Dynamic Reduction of Voltage Margins by Leveraging On-chip ECC in Itanium II 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

Lowering supply voltage is one of the most effective approaches for improving the energy efficiency of microprocessors. Unfortunately, technology limitations, such as process variability and circuit aging, are forcing microprocessor designers to add larger voltage guardbands to their chips. This makes supply voltage increasingly difficult to scale with technology. This paper presents a new mechanism for dynamically reducing voltage margins while maintaining the chip operating frequency constant. Unlike previous approaches that rely on special hardware to detect and recover from timing violations caused by low-voltage execution, our solution is firmware-based and does not require additional hardware. Instead, it relies on error correction mechanisms already built into modern processors. The system dynamically reduces voltage margins and uses correctable error reports raised by the hardware to identify the lowest, safe operating voltage. The solution adapts to core-to-core variability by tailoring supply voltage to each core's safe operating level. In addition, it exploits variability in workload vulnerability to low voltage execution. The system was prototyped on an HP Integrity Server that uses Intel's Itanium 9560 processors. Evaluation using SPECjbb2005 and SPEC CPU2000 workloads shows core power savings ranging from 18% to 23%, with minimal performance impact.

1. INTRODUCTION

Power consumption is now a primary constraint on microprocessor design spanning the entire spectrum of computing devices, from smartphones to servers. Although semiconductor technology continues to sustain the historical rate of growth in transistor integration, the power efficiency of such transistors is increasing at a much slower rate. Lowering supply voltage (V_{dd}) of chips is one of the most effective techniques for improving their energy efficiency. However, reducing the V_{dd} is becoming increasingly difficult going forward. High variability in transistor parameters coupled with increased sensitivity to fluctuations in the supply voltage are forcing more conservative design choices. To ensure correct execution, high guardbands are added to the supply voltage to account for the worst case behavior of a chip. These guardbands need to consider various factors, such as temperature, circuit aging, process variability and workload characteristics. As a result, V_{dd} margins are often unnecessarily conservative and translate to wasted energy.

A significant body of existing work has explored approaches to dynamically reducing V_{dd} margins by adapting to runtime operating conditions – a technique generally refered-to as "voltage speculation." For example, the well-known Razor [5] design dynamically lowers V_{dd} until occasional timing errors occur. Additional latches running on a delayed clock are used on the vulnerable paths to detect and recover from these errors. More recent work by Lefurgy et al. [15] uses on-chip critical path monitors to detect when a processor is approaching its timing margin as a result of voltage speculation. When that occurs, a control system slows the processor frequency to avoid a timing violation.

This paper presents a new mechanism for dynamically reducing voltage margins and lowering V_{dd} while maintaining the chip operating frequency constant. Unlike previous approaches that rely on dedicated hardware to avoid or recover from timing violations, our solution does not require additional hardware. Instead, it maintains low, but safe operating voltage margins by leveraging error correction (ECC) support already available in modern processors. A key observation made in this work is that, as V_{dd} is lowered, correctable errors in ECC-protected functional units manifest before uncorrectable errors or data corruption. Starting from this observation, we implement a voltage speculation system that uses the rate and type of runtime correctable errors as indicators for finding the lowest safe voltage point at which each core can operate. The solution adapts to on-chip core-to-core variability by tailoring V_{dd} to each core's safe operating level. It also adapts to variability in workload vulnerability to low voltage execution.

The system was prototyped on an HP Integrity Server that uses Intel's Itanium 9560 processors. A control system that dynamically adjusts the V_{dd} of four core pairs in the 8-core chip was implemented in System Firmware. The control system monitors the rate of correctable errors posted by the hardware and makes voltage assignment decisions.

*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.

ISCA '13 Tel-Aviv, Israel

The appropriate voltage level is selected to keep the processor operating close, but still above the safety margin to ensure correct operation. Evaluation using SPECjbb2005 and SPEC CPU2000 shows core power savings ranging from 18% to 23%, with minimal performance impact. The paper also illustrates the effects of process variability on core-to-core sensitivity to low-voltage operation.

Overall, this paper makes the following contributions:

- Presents a novel firmware-based technique for voltage speculation.
- Makes the observation that on-chip correctable errors can be used as a proxy for estimating timing margins.
- Evaluates a prototype implementation of the voltage speculation system on an HP server using Intel's Itanium 9560 processors.
- Characterizes the impact of process variation on core-to-core distribution of timing margins using measurements on real processors.

The rest of this paper is organized as follows: Section 2 outlines the motivation for this work and examines some experimental data. Section 3 details the design and algorithms for the proposed ECC-based voltage speculation system. Section 4 presents a prototype implementation of the proposed system. Sections 5 and 6 present an experimental evaluation. Section 7 details related work; and Section 8 concludes.

2. MOTIVATION

Supply voltage levels in modern microprocessors have to account for multiple sources of uncertainty in the design process and runtime environment. Significant challenges in manufacturing chips with low nanometer critical dimensions lead to high variability in circuit properties, such as speed and power consumption. Other sources of uncertainty are related to runtime effects and include temperature variation, circuit aging, etc. To account for these factors, microprocessor designers add large guardbands to the $V_{\rm dd}$. These guardbands, which can be as high as 20% [13, 22] in modern processors, make microprocessors less energy efficient.

In order to quantify voltage margin levels in a modern microprocessor, experiments were conducted on two Intel Itanium II 9560 8-core processors [24] running on an HP Integrity BL860c-i4 server. More details about the evaluation platform and methodology are presented in Section 5. The frequency of the chips was set at their nominal values according to the product specifications. The $V_{\rm dd}$ was gradually lowered for each core, while running a stress test workload until system crashes or data corruption occur. The lowest safe voltage at which each core runs reliably is recorded. The results showed that all cores run reliably at voltages that are significantly lower than nominal values – 12% lower, on average. In addition, design-identical cores within the same chip have different minimum safe voltages. This is most likely due to the impact of process variability on circuit delay distribution within the chip. Figure 1 shows the nominal and lowest safe voltages (*Safe/Min*) for each core in one processor. The safe $V_{\rm dd}$ ranges between 0.95V and 1V vs. the 1.1V nominal. This core-level variability outlines the potential for core-level voltage adaptation in systems that have the capability to regulate and deliver multiple supply voltages within the chip.

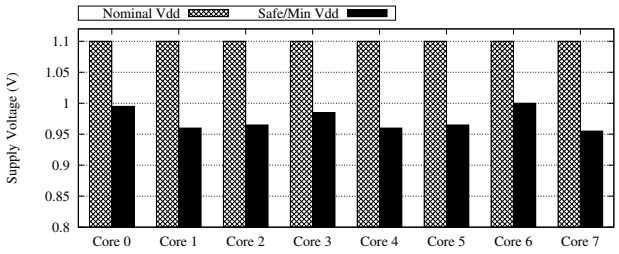


Figure 1: (a) *Nominal* and *Safe/Min* $V_{\rm dd}$s for 8 cores of an Intel Itanium II processor.

