and a ceiling of 50%. Figure 12 shows a trace of the supply voltage over time for parts of two SPECint benchmarks running back to back: *mcf* and *crafty*. The correctable error rate for the same interval is also shown in the figure.

We can see the system is able to match changing workload conditions and maintain the error rate within the targeted range. Note that the figure only shows steady-state error rate and does not include the brief transients that fall below the floor or above the ceiling $V_{dd}$s and trigger voltage changes.

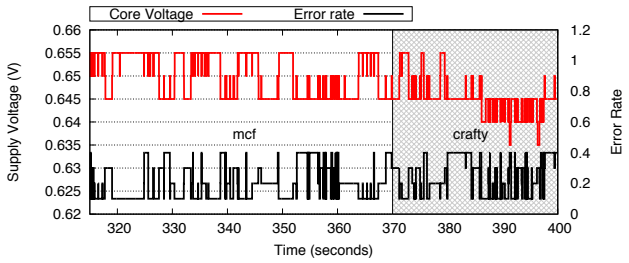



Figure 12. Dynamic adaptation of supply voltage to runtime conditions while executing *mcf* followed by *crafty* from the SPECint benchmark.

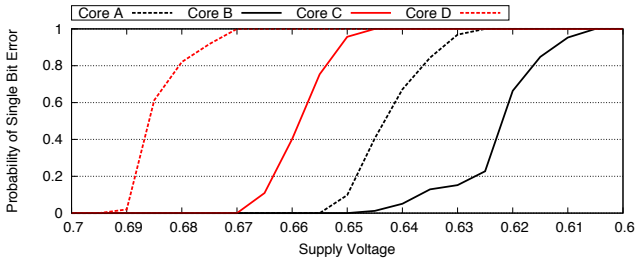



Figure 13. The probability of a single bit failure of a cache line for different cores while running the cache line self-test.

The system adapts well to context switches as the workload transitions from running *mcf* to *crafty*.

### C. Cache Line Sensitivity at Low Voltages

Our system relies on the gradual change in the probability of correctable errors in the cache lines targeted for monitoring. In order to characterize error rate sensitivity to supply voltage, we selected four cores that exhibited different error distribution profiles. We then ran the targeted self-test on one line of each core while progressively lowering $V_{dd}$. Figure 13 shows the probability of single bit errors vs. supply voltage for each of these cores. In general, the onset of errors is relatively slow. The ramp-up range (going from 0% to 100% errors) spans between 20mV for core D to over 50mV for core B. We change $V_{dd}$ in 5mV increments which gives the system sufficient resolution to keep the error rate between the floor and ceiling values.

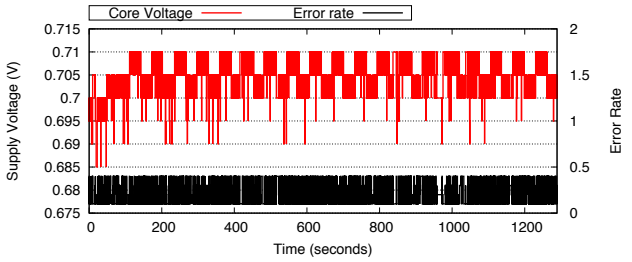
Figure 13 shows that margins of 10-20mV exist above the 50% error ceiling we used. This gives the system a margin for handling abrupt changes in dynamic conditions. In addition, correct operation continues well beyond the 100% mark before the lowest safe $V_{dd}$ is reached. This indicates that there is some potential for tailoring the values of the floor or ceiling $V_{dd}$s. We leave such optimizations for future work.

There is also significant variability between the voltages at which the 50% ceiling is reached by the different cores (0.625-0.685V). This highlights the benefits of core-level voltage assignment and adaptation.

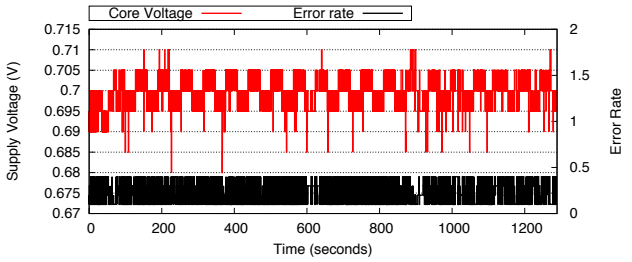
### D. Algorithm Robustness and Sensitivity to Voltage Noise

In order to evaluate the robustness of our voltage speculation algorithm, we conducted a series of tests to stress the stability of the supply voltage. The goal was to examine how the speculation system adapts to extreme operating conditions.

