# UniFI: Leveraging Non-Volatile Memories for a Unified Fault Tolerance and Idle Power Management Technique

Somayeh Sardashti
Department of Computer Sciences
University of Wisconsin-Madison
somayeh@cs.wisc.edu

David A. Wood
Department of Computer Sciences
University of Wisconsin-Madison
david@cs.wisc.edu

## ABSTRACT
Continued technology scaling presents new challenges for system-level fault tolerance and power management. Decreasing device sizes increases the likelihood of both transient and permanent faults. Increasing device count, together with the end of Dennard scaling, makes power a critical design constraint. Techniques that seek to improve system reliability frequently use more power. Similarly, many techniques that reduce power hurt system reliability. Ideally system designers should seek out techniques that mutually benefit both fault tolerance and power management.

In this paper, we develop a unified technique, called UniFI, for fault tolerance and idle power management in shared memory multi-core systems. UniFI leverages emerging non-volatile memory technologies to provide an energy-efficient lightweight checkpointing technique. In addition to tolerating a large class of faults, UniFI's frequent checkpoints permit near-instant transition to a deep sleep mode to reduce idle power. UniFI incurs very low performance and energy overheads during fault-free execution—less than 2%—while taking checkpoints every 0.1ms. For typical server workloads (such as DNS), UniFI reduces average power by 82% by shutting off during idle periods.

## Categories and Subject Descriptors
C.1 [**Processor Architectures**]: General

## Keywords
Energy, Reliability, Checkpointing, Idle power management.

## 1. INTRODUCTION
Continuing advances in technology scaling promise significantly higher levels of system integration, but to effectively exploit this potential requires concomitant advances in fault tolerance and power management. Decreasing feature sizes pose new challenges to system reliability, due to increased rates of transient and permanent faults [4]. Future multicore systems must also treat power as a first-class design constraint, and should seek not only to limit the maximum power dissipation but also to make power - proportional to performance by managing idle power. The twin challenges of fault tolerance and power management motivate our work towards a unified technique that addresses both.


*ICS'12,* June 25–29, 2012, San Servolo Island, Venice, Italy.


Power has become a primary design constraint for multicore processors and systems, and is projected to become ever more important as technology continues to scale [10]. Shrinking transistor dimensions results in the exponential growth of leakage power, while dynamic power no longer follows Dennard's classical scaling trend [36]. In addition, due to scaling conventional memory technologies encounter scaling difficulties and power issues. To answer these issues, emerging resistive memory technologies — such as Phase-Change Memory (PCM) and Spin-Torque Transfer Magneto-resistive RAM (STT-MRAM) – have been proposed [11-23]. They promise to not only eliminate the major sources of leakage power, but to also make main memory and caches non-volatile [14,20].

Server power is also a critical factor at data centers, where power consumption is growing rapidly [35]. In most data centers, servers have relatively low utilization, but exhibit bursty behavior and are rarely idle for long. Conventional idle power management techniques, such as OS-directed ACPI power modes, are insufficient due to high transition overheads. What is needed are low-overhead idle power management techniques, which rapidly transition between idle and active power modes [34]. On the other hand, power management techniques may degrade reliability if applied aggressively due to decreased noise margins and thermal cycling [42].

Technology scaling and power management both impact system reliability. Shrinking semiconductor feature sizes results in a higher possibility of process and operating condition variations, and increases the susceptibility of system components to both transient and permanent faults. In a data center, power management may also interrupt server power due to power outages, power over-subscription (e.g., power capping), or thermal emergencies [11]. To recover from a wide range of failures, various checkpointing techniques have been proposed operating at different software or hardware layers [7,24,25,26,27]. However, conventional checkpointing techniques incur high performance and power overheads or require high disk bandwidth and capacity [27].

This paper proposes UniFI, a Unified Fault tolerance and Idle power management technique for shared-memory multicore systems. UniFI helps both system reliability and power by exploiting the synergy between backward error recovery and idle power management. It leverages resistive memory and provides a lightweight energy-efficient checkpointing mechanism to recover from a wide range of transient and permanent faults. Relying on its light-weight global checkpoint mechanism, UniFI allows almost instant transition of the system to a deep sleep mode during idle periods and at power emergencies. Unlike other power management mechanisms, UniFI's checkpoint mechanism also helps tolerate any fault due to increased thermal cycling.

**Table 1. Taxonomy of fault tolerance and power management techniques.**

| | | System Reliability | | |
|---|---|---|---|---|
| | | **Tends to hurt** | **Neutral** | **Tends to help** |
| **System Power** | **Tends to hurt** | N/A | N/A | High-overhead global checkpointing [7],[24],[25], and redundancy [40] mechanisms |
| | **Neutral** | N/A | N/A | Power-aware reliability techniques [41], Rebound [45] |
| | **Tends to help** | Aggressive power management techniques [38][42] | Razor [39], MS-ECC [47], Word-disable and Bit-fix schemes [48], heterogeneous LLC [46] | UniFI [New] |

We evaluate UniFI's performance and energy overheads using full-system simulation with SPEComp [3], PARSEC and commercial workloads [2] running on an 8-processor multicore system. For continuous fault-free execution, UniFI incurs very low performance and energy overheads, less than 1% on average and less than 2% for the worst performing workloads. For low-utilization server workloads, we use Meisner, et al.'s power model [34] to show that UniFI significantly eliminates idle power.

The main contributions of this paper are as follows:

• To the best of our knowledge, UniFI is the first unified technique that addresses the synergies between fault-tolerance and idle power management.

• UniFI proposes an efficient checkpointing technique—using a novel combination of lazy flushing, in-cache logging, and safe replacement—that incurs low performance and energy overheads in the common case of fault-free execution, allowing frequent checkpointing.