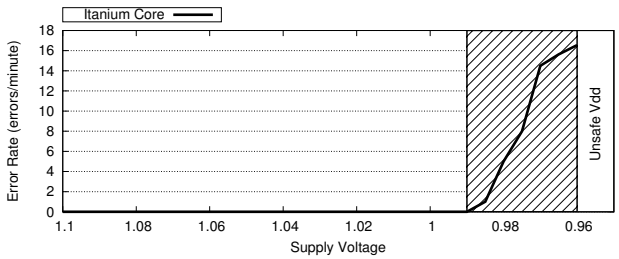


Figure 2: Correctable error rate as a function of $V_{\rm dd}$.

In the same experiment, we record reports of correctable errors flagged by the hardware. These correctable errors are benign events that do not trigger exceptions or affect application execution in any way. They are single-bit errors that occur in ECC-protected functional units. The ECC hardware corrects these errors and logs them for debugging purposes. At nominal $V_{\rm dd}$, correctable errors are very rare events. As the $V_{\rm dd}$ is lowered, timing margins become tighter and correctable errors occur with higher probability. Figure 2 shows the correctable error rate as a function of $V_{\rm dd}$ for one core. No errors are observed as the $V_{\rm dd}$ is dropped by more than 10% from 1.1V to below 1V. When the supply voltage reaches 0.99V, correctable errors start to occur. The correctable error rate increases gradually from about 1 error/minute to about 16 errors/minute as the $V_{\rm dd}$ is lowered further from 0.99V to 0.96V (shaded region in Figure 2). When the $V_{\rm dd}$ of the core is lowered below 0.96V, the system no longer operates reliably.

We observe that a significant voltage range exists above the failure $V_{\rm dd}$ at which correctable errors are frequent; but no failures or data corruption occur. In other words, as the voltage is lowered, cores always exhibit correctable errors before reaching the failure $V_{\rm dd}$. This behavior is consistent across all the cores and chips we tested when running the stress test workload. This indicates that correctable errors could be used as a predictor for approaching timing margins during voltage speculation.

3. ECC-BASED VOLTAGE SPECULATION

We present a new voltage speculation system that dynamically lowers $V_{\rm dd}$ and uses correctable error reports to ensure cores do not reach unsafe operating levels. In general, correctable error handling, as implemented in current processors, is invisible to the Operating System (OS) and the applications running on the system. Our system taps into correctable error logs by configuring the hardware to report correctable errors to a firmware layer that imple-

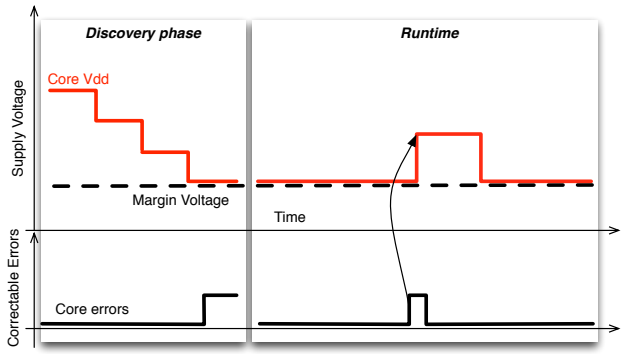


Figure 3: Illustration of the voltage margin discovery process. V_{dd} is gradually lowered until the first correctable errors are encountered. At runtime, V_{dd} can be raised above the margin voltage.

ments our monitoring functions. Implementing the error monitoring system in firmware as opposed to hardware or the OS has multiple advantages. Firmware has a faster response time and lower performance overhead compared to software solutions implemented at the OS level. In addition, the firmware-based voltage speculation system is transparent to the OS and does not require OS intervention or support. This makes the solution easily deployable and backward compatible. Finally, a firmware implementation can perform more sophisticated power management compared to what a hardware implementation can achieve. Our solution has two main components: an error monitoring system for identifying and logging correctable errors and a voltage speculation governor for dynamically controlling V_{dd}.

3.1 Margin Voltage Discovery

We empirically determine that, for most cores and applications, the highest V_{dd} at which correctable errors occur is safe and does not lead to system crashes or data corruption. We refer to this V_{dd} as the "margin voltage." The value of the margin voltage is different for each core because of process variation. In addition, this value can change at runtime in response to changing operating conditions. The margin voltage is used as a reference level by the voltage speculation algorithms when deciding the current operating V_{dd}.

The margin voltage for each core is determined through an online training/discovery phase run in firmware at system boot time. Figure 3 illustrates this process. During the discovery phase, the voltage of each core (or core-pair) is progressively lowered while running a stress test application. The stress test application is designed to exercise multiple critical paths within the core. When the first correctable errors are encountered, the margin voltage is computed and recorded in firmware for that core. At runtime, the firmware continues to monitor the processor for correctable errors. The correctable error rate can change continuously as a result of changes in dynamic conditions such as temperature, aging or workloads.

3.2 Aggressive vs. Conservative Cores

Because of process variation, cores react differently to low-voltage operation. Some cores exhibit a gradual increase in the correctable error rate for several voltage steps before a crash occurs. For these cores, which we call "aggressive," the distance between the failure voltage and the voltage margin is fairly large. For other cores, which we call "conservative," the failure voltage is much closer to the margin voltage. Figure 4 shows the error rate versus V_{dd} for an aggressive and a conservative core.

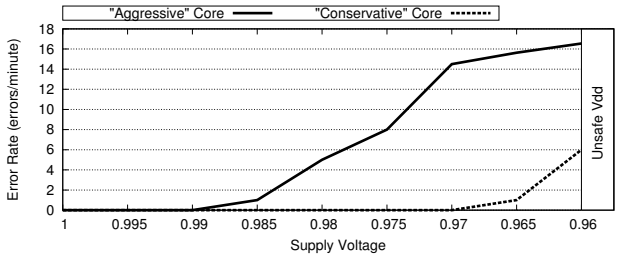


Figure 4: Correctable error rate as a function of V_{dd} for an aggressive and a conservative core.

The reason aggressive cores exhibit a more graceful degradation while conservative cores reach their failure voltage soon after exhibiting their first correctable errors is likely related to the type, number and distribution of critical paths that are slowed by variation in each core category. In aggressive cores, affected critical paths are likely better protected by ECC. This explains why errors are raised at voltages that are significantly higher than the failure voltage. Conservative cores, on the other hand, have slow critical paths that are less protected. This hypothesis is also supported by the types of correctable errors that each core class predominantly exhibits. Aggressive cores tend to trigger correctable cache errors while conservative cores mostly trigger correctable register file errors. Since the cache is larger and has more critical paths, it is likely to trigger more correctable errors when slowed down by variation compared to the register file.