*1) Robustness to Activity Variation:* Abrupt changes in workload intensity lead to variation in power demand that can rapidly depress supply voltage and cause errors. In order to test how our system behaves under such conditions, we construct a stress kernel designed to induce abrupt changes in power demand.




(a) Main core idle.




(b) Main core running SPECfp.

Figure 14. Dynamic adaptation of $V_{dd}$ to workload stress induced by the stress kernel runnning on the auxiliary core.

To conduct this test under realistic conditions, we leveraged the fact that in the chip we used, every two cores share a single $V_{dd}$ domain. Therefore, we could use one of the cores in a pair to run the main workload under test and the sibling core (auxiliary core) to run the stress kernel. This setup simulates conditions in which the regular workload is disturbed by additional load on the power supply. To induce load variation, the stress kernel was scheduled to run for 30 seconds and then abruptly throttled for another 30 seconds by having System Firmware interrupt the auxiliary core. The interrupted core would then go into a low-power spin-loop inside System Firmware for 30 seconds before resuming execution of the stress kernel.

We conduct two experiments: one in which the main core is idle and one in which the main core is under load running the SPECfp suite. Figure 14 shows the $V_{dd}$ and error rate over time for these two cases. Both experiments run for 20 minutes with the auxiliary core executing the stress kernel. In both experiments, we can clearly see the $V_{dd}$ pattern change every 30 seconds as the stress kernel is periodically throttled on the auxiliary core. When the stress kernel is active, the voltage droops, reducing the timing margin and increasing the correctable error rate. Our test system detects the change and raises the $V_{dd}$. The voltage is lowered as soon as the auxiliary core begins to idle, reducing the demand on the system. Throughout the execution, the algorithm attempts to reduce $V_{dd}$ to lower values (as indicated by the short-lived drops in voltage), but generally maintains the $V_{dd}$ within a fairly narrow band for both the heavy-loaded and light-loaded cases.

The main difference between the two experiments is that the average $V_{dd}$ is lower for the SPECfp run (Figure 14(b)) compared to the idle run (Figure 14(a)). These results show that our voltage speculation algorithm adapts very well to changes in workload and stress on the supply voltage and consistently maintains the error rate within the specified interval.

*2) Robustness to Voltage Noise:* To further stress our system, we designed a "voltage virus" meant to induce voltage noise on the power distribution network. The virus consists of high power instructions interleaved with varying numbers of NOPs as described in Section IV-B. By changing the NOP count, we are effectively varying the oscillation frequency of high/low-power phases in the virus workload.

We run the targeted self-test on the main core while the voltage virus runs on the auxiliary core. We count the number of errors raised during the self-test. Figure 15 shows the error count for multiple versions of the voltage virus with NOP counts ranging from 0 to 20. For each NOP point in the figure, a total of 5000 accesses to the weak cache line in the main core were performed.

The data clearly shows a spike in error rate for the runs between 8 and 11 NOPs, with a large peak at 8 NOPs. While there is some variability in data obtained in different runs, we found the 8 NOPs virus to repeatedly exhibit larger error counts. Note that as the number of NOPs in the virus increases, its power goes down, putting less pressure on the power delivery network. As a result, we would expect the error count to remain constant or decrease with the number of NOPs. The fact that the error rate spikes for the NOP-8 virus (and is low or zero for lower NOP counts) indicates that it is very likely oscillating close to the chip's resonance frequency [14], [19], [28], which leads to a larger droop and higher error rate.

We expand the same experiment to examine if the behavior is consistent across multiple voltage levels. Figure 16 shows the error rate as a function of $V_{dd}$ on the main core for three different workloads running on the auxiliary core. *Aux. Load NOP-8* is the voltage virus with 8 NOPs (worse case droop in the previous experiment). *Aux. Load NOP-0* is the same virus, but without any NOPs. The third run is simply leaving the auxiliary core idle (*No Aux. Load*). We observe that the NOP-8 case exhibits a higher error rate relative to both the idle case and the NOP-0 case throughout the entire voltage range. This is significant because the NOP-0 virus has higher intensity and power demand than the NOP-8 virus, so it should normally exhibit a higher error rate. This is further evidence that that the NOP-8 voltage virus likely exercises the resonance frequency.