• UniFI exploits the unique characteristics of resistive memories to efficiently recover from a wide range of permanent and transient faults and to provide efficient idle power management.

In the rest of the paper, we study the synergies between fault tolerance and power management techniques in Section 2, describe the background in non-volatile memories in section 3, present UniFI and its mechanisms in Section 4, give evaluation methods in Section 5, provide experimental results in Section 6, discuss related work in Section 7, and conclude the paper in Section 8.

## 2. SYNERGY BETWEEN FAULT TOLERANCE AND POWER MANAGEMENT

With continued transistor scaling, designers face new challenges in both system reliability and system power. Despite decades of fault-tolerance research, most studies on reliability have paid little attention to power overheads [27]. Similarly, recent power management techniques, which reduce average power and temperature, may degrade reliability due to decreased noise margins and thermal cycling [42]. System designers should seek solutions that synergistically improve both system reliability and power.

The taxonomy in Table 1 categorizes various fault tolerance and power management techniques based on their impacts on reliability and power. Different fault tolerance techniques can be categorized into Forward Error Recovery (FER), which uses redundant computation, and Backward Error Recovery (BER), which maintains checkpoints or logs. FER (e.g., ECC and triple-modular redundancy) tends to consume additional power [40]; however some recent work uses dynamic adaptation to provide a target reliability level with minimum resources and power [41]. Traditional BER techniques such as SafetyNet [25] and ReVive [24] incur significant power overhead for large on-chip log buffers [25], flushing caches and writing logs to main memory [24]. In a recent work, although Dong et al. use PCM for checkpointing [7], their proposed technique has also high power and performance overheads, and so low checkpoint frequency. More recently, Rebound reduces both performance and power overhead by using coordinated local, rather than global, checkpoints [45].

Aggressive power management techniques can hurt system reliability by increasing both transient and hard faults. For instance, aggressive idle power management techniques, such as PowerNap, can increase hard error rates due to thermal cycling [42]. Active power management techniques, such as DVFS and drowsy cache [49], can also hurt transient fault rates if being used aggressively[38]. Recent work has proposed using additional redundancy and/or selective reconfiguration to improve the reliability of caches in low-power operating modes [39,46,47,48].

UniFI seeks to help both system reliability and power by exploiting the synergy between backward error recovery and idle power management. It provides a light-weight global checkpoint mechanism, and unlike other aggressive idle power management techniques, it leverages its checkpoint mechanism to tolerate possible faults due to increased thermal cycling.

## 3. BACKGROUND IN NON-VOLATILE MEMORY TECHNOLOGIES

Since conventional memory technologies encounter scaling difficulties and power issues as semiconductor feature sizes shrink, the research community has proposed using resistive memory technologies, such as PCM, STT-MRAM, Resistive RAM (RRAM), and memristors, at different levels of the memory hierarchy [11-23]. While still relatively early in the technology development cycle, these technologies all promise scalability, non-volatility, high density, and energy-efficiency.

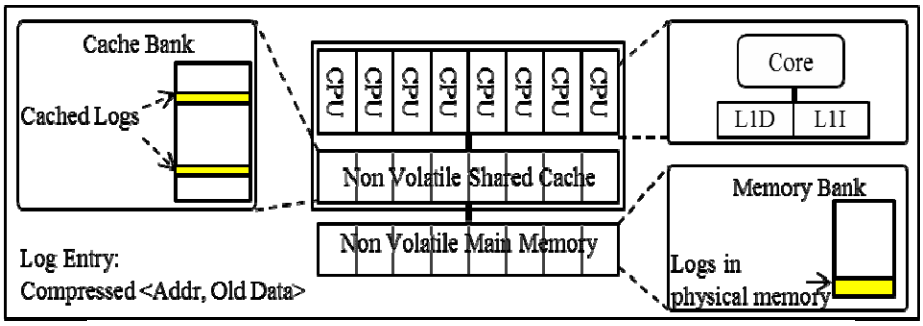


**Figure 1. A general view of a multicore system with UniFI's support.**

While UniFI is largely independent of which technologies ultimately dominate the market, it does assume that both main memory and the last level cache are non-volatile. To make our work concrete, we focus on PCM and STT-MRAM, which are two promising alternatives.

As DRAM technology faces scalability and power issues, PCM has gained attention in the research community as a DRAM replacement for main memory [11-19]. In comparison with DRAM, PCM is scalable, denser, non-volatile, and has near zero leakage power. On the other hand, it has lower write endurance, higher latency and higher writing power. Different techniques have been proposed to make PCM competitive with DRAM [12,13,15,16,17,19].

As vendors increase the number of cores per die, there is a commensurate demand for larger and low power on-chip caches. Conventional SRAM-based caches cannot meet both objectives due to their low density and high leakage power. Recent studies show that STT-MRAM is a good candidate for the next generation of large on-chip caches [20-22]. STT-MRAM is scalable, fast (similar to SRAM) and dense (similar to DRAM), and has near zero leakage power and very high write endurance. On the other hand, it takes longer than SRAM to write, which can be addressed by different techniques [20,21].

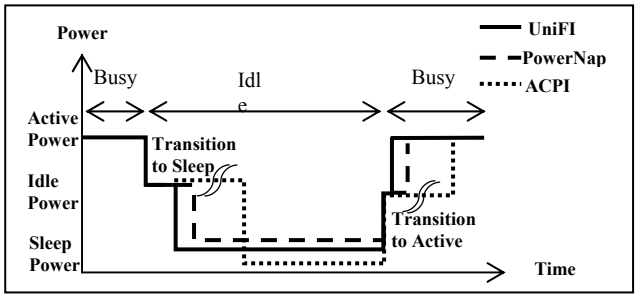


**Figure 2. Different idle power management techniques.**

# 4. UNIFI: A UNIFIED TECHNIQUE FOR FAULT TOLERANCE AND IDLE POWER MANAGEMENT

In this section, we first describe a general overview of UniFI. We then provide a detailed explanation of all UniFI mechanisms.