Our system takes advantage of the more predictable behavior of aggressive cores in order to extract additional power savings. For these cores, a more aggressive (but still safe) voltage speculation algorithm is used to lower the supply voltage below the margin voltage.

3.3 Dynamic Voltage Speculation

A firmware-based Voltage Speculation Governor is responsible for implementing the dynamic voltage speculation algorithms at core and core-pair granularities. The governor receives input from the error monitoring system and reacts to information about error rates according to predefined algorithms. The governor is also responsible for coordinating the margin voltage discovery phase. Aggressive and conservative cores are handled differently by the voltage speculation algorithms.

3.3.1 Conservative Speculation

Conservative cores are more vulnerable to low-voltage operation. To ensure correct execution, we add a small safety padding (10mV in our experiments) when computing their margin voltage. Conservative cores are never allowed to run below the margin voltage. The V_{dd} can, however, be raised temporarily above the margin voltage if correctable errors occur, as illustrated in Figure 3. If a conservative core encounters a correctable error, the governor raises its

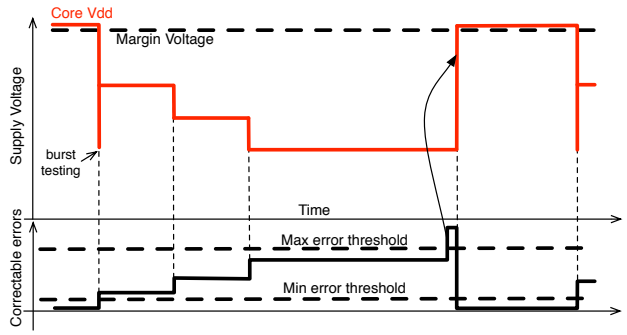


Figure 5: Diagram of voltage speculation algorithm for aggressive cores.

V_{dd} in increments of 10 mV per correctable error. The voltage continues to be increased until no correctable errors are observed. If no error is observed for a while, the governor attempts to lower the V_{dd} again in 5 mV decrements after every minute. Decrementing the voltage continues until the margin voltage is reached again. If correctable errors occur repeatedly at the margin voltage, the margin can be raised permanently.

3.3.2 *Aggressive Speculation*

Aggressive cores can, in many cases, operate below the margin voltage. As a result, they are expected to periodically encounter correctable errors. In order to ensure reliable operation, the correctable error rate has to be maintained below an empirically determined threshold. The Voltage Speculation Governor is responsible for discovering and maintaining that safe V_{dd} at runtime. This process is illustrated in Figure 5. The discovery process begins with the core at the margin voltage. The Voltage Speculation Governor gradually lowers V_{dd} below the margin as long as the error rate remains below the Max error threshold. In our implementation, that threshold is 1 error per minute. Once that threshold is reached, the governor will maintain a constant V_{dd} and continue to monitor the error rate.

Some workloads do not exercise critical paths that expose correctable errors. Aggressive speculation is not appropriate for such applications. This is because, without the feedback obtained from the correctable error rate, the governor cannot safely search for the safe operating point. Therefore, the governor has to make a determination as to whether an application is suitable for aggressive treatment or not. This is achieved by performing "burst testing" (Figure 5) during which the voltage is dropped from the margin by 20mV. The voltage rail is modulated over a 6 second period (V_{dd} below safety for 5 seconds followed by V_{dd} back to safety for 1 second) with up to 5 retry cycles. If a correctable error is triggered during burst testing, the core enters aggressive mode and the V_{dd} is lowered below the margin. If correctable errors are not observed during burst testing, aggressive speculation is disabled for the next interval and the core continues to run at the margin voltage. The potential for aggressive speculation is evaluated once every minute.

Aggressive speculation ends if any of the following events occur: (1) correctable error rate exceeds the Max error threshold indicating approaching timing margins, (2) correctable error rate is below the Min error threshold indicating the workload is no longer suitable for aggressive speculation, or (3) a context switch is signaled by the OS indicating the need to reevaluate suitability for aggressive speculation.

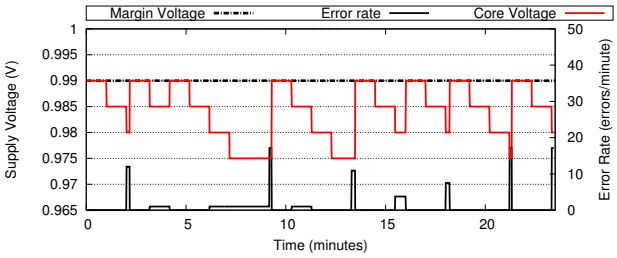


Figure 6: The effects of the voltage speculation algorithm on V_{dd} while running the stress test application on an aggressive core.

Core mix	Speculation	Margin Voltage
All conservative	Conservative	MAX(All Margin V_{dd}s)
Some conservative	Conservative	MAX(All Margin V_{dd}s)
All aggressive	Aggressive	MAX(All Margin V_{dd}s)

Table 1: Speculation classification and margin voltage selection based on the type of cores sharing a common voltage line within a core cluster.

To illustrate the behavior of the voltage speculation algorithm, Figure 6 shows a 25 minute trace of V_{dd} and the correctable error rate for an aggressive core running the stress test workload. We can see that the algorithm progressively lowers supply voltage until the correctable error rate reaches its target. It then keeps the V_{dd} at this level until the number of correctable errors increases above its Max error threshold. When that happens, the voltage is immediately raised to the margin voltage. We can see that occur around minutes 2, 9, 14, 17 and 22.

3.3.3 *Speculation in Core Clusters*

Most processors do not currently support V_{dd} assignment at core granularity. In some cases, all cores are assigned to a single V_{dd} domain, while in others, V_{dd} lines are shared by clusters of cores. For instance, in the Itanium II processor used in our experiments, V_{dd} lines are shared by core pairs. When multiple cores share a voltage domain, voltage speculation is performed at domain granularity. The characteristics of all cores in a domain have to be considered when choosing the margin voltage or the aggressiveness of the speculation. Table 1 summarizes the speculation options as a function of the core mix. If a cluster has at least one conservative core, speculation for that cluster has to be conservative. If all cores in a cluster are aggressive, speculation for the cluster can be aggressive. The margin voltage for the cluster is set to the highest of the core margins.