This is an important finding for two reasons: first, it shows that correctable errors in cache lines are sufficiently sensitive to capture voltage noise effects, an observation that as far as we know has not been documented before. Second, given that our algorithm uses feedback from these lines

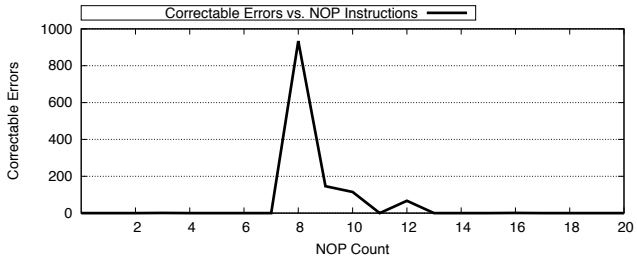



Figure 15. Cache line sensitivity to voltage noise on the main core while running a voltage virus on the auxiliary core.

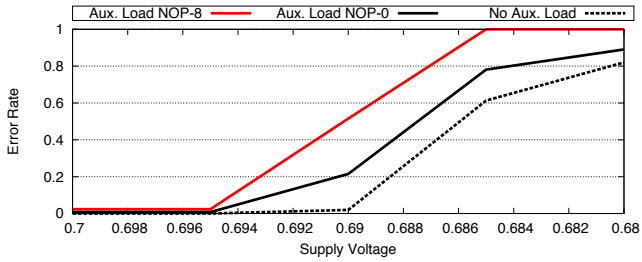



Figure 16. Error rate comparison of the main core with the auxiliary core idle or running different voltage viruses.

to control speculation, our system should be robust under voltage noise. To test this theory, we conducted multiple runs of benchmarks on the main core with the NOP-8 voltage virus on the auxiliary core. All tests completed successfully without crashes or data corruption.

### *E. Characterizing the Source of Errors at Low-Voltage*

A set of experiments were conducted to characterize the nature of the correctable errors triggered during voltage speculation. We ran a test to determine if any retention errors were encountered while self-testing a given cache line. This was achieved by performing a targeted cache line test through the following steps. First, we raised $V_{dd}$ by 80mV above the nominal voltage of 810mV. Once the voltage was raised, data was written into the cache line under test. Writing the data at this high voltage was done to ensure that write operations would complete without any error. The core was then spun in a tight loop while $V_{dd}$ was lowered to a level that has a 100% probability of triggering a correctable error. The core continued to spin at this low voltage for one minute. After that, the voltage was raised to the original 80mV above nominal level and the cache line was read back. We did not observe any correctable errors after applying the aforementioned steps even though the same experiment was repeated multiple times. This indicates that the correctable errors triggered in our system are not memory retention errors, but rather timing errors caused by excessive delay in the memory access logic, or read disturb errors that corrupt the data upon access.

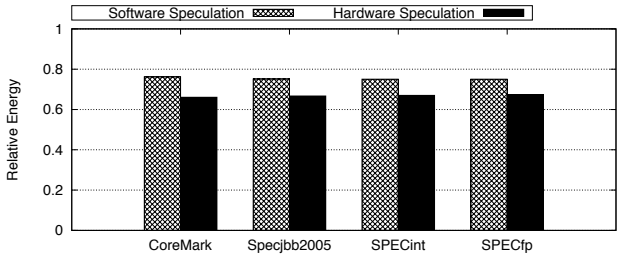



Figure 17. Energy comparison of the hardware and software speculation techniques relative to the low-voltage nominal $V_{dd}$.

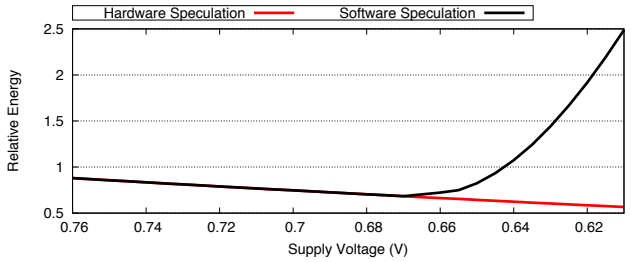



Figure 18. Core energy as a function of $V_{dd}$ for the hardware and software speculation techniques relative to the energy at nominal $V_{dd}$.

### *F. Hardware vs. Software Speculation*

We conducted a set of experiments to compare the energy reduction achieved by our hardware-based speculation to the software-based solution presented in prior work [4]. For this comparison, we run both techniques at low-$V_{dd}$ with the same benchmarks on the same system. Figure 17 shows the energy reduction for the two techniques relative to the low-$V_{dd}$ nominal. We can see that the hardware speculation achieves lower energy than software-based speculation for all benchmark sets. While the software technique reduces energy by 22% on average, the hardware speculation delivers 11% additional energy savings.