## 4.1 UniFI Overview

We propose UniFI, a unified mechanism that helps provide high availability and power proportionality. UniFI's main goals are providing low-overhead checkpointing and recovery, rapid transition to/from low-power mode, and recovery from a wide range of faults. To achieve these goals, UniFI proposes different mechanisms, all described below. UniFI leverages emerging non-volatile memory technologies to make both the last-level cache and main memory persistent across power outages. It provides a low-overhead global checkpointing mechanism, and creates safe checkpoints (i.e., stored in non-volatile storage) periodically without ever quiescing the system. Using its checkpointing mechanism, it recovers from a wide range of faults, described in Section 4.5, including power outages.

Figure 2 shows how different idle power management techniques eliminate idle power. To provide an efficient idle power management mechanism for server workloads with short idle periods, deep sleep mode and rapid transition between sleep and active modes are required. To eliminate idle power efficiently, UniFI provides very rapid transition (almost instant) of the system to/from a very deep sleep mode. UniFi's system-level idle power management mechanism, described in Section 4.4, treats transitions to deep low-power states much like unintentional power outages. When an idle period is detected, it rapidly transitions the system to a very deep sleep mode with powered-off cores, caches, and main memory. In response to instantaneous load, it quickly transitions the system to the high-performance active state by recovering the system to the last safe checkpoint. Unlike other power management techniques (such as PowerNap), UniFI compensates for the possible higher error rates caused by its aggressive power management with its efficient checkpoint recovery mechanism.

UniFI is largely independent of any specific fault detection mechanism [43]. It keeps multiple checkpoints, so it can support relatively long-latency, signature-based fault detection mechanisms. In addition, to keep I/O consistent, UniFI leverages existing techniques, e.g., ReViveIO [44], that buffer input and output in non-volatile buffers until they are known to be safe. As UniFI's checkpointing interval is short (e.g., 0.1ms), buffering outputs has little impact on most application's performance.

## 4.2 UniFI System Model

Figure 1 shows a general view of a shared-memory multi-core system with UniFI's support. Cores have one or more levels of private, volatile caches, and share a non-volatile last-level cache and non-volatile main memory.

Each core has volatile, private L1 instruction and (write-back) data caches implemented using conventional SRAM. They share non-volatile STT-MRAM L2 cache and PCM main memory. To hide STT-MRAM high write latency, we use multiple banks and sub-bank buffers in the L2 cache [21].

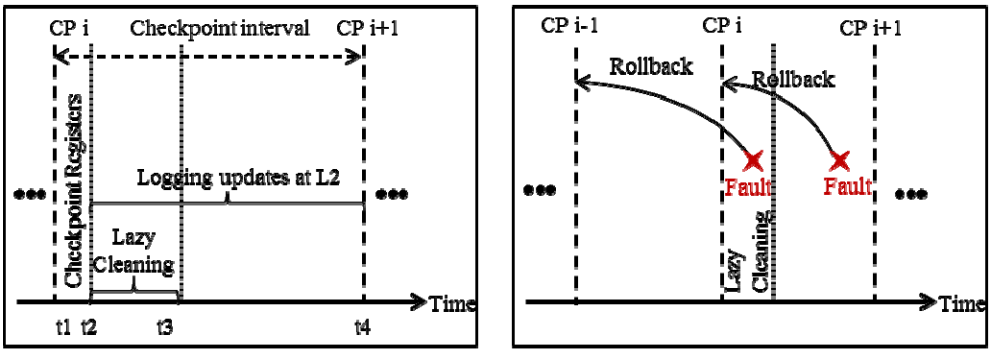


**Figure 3. different phases of creating a checkpoint (a) and rollback recovery from different faults.**

In addition, to make PCM main memory competitive with DRAM, we apply different techniques including: multiple narrow buffer rows in memory banks and word-level dirty bits in the caches, to mitigate PCM's high write latencies and to improve its energy and endurance [13].

## 4.3 UniFI Checkpointing Mechanisms

Figure 3 (a) illustrates different phases of creating a checkpoint with UniFI. Checkpoints occur at a logical time (e.g., time t1 in Figure 3) defined as the edge of a checkpointing clock, which is a loosely synchronized clock received at different modules (e.g., caches and cores). A checkpoint can be used as a recovery point when it is safe in non-volatile memory, which is any time after all checkpoint state resides in stable storage (e.g., at time t3 in Figure 3 (a)). UniFI ensures that there is always a safe checkpoint by keeping at least two checkpoints at a time.

### *4.3.1 Checkpointing at Processors*

To establish a new persistent checkpoint, UniFI checkpoints processor registers first. As there are a limited number of registers, performance overhead of register checkpointing is negligible. Therefore, processors checkpoint their registers explicitly, storing them into their private caches via their load/store units. Register checkpoints will be safe in stable storage (i.e., non-volatile L2 and main memory) once the L1 caches are cleaned (described next).

### *4.3.2 Lazy Cleaning of Volatile Private Cache*

To create a safe checkpoint, UniFI makes all updates preceding that checkpoint persistent before they are modified or lost. Figure 4 illustrates UniFI's checkpointing mechanism at caches. To keep checkpointing overhead low, UniFI handles it entirely within the hardware. It augments the L1 cache tags with a flash clear bit column (CP bit in Figure 4) [31] indicating if the line has been cleaned in the current checkpointing interval.