4. PROTOTYPE IMPLEMENTATION

The proposed voltage speculation system was prototyped on an HP Integrity Server that uses Intel's Itanium 9560 processors. Figure 7 depicts the overall architecture of the prototype. It outlines the main components of the System Firmware: the Processor Abstraction Layer (PAL), the System Abstraction Layer (SAL) and how our solution integrates into the existing framework. Our prototype integrates

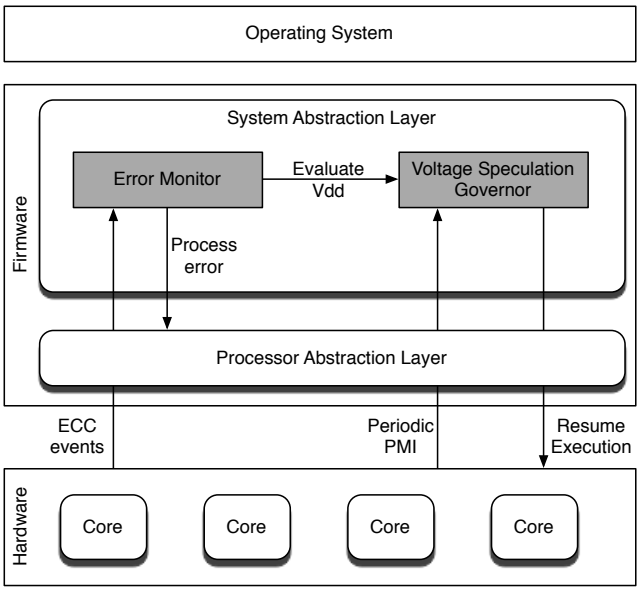


Figure 7: Main components of the firmware-based implementation of our voltage speculation system in the Intel Itanium II processor. The components we added to the system firmware are highlighted in gray.

primarily into the SAL layer; but modifications outside of SAL are needed to facilitate the evaluation of our solution.

The Processor Abstraction Layer (PAL) encapsulates interfaces to various processor specific functions made available to the System Firmware modules for consumption during runtime. These include power management, error record extraction and virtualization to name a few [9]. In addition to PAL serving the role of abstracting common features through procedure calls, it provides hooks for communicating with the rest of System Firmware via architected entry-points. Access to entry-points is needed in order to transfer control between PAL and other System Firmware entities. Transfer of control between PAL and other entities takes place upon the occurrence of hardware events, such as cache errors and resets. Our solution mainly uses PAL calls for extracting and clearing error information generated by the processor hardware.

The second level of System Firmware in this design is the System Abstraction Layer (SAL). SAL serves the purpose of initializing, configuring and testing the platform's hardware resources during the initial boot phase. This is done in preparation for handoff to the OS. SAL also plays the role of providing the interface to the OS. Since the Operating System doesn't have enough knowledge about the specifics of a given platform, it relies on SAL to perform certain actions on its behalf. Such actions include the gathering of machine state information in response to correctable cache and register file error events, as well as the re-initialization of processors [12]. The Error Monitor and Voltage Speculation Governor modules of our prototype are implemented at the SAL level.

4.1 Error Monitor for Margin Detection

The Error Monitor is responsible for capturing correctable error activity from the processor cores. This component is implemented in the SAL layer of the System Firmware. Normally, processor correctable errors are not immediately made visible to the firmware. This is because hardware has the ability to correct them without software intervention. The firmware code is therefore modified to expose these errors to the Error Monitor. The processor is configured to always hand off to System Firmware whenever correctable errors occur. Upon the detection of a correctable error, the faulted core vectors to an error handler. This is shown as an "ECC event" in Figure 7. The handler begins by saving the necessary processor state information that allows the core to seamlessly resume normal execution when the firmware interrupt completes. The handler then proceeds to set up the execution resources needed to run the high-level code of the Error Monitor (C-based code). This way, firmware can make stacked procedure calls without clobbering what the OS was using prior to the interrupt.

Once inside the Monitor, the core calls into PAL to extract the error record. The error record is decoded to determine the type of error that occurred on the core. The Error Monitor can distinguish between multiple error types such as cache, register-file, TLB and microarchitecture related errors. It can also differentiate between instruction cache errors, data cache errors and the corresponding levels at which they occur. This information is not currently used by the voltage speculation algorithm, but could prove useful in future work. The extracted error is timestamped; and the error rate information is updated in a shared repository.

The Error Monitor treats error logging differently for aggressive cores compared to conservative ones. A correctable error flagged by a conservative core will trigger a call to the Voltage Speculation Monitor. This is because all correctable errors lead to raising the voltage in conservative cores. For aggressive cores, the Error Monitor is responsible for updating the error rate data and timestamp information for every correctable error. The Voltage Speculation Governor will not be invoked on every ECC event. The Governor will instead periodically examine the error rate reported by the Error Monitor and take appropriate action. Finally, the Monitor handler clears the error from the processor and resumes the interrupted core to normal execution.

4.2 Voltage Speculation Governor

The Voltage Speculation Governor is responsible for making dynamic voltage adaptation decisions based on feedback from the Error Monitor. At runtime, the governor evaluates and adjusts the operating voltage periodically (once a minute in our experiments). This is achieved by setting up the hardware to periodically generate a special interrupt called a Platform Management Interrupt (PMI) that is handled by System Firmware rather than by the OS. This mechanism prompts the interrupted core to enter the PMl handler where the Voltage Speculation Governor code runs. The OS has no knowledge that this is taking place.

The Governor performs multiple tasks. For aggressive cores, it verifies that the current error rate is between the Min and Max thresholds. If it is either below Min or above Max, it will restore the voltage to a safer level. If, on the other hand, the error rate is significantly below the Max threshold, the Governor will attempt to lower V_{dd} by another increment. For conservative cores the V_{dd} is raised by an increment for each correctable error flagged. Once the Governor determines the appropriate action to take, it writes

Processor Architecture	
Type	Itanium II 9560
Cores	8, out-of-order
Frequency	2.53GHz nominal
Nominal V_{dd}	1.1V
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
Other	
Memory	DDR3 32GB
Disk Drive	SAS 76GB
Operating System	HP-UX 11i v3
Server	HP BL860c-i4 blade
Independent voltage domains	6

Table 2: Architectural and system details of the BL860-i4 Integrity Server and Itanium 9560 processor used in the evaluation [10, 11].

Structure	ECC Type
Integer Register-file	SECDED
Floating-point Register-file	SECDED
L2 instruction cache (data)	SECDED
L2 instruction cache (tags)	SECDED
L2 data cache (data)	SECDED

Table 3: Summary of the ECC protection in the Intel Itanium 9560 processor core [23].

to the necessary registers to initiate the power management action. When the requested voltage setting is reached, the Governor releases the core back to the OS to resume execution.

5. EVALUATION METHODOLOGY

5.1 System Architecture

A BL860c-i4 Integrity Server from HP was used for prototyping and evaluation purposes. The server was equipped with two Intel Itanium 9560 processors, each possessing eight cores with hyperthreading. The server was provisioned with 32GB of DDR3 memory and a 73GB SAS drive. The system ran the HP-UX Operating System. Table 2 lists additional detailed information about the evaluation system, including the processor's architectural features, power and frequency specifications, technology, etc.