There are two primary reasons why the software solution is less efficient. First, it cannot be as aggressive in lowering the voltage because it relies on the workload to exercise weak cache lines. It generally operates at voltage levels at which few or no correctable errors are triggered. The second reason for the higher energy is the performance cost of handling correctable errors in software/firmware rather than hardware.

In the hardware based design, the main source of performance impact lies in the self-test mechanism. However, since access to the cache line under test is performed by the hardware during idle cache cycles, the runtime overhead is negligible. Cache storage is also largely unaffected since only a single cache line is disabled for self-test purposes.

The cost of handling correctable errors in software can also be a significant barrier to more aggressive speculation. At lower voltages, the energy of the software solution can start to increase. This is because the performance overhead goes up rapidly as the number of errors increases. Figure 18 shows the energy of the hardware and software solutions as a function of supply voltage for one core. The energy decreases with voltage for both techniques until they reach 670mV. From that point, correctable errors start to occur and the energy of the two solutions begins to diverge. The energy of the software speculation starts to increase rapidly as the error rate ramps up. The energy of the hardware solution continues to decrease until the minimum safe voltage is reached.

## VI. Related Work

The efficiency of very low voltage designs has been demonstrated in many previous studies [7], [10], [11], [21], [38]. In addition, several improvements geared towards enhancing large cache operation in low voltage through more reliable designs have been proposed [13], [24]. Despite the significant progress in implementing such work into production [32], various challenges remain when considering reliability and high variation.

Runtime reduction of voltage and timing margins has been explored in multiple bodies of work. For example, Razor [12], a well-known technique in this space, employs shadow latches that are running on a delayed clock. Such latches serve the purpose of detecting and recovering from timing errors. This enables their system to aggressively lower voltage. EVAL [30] is another solution that targets improving performance in the context of process variation. It dynamically adapts supply voltage and body bias through machine learning. Other dynamic solutions include the one proposed by Lefurgy et al. [20]. This work entails reducing voltage guardbands by inserting critical path monitors into different units within an IBM POWER7 processor. The system quickly reduces the clock frequency whenever a timing violation is approached. Manageability firmware is then used to adjust the voltage to an appropriate level. Other work by Wang and Calhoun [33] targets the reduction of voltage margins during standby. They employ custom SRAM devices that are designed to prevent data retention failures through the addition of canary cells. Such cells are purposely calibrated to fail at higher voltages to avoid retention failures in the usable SRAM bits.

In previous work [4], we proposed using correctable error reports from ECC-protected on-chip SRAM structures to control a firmware-based voltage speculation system running at nominal $V_{dd}$. The mechanism gradually lowers supply voltage while keeping the processor frequency constant until correctable errors are reported by the ECC logic. That system reduces $V_{dd}$ by 10% on average. However, because it relies on the actual workload to exercise the sensitive memory structures the system is overly conservative with most cores running at safe voltage levels determined during

off-line calibration. In addition, because the error handler involves software, it has a high runtime overhead. This leads to diminishing returns in energy savings if the system triggers a constant stream of correctable errors.

This paper, on the other hand, presents an ECC-based voltage speculation system that is more reliable, precise, and aggressive. Our system is also specifically designed for very low $V_{\text{dd}}$ operation. In addition, we show that error rates in cache lines are sufficiently sensitive to respond to voltage noise effects, an observation that, as far as we know, has not been made before. Overall, this study provides several improvements compared to [4], where we show, on average, an additional 8% in voltage reduction, an additional 12% in power savings, and no performance impact.

Deployment of error-correcting hardware is widespread in modern processors [3], [9], [15], [22], [25], [29] mainly as protection against soft errors. Many novel types of ECC for protecting memory structures have been proposed by prior research [8], [18], [24], [31], [34], [36]. The trend of decrease in the reliability of future CMOS generations is expected to promote more on-chip ECC coverage as the process technology continues to shrink. This will make the deployment of ECC-guided speculation practical to more types of processors from servers to mobile systems.

## VII. Conclusion and Future Work

This paper presented a new technique for using ECC as feedback to aggressively reduce guardbands at low voltages. We show that using targeted self-tests to a single cache line is sufficient to effectively guide voltage speculation for multiple cores without compromising safe operation of the system. Evaluation on real hardware showed significant power savings of 33% across a wide range of applications.