UniFI uses a lazy checkpointing mechanism at caches, and cleans L1 caches without stopping the running applications. It uses a simple state machine in the L1 cache controller to walk through the L1 cache and clean a cache line (i.e., copy it to the non-volatile L2 cache) if it is dirty and not cleaned yet (e.g., Cache Line A in Figure 4). If there is an update request to such a line UniFI first makes it safe by copying it back to the non-volatile L2 before processing the new request. Since update requests are not on the critical path, as shown in Section 6, its overhead is negligible.

### *4.3.3 Logging Updates at the Shared Cache.*

Between two successive checkpoints, UniFI logs data updates to the L2 cache by storing their physical addresses and old data as log entries. UniFI keeps logs as part of the physical address space (e.g., in an address assigned at boot time). To provide low-overhead checkpointing and fast recovery, UniFI caches logs in the L2 cache, in the same bank as the original cache line resides. By caching the logs, UniFI avoids extra costs of special log buffers [7][25] and high energy and performance overheads of storing logs in the main memory [24]. Also, since most of the logs reside in the cache, as shown in Section 6, recovery is fast and energy efficient.

To reduce the possible overheads of caching logs (e.g., cache pollution), UniFI employs two techniques: logging only first updates in a checkpointing interval and compressing logs. During a checkpointing interval, UniFI only logs on the first update to a cache line using the CP bit added to the L2 cache (Figure 4). UniFI also compresses logs using Frequent Pattern Compression (FPC) [1], which is a hardware-based low overhead compression technique. FPC achieves high compression ratio for most applications by compressing frequent patterns (e.g., consecutive 0s and 1s) of data [1]. By applying these techniques, and due to the fact that UniFI creates checkpoints frequently and keeps a limited number of them (e.g., two), its overhead due to polluting the cache is low. As shown in Section 6, each UniFI's checkpoint is about 23 KB or 1.14% of the L2 cache on average.

Since UniFI cleans L1 caches lazily, it keeps two active checkpoints (the previous and the current checkpoints). It logs updates generated by cleaning the L1 caches as part of the previous checkpoint, while logging others in the current checkpoint.

### *4.3.4 Safe Data Replacements*

Although UniFI assumes non-volatile shared cache and main memory, there are usually volatile buffers in between, such as write-back buffers and buffers in the memory controller. To achieve recovery from instant power-off, UniFI ensures that data in these buffers are not lost using a safe replacement policy. In a directory-based coherence protocol with 1-phase write-back technique, which the cache controller sends the replaced address and data to the memory controller at the same time, or 3-phase write-back technique, which the cache controller waits for an acknowledgement from the memory controller before sending the data and unblocking the cache line, UniFI adds one more phase to ensure data is safe. It basically blocks the cache line until it

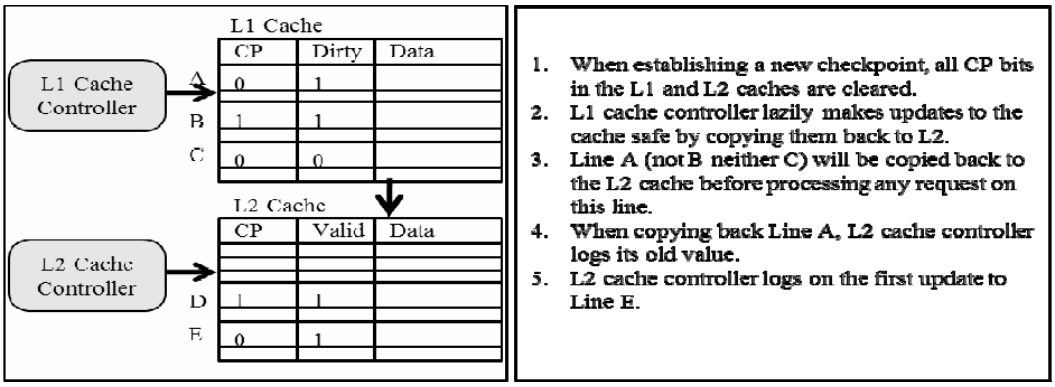


**Figure 4. UniFI checkpointing mechanism.**

receives an unblock acknowledgment message from the memory controller indicating that it has issued the request to the persistent main memory (i.e., 2-phase or 4-phase write-back). In this way, even in the case of power-off, data can be found in the cache.

### *4.3.5 Synchronization*

Since UniFI logs all updates at the L2 cache, unlike global checkpointing techniques such as SafetyNet [25], it needs a simple synchronization technique. To create globally synchronized checkpoints, UniFI needs processors and cache controllers to receive a checkpointing clock at which they start creating a new checkpoint. Since the L2 cache controller handles all logs, a loosely synchronized clock (i.e., it could have skew) can be used at different modules. In addition, as UniFI lazily cleans the L1 caches, when L1 cache controllers complete cleaning the caches, they need to notify the L2 cache controller (e.g., by sending an ack) of the new safe checkpointing.

### *4.3.6 Rollback Recovery*

When a fault is detected, UniFI recovers the system to a safe checkpoint preceding the fault. The system first diagnoses the error (e.g., using a service processor), reconfigures it in case of permanent failure, and reinitializes the hardware. These steps are beyond this paper's scope. UniFI handles recovery entirely within the hardware. To recover a checkpoint, as shown in Figure 3 (b), processors first restore their register checkpoints in parallel, reading and unrolling them via their load/store units. The L2 cache controller then reads and undoes the checkpoint logs in reverse order from the most recent checkpoint to the recovery point.

UniFI provides rapid recovery using different techniques including: caching logs, parallel undo, and pipelined recovery. Since UniFI caches logs, it finds most of the logs in the L2 cache at the time of recovery. UniFI also keeps logs in the same bank as the original cache line resides, so it can unroll them in parallel. In addition, it pipelines reading logs, decompressing them, and restoring their data to the cache or processors in order to reduce recovery overhead.