According to the available documentation [23], the Itanium 9560 processor has several functional blocks protected by ECC. These are listed in Table 3.

5.2 Experimental Framework

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 and minimized interference with the workloads under test. However, in order to facilitate such retention of hardware 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 and temperature in addition to the error rate information. System Firmware relied on mechanisms defined in ACPI to claim ownership of a core for evaluation purposes. Changes in the UEFI component were restricted to the system's memory map. System Firmware used UEFI interfaces to mark a set of memory pages for private use in order to log the aforementioned data. Marking these regions was necessary to prevent them from being reclaimed by the OS.

Power consumption information was collected via System Firmware by sampling a set of processor registers. We collect the power information for each core pair in addition to the uncore component. We also log 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 correctable 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 [24]. The supply voltage of each of these power lines can be independently modulated. The voltage speculation module in System Firmware controls this modulation. In order to accommodate the testing of multiple configurations without rebuilding the firmware, an NVRAM based configuration file is used. Based on this information, the governor modulates the appropriate voltage rails according to our algorithm.

Experiments that examined the sensitivity of each core in response to low voltage were conducted by exercising a single core at a time. The core pair that shares a supply line with the one under evaluation was left idle. This allowed the collection of data at core granularity even with core pairs sharing voltage rails.

5.3 Benchmarks

Two benchmark suites were used in the evaluation: SPEC CPU2000 and SPECjbb2005. SPECjbb2005 was configured to run a total of 8 warehouses that were launched on both threads of each core under test. For SPEC CPU2000, a full instance of each benchmark was launched on each thread of all cores under test. We ran all benchmarks from CPU2000 (listed in Table 4) except for *wupwise* and *apsi*, which we could not successfully run on this system. 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

6. EVALUATION

This section examines the power and energy savings from our dynamic voltage speculation system as well as its impact on system performance. We begin by characterizing the process variation effects on voltage margins and the types of errors triggered at low V_{dd}.

Suite	Benchmark
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 4: Applications and benchmarks used in the evaluation.

6.1 Process Variation Effects

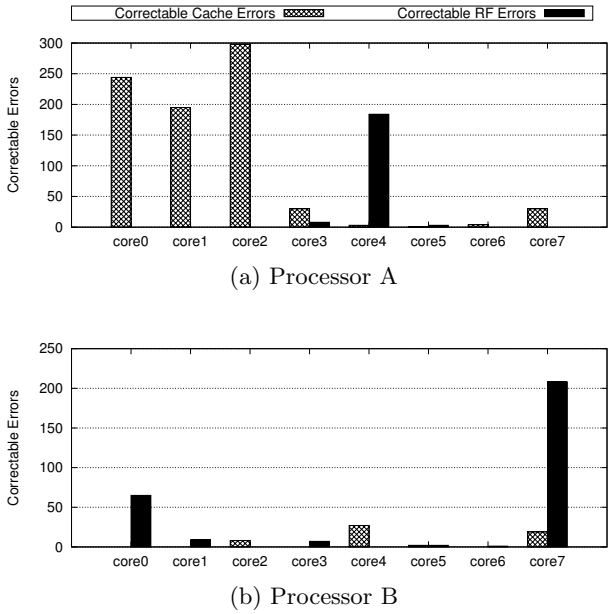
In order to characterize the effects of process variation on voltage margins, we run the stress test application on each core while progressively lowering V_{dd}. We record the lowest supply voltage at which the stress test application runs successfully for at least 20 minutes with no crashes or data corruption. The distribution of minimum safe V_{dd} was shown in Figure 1 and discussed in Section 2. In the same experiment, we logged all correctable errors reported by the hardware at the *Safe/Min* V_{dd}.

Figure 8 shows the distribution of correctable errors for each core for the two Itanium II processors in our system. Both processors exhibit large core-to-core variability in the type and rate of correctable errors. In processor A (Figure 8a), cores 0,1 and 2 trigger a large number of correctable cache errors while core 4 exhibits a large number of correctable register file errors. Processor B (Figure 8b) shows similar high variability, but with a different distribution of error rates and types. Processor B triggers fewer cache errors and slightly more correctable register file errors. The high variability in behavior is especially interesting given that the tests are conducted consistently on all cores, running the same exact workload under the same conditions. Moreover, the error rates and distributions are largely reproducible over multiple, identical runs. Within-die process variability is very likely the root cause of the observed error distributions.

In general, we observed that correctable cache errors have a more graceful onset than register file errors. As a result, correctable cache errors are a more reliable predictor for aggressive cores. The presence of correctable register file errors, on the other hand, is an indication that the core's execution pipeline is in the critical path. As a result, these cores are less tolerant of aggressive voltage speculation.

6.2 Dynamic Adaptation to Workload

The Voltage Speculation Governor continuously adjusts the supply voltage to ensure reliable operation. For aggressive cores, the governor attempts to lower the supply voltage below the margin voltage, as long as the rate of correctable errors is maintained at a targeted level. Figure 9 shows a trace of the supply voltage over time for the SPECjbb workload. The correctable error rate for the same interval is also shown. The V_{dd} is initially set at the margin voltage. The Voltage Speculation Governor lowers the voltage in 5mV decrements every minute of operation. This continues as long as the core exhibits an error rate of 1 correctable error per minute. The voltage is immediately raised back to the safety voltage when one of two events occurs: (1) the error



(a) Processor A

(b) Processor B

Figure 8: Distribution of correctable error rates and error types over a 20 minute run of the stress test application at the *Safe/Min* voltage, for two Itanium 8-core processors.

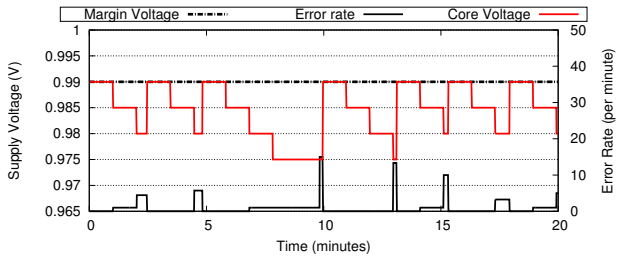


Figure 9: Dynamic adaptation of supply voltage to runtime conditions in SPECjbb running on an aggressive core.

rate increases above 1 error per minute or (2) no correctable errors are triggered over the previous interval. The parameters for the voltage speculation algorithm are empirically validated and conservative to ensure reliable operation.