We hope this work will spark interest in using ECC as a mechanism for voltage speculation. This study shows that our ECC feedback is sufficiently sensitive to detect voltage noise conditions, an observation that could have implications on other mechanisms to detect and recover from voltage noise-related issues. We believe this is a promising area for future exploration. We also show that the speculation range varies greatly with the operating voltage and frequency. We would like to explore this variation in speculation range across additional frequencies and examine how it impacts energy usage.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable feedback on this work. We would also like to thank Kristin Barber, Xiang Pan, Naser Sedaghati, Renji Thomas, and Li Zhou from the Computer Architecture Research Lab at OSU for their comments on the camera ready. Special thanks to HP for providing equipment support for this research.

## References

[1] A. Agarwal, B. Paul, S. Mukhopadhyay, and K. Roy, "Process variation in embedded memories: failure analysis and variation aware architecture," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 9, pp. 1804–1814, September 2005.

[2] A. R. Alameldeen, I. Wagner, Z. Chishti, W. Wu, C. Wilkerson, and S.-L. Lu, "Energy-efficient cache design using variable-strength error-correcting codes," in *International Symposium on Computer Architecture (ISCA)*, June 2011, pp. 461–472.

[3] H. Ando, K. Seki, S. Sakashita, M. Aihara, Kan, and K. Imada, "Accelerated testing of a 90nm SPARC64 V microprocessor for neutron SER," *IEEE Workshop on Silicon Errors in Logic - System Effects (SELSE)*, 2007.

[4] A. Bacha and R. Teodorescu, "Dynamic reduction of voltage margins by leveraging on-chip ECC in Itanium II processors," in *International Symposium on Computer Architecture (ISCA)*, June 2013, pp. 297–307.

[5] B. Calhoun and A. Chandrakasan, "A 256-kb 65-nm subthreshold SRAM design for ultra-low-voltage operation," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 3, pp. 680–688, 2007.

[6] A. Chandrakasan, D. Daly, D. Finchelstein, J. Kwong, Y. Ramadass, M. Sinangil, V. Sze, and N. Verma, "Technologies for Ultradynamic Voltage Scaling," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 191–214, February 2010.

[7] L. Chang, D. Frank, R. Montoye, S. Koester, B. Ji, P. Coteus, R. Dennard, and W. Haensch, "Practical strategies for power-efficient computing technologies," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 215–236, February 2010.

[8] Z. Chishti, A. R. Alameldeen, C. Wilkerson, W. Wu, and S.-L. Lu, "Improving cache lifetime reliability at ultra-low voltages," in *International Symposium on Microarchitecture (MICRO)*, December 2009, pp. 89–99.

[9] J. Dorsey, S. Searles, M. Ciraula, S. Johnson, N. Bujanos, D. Wu, M. Braganza, S. Meyers, E. Fang, and R. Kumar, "An integrated quad-core Opteron processor," in *International Solid-State Circuits Conference (ISSCC)*, February 2007, pp. 102–103.

[10] R. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, February 2010.

[11] R. G. Dreslinski, G. K. Chen, T. Mudge, D. Blaauw, D. Sylvester, and K. Flautner, "Reconfigurable energy efficient near threshold cache architectures," in *International Symposium on Microarchitecture (MICRO)*, December 2008, pp. 459–470.

[12] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *International Symposium on Microarchitecture (MICRO)*, December 2003, pp. 7–18.

[13] H. R. Ghasemi, S. Draper, and N. S. Kim, "Low-Voltage On-Chip Cache Architecture Using Heterogeneous Cell Sizes for High-Performance Processors," in *International Symposium on High Performance Computer Architecture (HPCA)*, February 2011, pp. 38–49.

[14] M. S. Gupta, J. Oatley, R. Joseph, G.-Y. Wei, and D. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Design Automation and Test in Europe (DATE)*, April 2007, pp. 624–629.

[15] "Intel Core™ i7 Processor," http://www.intel.com.

[16] "Intel Itanium processor 9500 series reference manual, 2012, revision 0.2," http://www.intel.com.

[17] "Intel Itanium processor 9560 (32M cache, 2.53 GHz)," http://www.intel.com.

[18] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe, "Multi-bit error tolerant caches using two-dimensional error coding," in *International Symposium on Microarchitecture (MICRO)*, December 2007, pp. 197–209.