## 4.4 UniFI Idle Power Management Mechanisms

UniFI leverages non-volatile memory technologies and its efficient checkpointing mechanism to provide both fast transitioning and very low power system-level sleep mode. Figure 5 illustrates two mechanisms to eliminate idle power with UniFI.

In the first mechanism, shown in Figure 5 (a), UniFI instantly transitions the system to the deep sleep mode in case of a power emergency or a detected idle period. Later, when there is a new job to process or the power failure is fixed, UniFI rapidly transitions the system to the active mode by recovering the system to the most recent safe checkpoint before the transition (CPi in Figure 5 (a)). It then finishes the previous job (i.e., redo the lost work), and starts the new job. This mechanism is mostly useful for emergencies, such as thermal emergencies and power outage, since it does not let the previous job commit (i.e., buffer its I/O outputs) until the system transitions back to the active mode and it creates another safe checkpoint.

On the other hand, when there is no emergency (e.g., to eliminate a detected idle period), UniFI can let the previous job commit before transitioning to the sleep mode. In this mechanism, shown in Figure 5 (b), UniFI makes a new safe checkpoint just before switching to the sleep mode by checkpointing processors' registers, flushing their L1 caches, and committing buffered I/O outputs. On a new job arrival, it activates the system by recovering to this safe point, which only includes restoring registers. In addition to system level power management, UniFI can provide fine-grained power management at core level by checkpointing registers of a core and flushing its L1 caches. That core or even another core (e.g., using server consolidation) can later continue running by reloading its register.

## 4.5 UniFI Fault model

UniFI is designed to recover from a wide class of faults. It recovers from multiple simultaneous transient or permanent faults at any part of the system as long as it can access a safe checkpoint. As UniFI keeps checkpoints in the non-volatile shared cache and main memory, it must assume they are error free. Fortunately, UniFI can leverage well-known techniques, ranging from conventional ECC to RAID-like memories [24], to protect memory.

UniFI tolerates various transient or permanent faults including those result in loss of all data in volatile caches (e.g., due to a glitch on reset signal), permanent failure of a core and its private cache. In addition to transient and permanent faults, one of our goals is recovering from instant power-off. Instant power-off may happen unintentionally due to power failures (e.g., power outage or thermal emergencies), or intentionally for idle power management. In all of these cases, UniFI recovers the system state to a consistent fault-free state preceding the fault or the sleep mode.

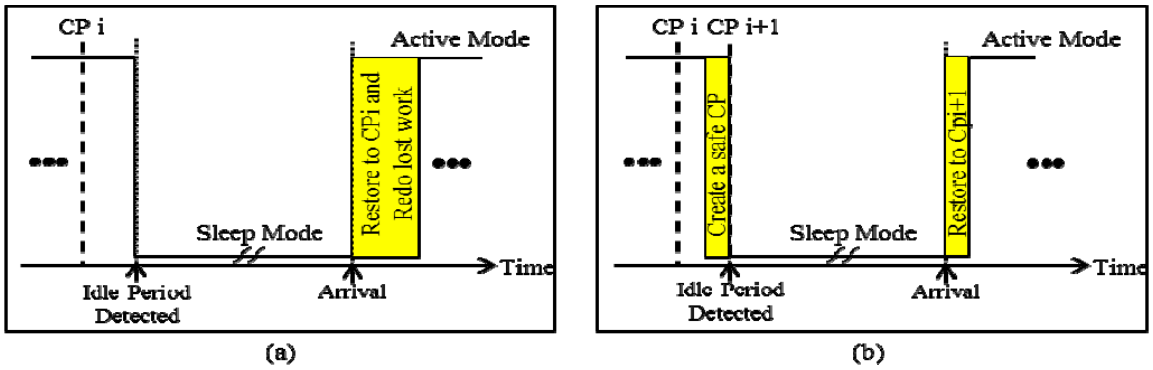


**Figure 5. Two different mechanisms to eliminate idle power with UniFI.**

## 5. EVALUATION SETUP

In order to evaluate UniFI, we use two mechanisms: full system simulation and analytical modeling. To evaluate UniFI's fault tolerance mechanism, we have implemented UniFI in a full-system simulator based on GEMS [28]. We use CACTI [30] and McPAT [50] to model power at 32nm. We model a multi-core system with 8 cores, shared STT-MRAM L2 cache, and PCM main memory (Figure 1) modeled after the parameters in Table 2. We also consider overheads of FPC compression, which take 0.41ns and consumes 0.112nJ for compressing/decompressing 64-byte data [6]. We use a similar system with same configurations but without UniFI's support as our baseline. We have also enhanced our simulator to model the energy and delay of STT-MRAM and PCM with the parameters in Table 3. In this Table, the STT-MRAM cache cells are optimized for density and access energy, while its corresponding SRAM-based cache is designed for low leakage and high speed access [21].

**Table 2. Simulation Parameters**

| | |
|---|---|
| **Cores** | 8 OOO cores, 4GHz frequency, 4-wide issue, 192 physical registers |
| **L1 Caches** | Private, 32-KB iL1/dL1, 4-way associative, 3-cycle access latency |
| **L2 Caches** | Shared, 4-MB, 8-way associative, 8 banks, 8-cycle Read latency, 24-cycle Write latency, MESI Coherency |
| **Main Memory** | 4GB, 16 banks, 400MHz bus frequency |
| **Checkpointing Parameters** | 2 checkpoints, 400K cycles (0.1ms) checkpointing interval |

We evaluate UniFI's checkpointing mechanism with a large variety of applications from PARSEC with simlarge input sets (Blackscholes, Bodytrack, Canneal, Fluidanimate, Freqmine, Streamcluster, and Swaptions), SPEComp (Ammp, Applu, Equake, Fma3d, Gafort, Mgrid, Swim, and Wupwise), and commercial workloads (Apache, Oltp, Jbb, and Zeus). Since these benchmarks are multi-threaded, we use a work-related metric, run each workload for a fixed number of transactions/iterations (for approximately 100M instructions), and report the average of 15-20 runs using randomized memory delays to address workload variability [37]. We evaluate UniFI's overheads with a very frequent checkpointing, creating a checkpointing every 0.1 ms (i.e. checkpoint frequency of 10KHz or 400K core cycles). We keep two checkpoints (one active, one safe). In this way UniFI can support long fault detection techniques with latency of up to 800,000 cycles.