Voltage speculation can also be jointly applied to core pairs when they share a voltage line. Aggressive speculation can be used if both cores and applications satisfy the requirements for aggressive operation. Figure 10 shows V_{dd} for a pair of aggressive cores, each running an instance of the stress test application. The core pair follows the aggressive speculation algorithm and lowers V_{dd} below the margin. At some points in the execution, however, the Governor has to raise the V_{dd} above the margin voltage. This happens when the combined correctable error rate for the two cores exceeds the Max threshold. The V_{dd} is raised in 10mV increments until the correctable error rate falls below the threshold.

6.3 Power Savings

Voltage Speculation lowers the supply voltage by an average of 9-11% across all the benchmarks we test, as shown in Figure 11. The data is collected on both Processors A and B. For processor A, we identify cores 0,1,2 and 6 as

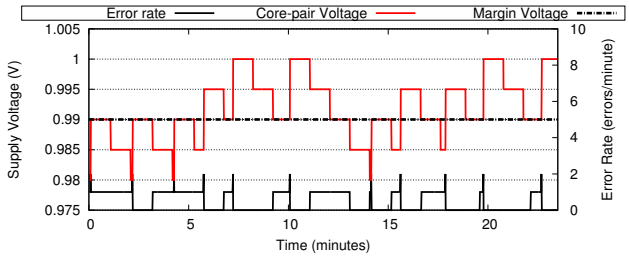


Figure 10: Dynamic adaptation of V_{dd} for a core pair running the stress test workload with aggressive speculation.

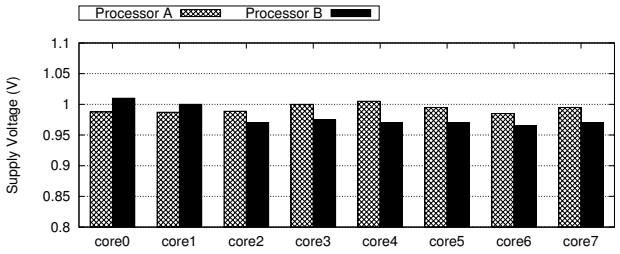


Figure 11: Average supply voltage for all cores of processors A and B, across all benchmark sets.

aggressive and 3,4,5 and 7 as conservative.

Not all applications benefit equally from running on aggressive cores. Figure 12 shows the average supply voltage for each core and each benchmark set while running on processor A. SPECjbb benefits most from running on the aggressive cores 1,2 and 3, achieving lower average V_{dd}s than SPECint and SPECfp. This is because SPECjbb exercises execution paths that trigger the constant stream of correctable errors necessary for aggressive mode speculation. We do not see the same benefit on core 6, even though it is aggressive. This is because core 6 has a relatively small window for further voltage speculation below the margin. Many of the SPECint and SPECfp applications do not exercise paths that trigger a sufficient stream of correctable errors. As a result, they do not see the same V_{dd} reduction as SPECjbb even when running on aggressive cores.

Conservative cores see less dynamic adaptation across applications, spending most of their execution time at the margin voltage. Their V_{dd} is only changed when they encounter a correctable error, which is a rare event at the margin voltage in all benchmarks.

The large reduction in average V_{dd} for all cores results in significant power savings for the processor. Figure 13 shows the average power consumption (geometric mean) for each benchmark suite, relative to the power consumed while running at the nominal V_{dd} of 1.1V. Overall, power consumption of the core-only section of the processor is reduced by an average of 22% for SPECjbb, 23% for SPECint and 18% for SPECfp. The so-called "uncore" section of the chip, which in the Itanium includes the L3 cache and memory controllers, cannot be voltage-scaled in this system. Therefore, the uncore fraction of the total chip power consumption is constant in both the nominal and margin cases. As a result, the chip-wide power savings, which include both the core and the uncore sections, are slightly lower at 14% for SPECjbb, 15% for SPECint and 11% for SPECfp.

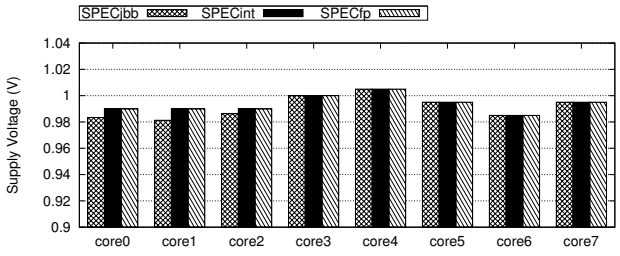


Figure 12: Average supply voltage for each core and each benchmark set while running on Processor A.

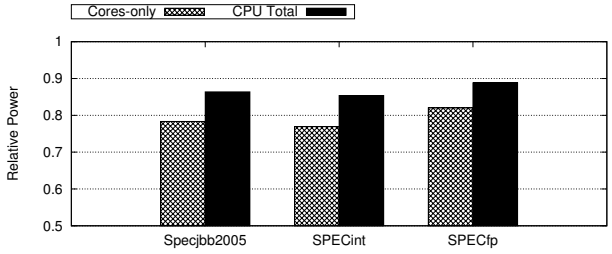


Figure 13: Average power consumption with voltage speculation relative to nominal V_{dd} for the cores-only fraction of the chip and for the entire CPU.

6.4 Runtime Overhead and Energy

There are two sources of runtime overhead in our system: the cost of handling correctable error events raised by the hardware and the cost of periodically running the voltage speculation algorithms.

The current version of our prototype implementation was mainly geared towards collecting profiling information for the purposes of this study. It was not optimized for low overhead. We approximated the cost of running the Error Monitor and Voltage Speculation Governor handler code at about 260 ms. We traced the majority of the cost to be concentrated in displaying live events of correctable errors as they occur. Sending events over I/O for display and performing the necessary handshaking between System Firmware and the hardware consumes several milliseconds. This overhead can be significantly reduced in a production implementation by eliminating some of the logging support and optimizing the code.

Figure 14 shows the performance overhead of the voltage speculation system for each core and each set of benchmarks. The performance overhead is relative to the baseline system without voltage speculation. Even with the unoptimized prototype implementation, the overall impact of this overhead is minimal, averaging at less than 1% across all cores and all benchmarks. The overhead is slightly higher at 3-4% for the aggressive cores (0,1 and 2) when running SPECjbb. The increased overhead is due to the cost of handling correctable errors which are more numerous in SPECjbb. SPECint and SPECfp applications are generally treated more conservatively by the Voltage Speculation Governor. This is because they tend to not exercise execution paths that trigger a sufficient rate of correctable errors to make them good candidates for aggressive speculation.

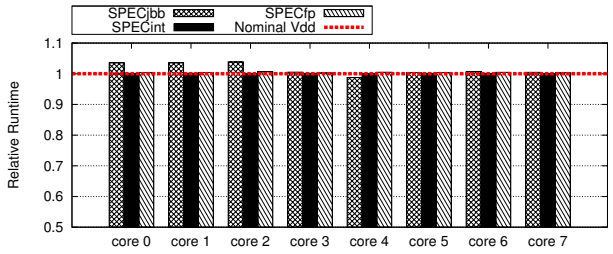


Figure 14: Performance overhead of the voltage speculation governor.