[19] Y. Kim, L. K. John, S. Pant, S. Manne, M. Schulte, W. L. Bircher, and M. S. S. Govindan, "AUDIT: Stress testing the automatic way," in *International Symposium on Microarchitecture (MICRO)*, December 2012, pp. 212–223.

[20] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, and J. B. Carter, "Active management of timing guardband to save energy in POWER7," in *International Symposium on Microarchitecture (MICRO)*, December 2011, pp. 1–11.

[21] D. Markovic, C. Wang, L. Alarcon, T.-T. Liu, and J. Rabaey, "Ultralow-power design in near-threshold region," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 237–252, February 2010.

[22] R. McGowen, C. A. Poirier, C. Bostak, J. Ignowski, M. Millican, W. H. Parks, and S. Naffziger, "Power and temperature control on a 90-nm Itanium family processor," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 1, pp. 229–237, January 2006.

[23] T. N. Miller, X. Pan, R. Thomas, N. Sedaghati, and R. Teodorescu, "Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips," in *International Symposium on High Performance Computer Architecture (HPCA)*, February 2012, pp. 27–38.

[24] T. N. Miller, R. Thomas, J. Dinan, B. Adcock, and R. Teodorescu, "Parichute: Generalized turbocode-based error correction for near-threshold caches," in *International Symposium on Microarchitecture (MICRO)*, December 2010, pp. 351–362.

[25] J. Mitchell, D. Henderson, and G. Ahrens, "IBM POWER5 processor-based servers: A highly available design for business-critical applications," *IBM Technical Report*, 2006.

[26] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, "Statistical design and optimization of SRAM cell for yield enhancement," in *International Conference on Computer-aided Design (ICCAD)*, 2004, pp. 10–13.

[27] Y. Pan, J. Kong, S. Ozdemir, G. Memik, and S. W. Chung, "Selective wordline voltage boosting for caches to manage yield under process variations," in *Design Automation Conference (DAC)*, 2009, pp. 57–62.

[28] M. D. Powell and T. N. Vijaykumar, "Pipeline muffling and a priori current ramping: architectural techniques to reduce high-frequency inductive noise," in *International Symposium on Low Power Electronics and Design (ISLPED)*, August 2003, pp. 223–228.

[29] R. Riedlinger, R. Bhatia, L. Biro, B. Bowhill, E. Fetzer, P. Gronowski, and T. Grutkowski, "A 32nm 3.1 billion transistor 12-wide-issue Itanium processor for mission-critical servers," in *International Solid-State Circuits Conference (ISSCC)*, February 2011, pp. 84–86.

[30] S. Sarangi, B. Greskamp, A. Tiwari, and J. Torrellas, "EVAL: utilizing processors with variation-induced timing errors," in *International Symposium on Microarchitecture (MICRO)*, November 2008, pp. 423–434.

[31] H. Sun, N. Zheng, and T. Zhang, "Realization of L2 cache defect tolerance using multi-bit ECC," in *Defect and Fault Tolerance of VLSI Systems*, October 2008, pp. 254–262.

[32] S. Vangal, "A solar powered IA core? No way!" Research@Intel, September 2011, http://blogs.intel.com/intellabs/2011/09/15/ntvp/.

[33] J. Wang and B. H. Calhoun, "Canary replica feedback for near-DRV standby Vdd scaling in a 90nm SRAM," in *Custom Integrated Circuits Conference*, September 2007, pp. 29–32.

[34] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar, and S.-L. Lu, "Reducing cache power with low-cost, multi-bit error-correcting codes," in *International Symposium on Computer Architecture (ISCA)*, 2010, pp. 83–93.

[35] C. Wilkerson, H. Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and S.-L. Lu, "Trading off cache capacity for reliability to enable low voltage operation," in *International Symposium on Computer Architecture (ISCA)*, June 2008, pp. 203–214.

[36] D. Yoon and M. Erez, "Memory mapped ECC: Low-cost error protection for last level caches," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3, pp. 116–127, 2009.

[37] B. Zhai, D. Blaauw, D. Sylvester, and S. Hanson, "A Sub-200mV 6T SRAM in 0.13$\mu$m CMOS," in *International Solid-State Circuits Conference (ISSCC)*, February 2007, pp. 332–606.

[38] B. Zhai, R. G. Dreslinski, D. Blaauw, T. Mudge, and D. Sylvester, "Energy efficient near-threshold chip multi-processing," in *International Symposium on Low Power Electronics and Design (ISLPED)*, August 2007, pp. 32–37.