To evaluate power savings of UniFI's power management, we extend the power model presented in Meisner et al. [34], which model power saving in a server using an M/G/1 queuing model. Their model calculate power savings of an idle power management technique based on workload parameters (e.g., the work-load's average busy time), and the power management mechanism's parameters (e.g., transition latency between active and sleep modes, and sleep power). We use this model, which we skip its details here, and augment it with UniFI's checkpointing power and performance overheads to find UniFI's power savings in a typical blade. We also find UniFI's impact on response time using the model presented in their paper [34].

**Table 3. STT-MRAM cache and PCM memory parameters**

| **L2 Cache Parameters** | **4MB SRAM** | **4MB STT-MRAM** |
|---|---|---|
| Rd/Wrt Latency (core cycle) | 10/10 | 8/24 |
| Energy per Rd/Wrt (pJ/64B) | 1268/1268 | 798/952 |
| Leakage Power (mW) | 6578 | 3343 |
| **Memory Parameters** | **DRAM** | **PCM** |
| tCL/tRCD/tWTR/tWR/tRTP/ tRP/tCCD/tWL (mem cycle) | 5/5/3/6/3/5/ 4/4 | 5/22/3/6/3/60/4 /4 |
| Energy per Rd/Wrt (pJ/bit) | 1.17/0.39 | 2.47/16.82 |

Table 4 shows the power consumption of a typical blade [34], which has two cores operating at 2.4 GHz and eight DRAM DIMMS. We use the RAILS power supply units (PSU) [34], which significantly reduces energy costs of PSU when idle power management techniques are being used. We evaluate UniFI and compare it with PowerNap using different server workloads [34].

## 6. EVALUATION

In this section, we first evaluate our baseline system and study impacts of using non-volatile memories. We then evaluate UniFI's checkpointing mechanism and idle power management.

### 6.1 Evaluation: Baseline

To study impacts of using non-volatile memories in the system, we evaluate four baseline configuration listed in Table 5 with the parameters in Table 2. The last two configurations employ sub-bank buffers at the STT-MRAM L2 cache [21], buffer reorganization and partial writes at PCM main memory [13].

**Table 4. Component power consumption of a typical blade with/without power management techniques**

| Blade Components | Active Power | Idle Power | Sleep Power with PowerNap | Sleep Power with UniFI |
|---|---|---|---|---|
| CPU Chip | 80-150W | 12-20W | 6.8W | 0W |
| DRAM DIMMs | 3.5-5W | 1.8-2.5W | 1.6W | 0W |
| Other modules (PSU, SSD, etc.) | 110-262W | 210-230W | 2W | 2W |
| **Total** | **450W** | **270W** | **10.4W** | **2W** |

Figure 6 and Figure 7 illustrate performance and energy of baseline configurations normalized to SRAM-DRAM baseline system. A baseline of STT-MRAM and PCM system without any extra technique (the second configuration) is on average 1.5x slower and requires 1.2x more energy than a SRAM-DRAM system. Applying different techniques in non-volatile memories reduce this delay and energy gap to 1.08x and 0.92x, and makes the baseline competitive with SRAM-DRAM system. To evaluate UniFI, we use this configuration as our baseline system.

**Table 5. Baseline Configurations**

| | |
|---|---|
| **SRAM-DRAM** | 4-MB SRAM L2$, DRAM memory |
| **STT-PCM w/o techs** | 4-MB STT-MRAM L2$, PCM memory |
| **STT-PCM w techs (our baseline)** | 4-MB SRAM L2$, PCM memory, extra techniques applied |
| **STT-PCM w large cache** | 8-MB SRAM L2$, PCM memory, extra techniques applied |

In addition to these techniques, a large DRAM buffer can be used before PCM main memory to hide its high latencies and energy overheads [16]. In the last configuration, we study using a larger STT-MRAM on-chip cache instead since STT-MRAM is scalable, energy efficient and non-volatile with similar density comparing to DRAM. This configuration further reduces the energy and delay gap to 0.95x and 0.88x.

## 6.2 Evaluation: UniFI

Figure 8 and Figure 9 show the breakdown of UniFI's performance and energy overheads over the baseline system during fault-free execution. UniFI incurs less than 2% (0.5% on average) performance and energy overheads for all the workloads (0.08% for PARSEC, 0.45% for SPEComp, and 1.46% for commercial workloads) while it creates checkpoints frequently (every 400K cycles or 0.1ms). UniFI also increases Energy-Delay product (ExD) up to 3.7% for OLTP, and 1.02% on average. The main sources of UniFI's overheads are caching logs in the L2 cache and cleaning L1 caches. Compressing logs and lazy cleaning significantly reduce these overheads. Compressing logs reduces performance overhead of caching logs by up to 4.1% (for OLTP) and 0.64% on average for all the workloads. In addition, lazy cleaning improves performance by up to 2.1% and 0.3% on average. Similarly, these techniques reduce energy overhead by up to 4.5% and on average 1%.