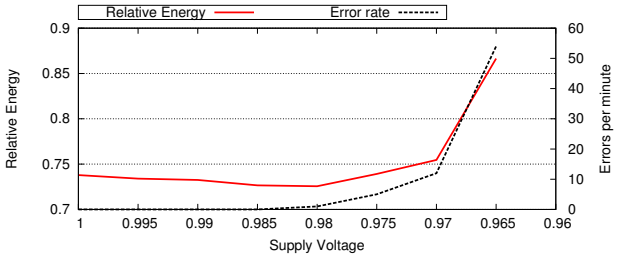


Figure 15: Core energy as a function of supply voltage and the number of correctable errors while running *gcc*.

6.5 Energy vs. Supply Voltage Sensitivity

In order to better understand the impact of the correctable error rate of an application on runtime and energy, we conducted a sensitivity study that swept multiple V_{dd} levels and recorded error rates and energy. Figure 15 shows the results from running *gcc* from SPECint on core 0 with the V_{dd} swept from 1V to 0.96V. The energy values are relative to the same benchmark running at the nominal V_{dd}. The correctable error rate is also shown for each V_{dd} level.

Between 1V and 0.985V, the benchmark completes without triggering any correctable errors. As a result, the energy decreases since the power consumption decreases at each step. Beyond 0.98V the benchmark triggers correctable errors and their number increases rapidly with lower voltages. The overhead of handling these errors also goes up, leading to an increase in runtime and energy even though power consumption continues to decrease. The minimum energy point is 0.98V which corresponds to an error rate of roughly one error per minute. We use this error rate as a target for the voltage speculation algorithm used by the aggressive cores.

Figure 16 shows the same experiment for the stress test application. This application is designed to stress the system and it therefore triggers a larger number of correctable errors at the same voltages: at 0.98V we see 23 correctable errors vs. just 1 for *gcc*. For the stress test application, the optimal energy point is 0.995V which corresponds to a region of operation that triggers no correctable errors.

6.6 Voltage Speculation at Lower Frequencies

We expect the voltage speculation system we developed to scale well to frequencies that are lower than the nominal system frequency. Our system can be used in combination with a dynamic frequency scaling solution to match the optimal supply voltage to the desired frequency. Figure 17 shows the potential for voltage margin reductions on Processor B running at 2.13Ghz compared to the nominal frequency of

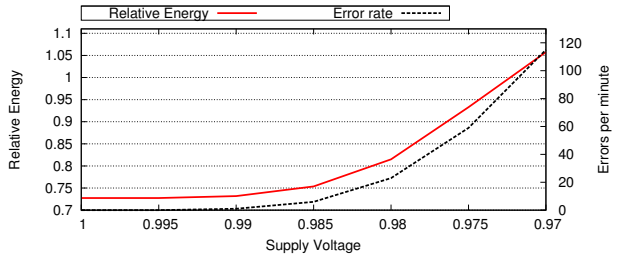


Figure 16: Core energy as a function of supply voltage and the number of correctable errors while running the stress test application.

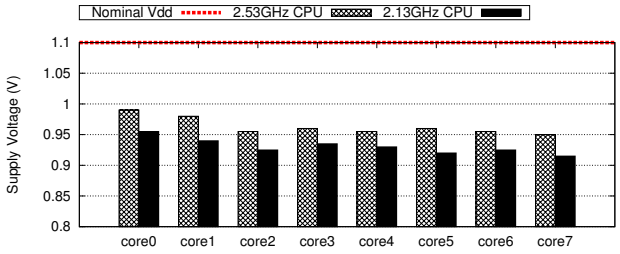


Figure 17: Lowest safe supply voltage for each core of Processor B at two different frequencies: 2.53GHz and 2.13GHZ.

2.53GHz. We can see that roughly the same variation pattern applies at both frequencies. The average voltage at 2.13GHz is 0.931V, which is 33mV lower than the average margin voltage at the nominal frequency. This shows significant potential for further power savings at lower frequencies through a combination of dynamic frequency scaling and voltage speculation.

6.7 Voltage Speculation Robustness

We conducted dozens of hours of testing of the voltage speculation algorithm presented in this paper. The algorithm ran reliably and consistently under multiple workloads and multiple load levels from single-core workloads to a fully-loaded system. One of the stress experiments we conducted ran the stress test application continuously for 24 hours on core-pair 0, which includes cores 0 and 1 of processor A. The voltage and error trace is shown in Figure 18. Supply voltage varies between 0.975V and 1.005V over this run, which means that the cores run both below and above the margin V_{dd} of 0.99V.

Additional experiments were conducted under elevated temperature conditions by slowing down the speed of the server fans. There was no observable difference in error rates or system behavior. All tests completed successfully.

7. RELATED WORK

Dynamic Margins Reduction. Reducing voltage and timing margins at runtime has been explored in multiple bodies of work. Well-known techniques such as Razor [5] use shadow latches on a delayed clock to catch timing violations. This allows the system to reduce voltage aggressively to save power. Techniques like Paceline [6] use checker cores, similar to those proposed by Austin [29], to reduce timing margins and improve performance. Other dynamic solutions for voltage adaptation include EVAL [25], which uses on-line

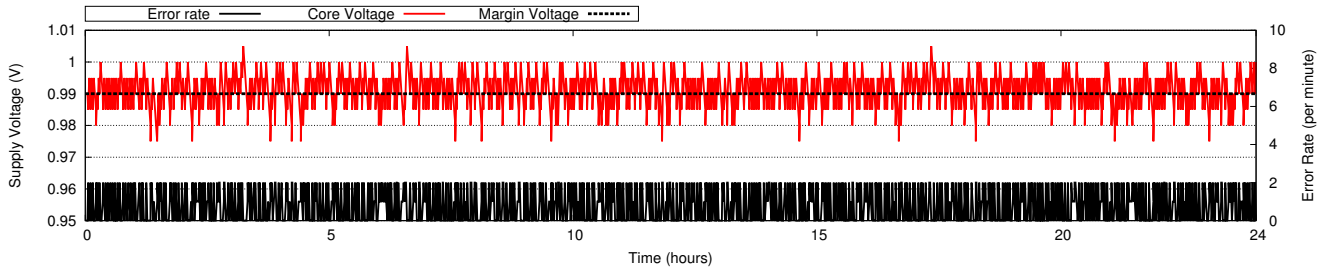


Figure 18: Trace of V_{dd} and error rate for a 24 hour run of the stress test application on a core-pair.

adaptation of supply voltage and body bias, controlled by a machine learning algorithm. EVAL is targeted at timing errors and improving performance in the face of process variation. More recently, Lefurgy et al. [15] developed a solution for lowering voltage margins in IBM Power 7 processors. Their solution uses critical path monitors built into the chip to detect when margins are about to be exceeded. This paper presents a solution that relies on resiliency mechanisms already built into the processor and does not require additional hardware support.

Error Correcting Codes. As transistors reach low nanometer scales, they become increasingly susceptible to a number of reliability issues. Such issues range from soft errors caused by cosmic radiation to permanent faults caused by premature aging or variability in the manufacturing process. To counter these effects, microprocessor designers are deploying significant error detection and correction capabilities in the most vulnerable components of the chip. Modern processors use various forms of hardware error correction [1, 4, 8, 17, 19, 21, 24]. A large body of research work on novel types of ECC designed to protect vulnerable memory structures exists [2, 14, 18, 26, 30, 31]. The expected decrease in reliability of future CMOS generations will most likely lead to an increase in ECC coverage on-chip. This will translate into better coverage of approaching timing margin violations as a result of voltage or frequency speculation. Our work leverages these ECC techniques and benefits from increased on-chip resiliency support.

Process Variation. Several researchers have proposed microarchitectural techniques to mitigate or tolerate parameter variation. They target register file and execution units [16], data caches [20], pipeline balancing [28], intelligent floorplanning [7] and core-to-core variation in power [3]. Other work proposed variation-aware thread scheduling [27] to exploit core-to-core variability. In this work, we characterize variation effects using measurements performed on real hardware. In addition, our voltage speculation algorithms adapt to the variation properties of individual cores by classifying them as conservative or aggressive and by tailoring the V_{dd} to each core's voltage sensitivity.

8. CONCLUSION AND FUTURE WORK

This paper presented a novel technique for voltage speculation that relies on on-chip correctable errors as a proxy for estimating timing margins. The solution was implemented in System Firmware and prototyped on an Intel Itanium-based server. Evaluation of the prototype with a range of benchmark applications showed significant power savings ranging from 18% to 23%, with minimal performance impact. The paper also characterized the impact of process variation on core-to-core distribution of timing margins using measurements on Itanium II processors. This study revealed significant core-to-core variability in voltage margins. It also revealed significant heterogeneity in the number and types of correctable errors that are triggered by identical applications running on design-identical cores.

We hope this work will open new avenues for research on voltage speculation using on-chip ECC. We plan to extend this study to other high performance processors that have strong on-chip error correction, such as the Intel Core i7. We would like to also explore other sources of dynamic variability, such as temperature. A preliminary experiment conducted on our platform revealed that temperature variations within 10°C do not cause measurable changes in correctable error rates. This was an unexpected result that we plan to investigate further, perhaps by forcing higher temperature variations.

9. ACKNOWLEDGMENTS

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

10. REFERENCES

[1] 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.

[2] 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.

[3] J. Donald and M. Martonosi. Power efficiency for variation-tolerant multicore processors. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 304–309, October 2006.

[4] 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)*, pages 102–103, February 2007.

[5] 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)*, pages 7–18, December 2003.

[6] B. Greskamp and J. Torrellas. Paceline: Improving single-thread performance in nanoscale CMPs through core overclocking. In *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 213–224, September 2007.

[7] E. Humenay, D. Tarjan, and K. Skadron. The impact of systematic process variations on symmetrical performance in chip multi-processors. In *Design Automation and Test in Europe (DATE)*, April 2007.

[8] Intel Core^{TM} i7 Processor. `http://www.intel.com`.

[9] Intel Itanium architecture software developer's manual, 2010, revision 2.3.

[10] Intel Itanium processor 9500 series reference manual, 2012, revision 0.2.

[11] Intel Itanium processor 9560 (32M cache, 2.53 GHz). `http://ark.intel.com/products/71699/Intel-Itanium-Processor-9560-32M-Cache-2_53-GHz`.

[12] Intel Itanium processor family system abstraction layer specification, 2008, revision 3.3.

[13] N. James, P. Restle, J. Friedrich, B. Huott, and B. McCredie. Comparison of split-versus connected-core supplies in the POWER6 microprocessor. In *International Solid-State Circuits Conference (ISSCC)*, pages 298–604, February 2007.

[14] 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)*, pages 197–209, December 2007.

[15] 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)*, pages 1–11, December 2011.

[16] X. Liang and D. Brooks. Mitigating the impact of process variations on processor register files and execution units. In *International Symposium on Microarchitecture (MICRO)*, pages 504–514. IEEE Computer Society, December 2006.

[17] 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*, 41(1):229–237, January 2006.

[18] 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)*, pages 351–362, December 2010.

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

[20] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou. Yield-aware cache architectures. In *International Symposium on Microarchitecture (MICRO)*, December 2006.

[21] N. Quach. High availability and reliability in the Itanium processor. *IEEE Micro*, 20(5):61–69, 2000.

[22] V. J. Reddi, S. Kanev, W. Kim, S. Campanoni, M. D. Smith, G.-Y. Wei, and D. Brooks. Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling. In *International Symposium on Microarchitecture (MICRO)*, pages 77–88, December 2010.

[23] R. Riedlinger, R. Arnold, L. Biro, B. Bowhill, J. Crop, K. Duda, E. Fetzer, E. Fetzer, O. Franza, T. Grutkowski, C. Little, C. Morganti, G. Moyer, A. Munch, M. Nagarajan, C. Parks, C. Poirier, B. Repasky, E. Roytman, T. Singh, and M. Stefaniw. A 32nm 3.1 billion transistor 12-wide-issue Itanium processor for mission-critical servers. *IEEE Journal of Solid-State Circuits*, 47(1):177–193, 2012.

[24] 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)*, pages 84–86, February 2011.

[25] S. Sarangi, B. Greskamp, A. Tiwari, and J. Torrellas. Eval: Utilizing processors with variation-induced timing errors. pages 423–434, November 2008.

[26] 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*, pages 254–262, October 2008.

[27] R. Teodorescu and J. Torrellas. Variation-aware application scheduling and power management for chip multiprocessors. In *International Symposium on Computer Architecture (ISCA)*, pages 363–374, June 2008.

[28] A. Tiwari, S. R. Sarangi, and J. Torrellas. ReCycle: Pipeline adaptation to tolerate process variation. In *International Symposium on Computer Architecture (ISCA)*, June 2007.

[29] C. Weaver and T. Austin. A fault tolerant approach to microprocessor design. In *International Conference on Dependable Systems and Networks (DSN)*, pages 411–420, July 2001.

[30] 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.

[31] D. Yoon and M. Erez. Memory mapped ECC: Low-cost error protection for last level caches. *ACM SIGARCH Computer Architecture News*, 37(3):116–127, 2009.