For workloads with large footprints in the L2 cache, such as commercial workloads and some of SPEComp (e.g., Swim) and PARSEC workloads (e.g., Canneal), UniFI's overhead is mainly due to caching logs in the L2 cache, and so increasing cache miss ratio. For instance, UniFI increases miss ratio by 1.41% to 80

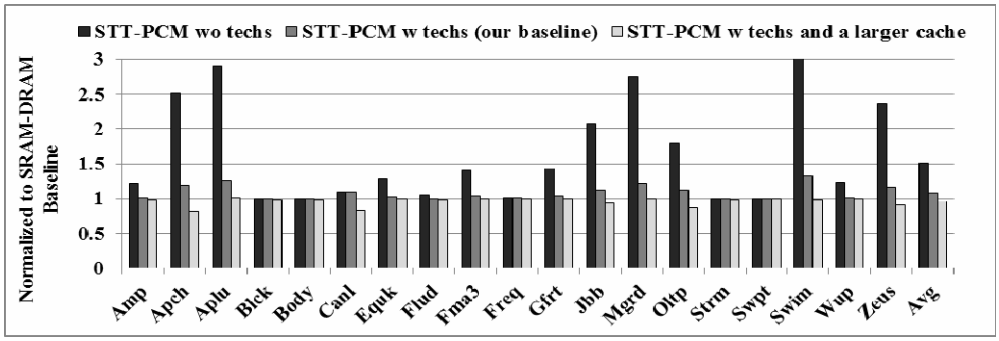


**Figure 6. Performance of different baseline configurations.**

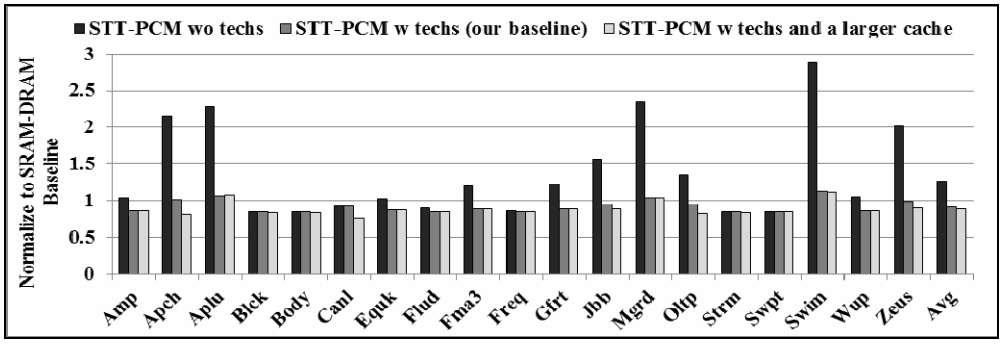


**Figure 7. Energy of different baseline configurations.**

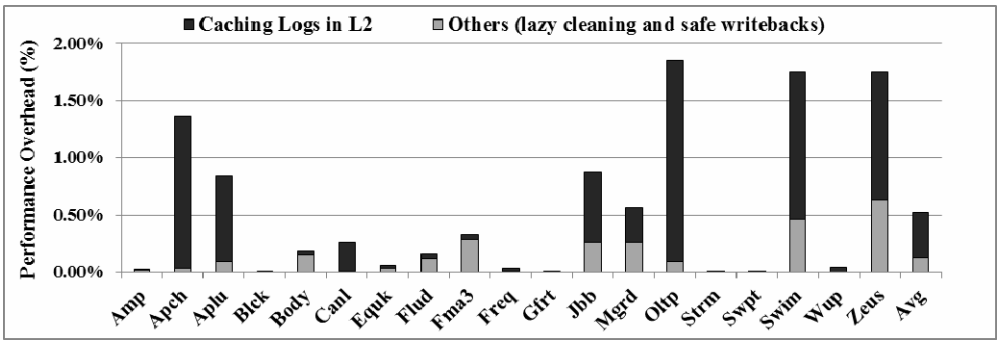


**Figure 8. Breakdown of UniFI's performance overhead over the baseline.**

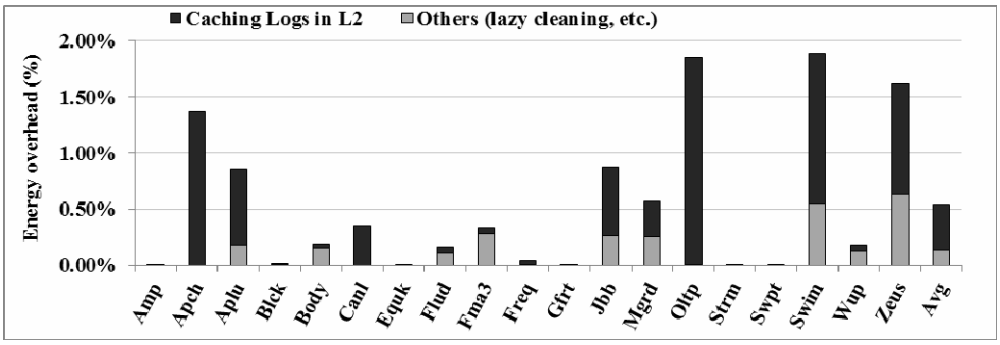


**Figure 9. Breakdown of UniFI's energy overhead over the baseline.**

MPKI (misses per kilo instructions) for Swim. This overhead is also a function of checkpoint size, which is fairly small for most of the workloads (23 KB or 1.14% of the L2 cache on average). For OLTP, UniFI creates checkpoints with average size of 117KB (5.7% of the L2 cache), which explains its high performance and energy overheads in comparison with other workloads.

As UniFI lazily cleans the L1 caches in the background, its overhead due to cleaning caches is low (on average 0.1%). This overhead is related to the number of dirty lines copied back when establishing a checkpoint, and so is higher for those with more dirty lines (such as Zeus, Swim, and Fma3d). Since access to the main memory is not on the critical path, safe data replacements at the L2 cache has negligible effects.

To estimate the overhead of recovery and unavailability of the system due to an error, we consider the worst case recovery scenario. For example, when an error occurs after establishing the first checkpoint and is detected at the end the second checkpoint interval (i.e., detection latency of about 800K cycles), UniFI unrolls both checkpoints. Since most the logs reside in the L2 cache (on average 99.5% of logs stay in the cache), the overhead of unrolling a checkpoint is low. It takes less than 40K cycles (10us) to unroll one checkpoint on average. Considering approximately 0.2ms for lost work (2 checkpointing intervals), UniFI provides about 99.999999% availability if one of these faults occurs per day.

In addition to evaluating UniFI's checkpointing mechanism, we study how it saves power by eliminating idle power. Figure 10 illustrates power saving (part (a)) and relative response time (part (b)) with UniFI and PowerNap in a typical blade with parameters in Table 4 for different server applications. UniFI's transition time between sleep and active states, and checkpointing overhead are very low, however we conservatively assume 0.1ms transition time and 2% performance/energy overheads. As shown in Figure 10, UniFI provides better power saving and response time than PowerNap for most applications because of its rapid transition to the deep sleep mode and leveraging non-volatile memories. Due to UniFI's rapid transitions, its checkpointing overhead is mostly amortized for most of the workload except Cluster as it has longer service time and higher utilization.

# 7. RELATED WORK

UniFI improves the standard global checkpointing technique proposed in SafetyNet and ReVive [24][25] to provide low-power checkpointing. UniFI provides frequent checkpointing with much less energy, performance and area overheads. In addition to leveraging non-volatile memory technologies, UniFI differs from ReVive by caching compressed logs in LLC (similar to regular data), and so significantly reducing power and memory bandwidth overheads of writing and recovering logs into/from the main memory. UniFI also eliminates the high overhead of flushing caches in ReVive by lazily cleaning dirty cache lines only in the private caches. In comparison with SafetyNet, UniFI eliminates the extra power and area costs of separate log buffers. Regarding to leveraging non-volatile memories, the technique proposed in [7] also leverages PCM, however unlike UniFI, they use special 3D stacked PCM storage as checkpoint buffers. They also have a costly mechanism to create a checkpoint, which stalls the system periodically and copies DRAM content to the PCM buffers.

UniFI also leverages its checkpointing mechanism to save power for server workloads comparably to PowerNap idle power management. Unlike PowerNap and other aggressive power management techniques, UniFI compensates the possible higher error rates due to aggressive power management with its low overhead checkpoint recovery mechanism.

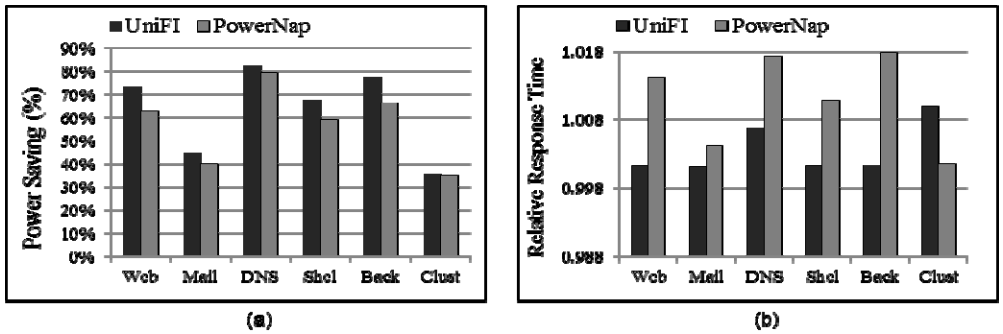


**Figure 10. UniFI's power savings and relative response time for server workloads Breakdown of UniFI's**

# 8. CONCLUSIONS

We propose UniFI, a unified technique that addresses two critical challenges of reliability and power management together. UniFI exploits resistive memory technologies, and proposes a light-weight energy-efficient checkpointing to recover from transient and permanent faults, and power failures. Exploiting UniFI's ability to recover from instant power-off, we use UniFI to eliminate idle power of servers with short idle periods. We demonstrate that it incurs less than 2% performance and energy overheads for a wide range of applications. We also show that for typical server workloads with short idle periods, UniFI can reduce average power by up to 82% by leveraging its low overhead checkpointing mechanism and non-volatile memories.

# 9. REFERENCES


[1] Alameldeen, A., and Wood, D. 2004. Adaptive Cache Compression for High-Performance Processors. In *Proc. of the 31st Annual Intnl. Symp. on Computer Architecture* (June 2004).

[2] Alameldeen, A., Mauer, C., Xu, M., Harper, P., Martin, M., Sorin, D., Hill, M., and Wood, D. 2002. Evaluating Non-deterministic Multi-threaded Commercial Workloads. In *Proc. of the 5th Workshop on Computer Architecture Evaluation Using Commercial Workloads* (Feb. 2002).

[3] Aslot, V., Domeika, M., Eigenmann, R., Gaertner, G., Jones, W., and Parady, B. 2001. SPEComp: A New Benchmark Suite for Measuring Parallel Computer Performance. *In Workshop on OpenMP Applications and Tools* (July 2001).

[4] Borkar, S. 2005. Designing Reliable Systems from Unreliable Components: the Challenges of Transistor Variability and Degradation. *IEEE Micro* (November 2005).

[5] Borkar, S., Jouppi, N., and Stenstrom, P. 2007. Microprocessors in the era of terascale integration. In *Proc. of the conference on Design, automation and test in Europe*.

[6] Das, R., Mishra, A., Nicopoulos, C., Park, D., Narayanan, V., Iyer, R., Yousif, M., and Das, C. 2008. Performance and power optimization through data compression in Network-on-Chip architectures. In *Proc. of the 14th IEEE Symp. on High-Performance Computer Architectur*.

[7] Dong, X., Muralimanohar, N., Jouppi, N., Kaufmann, R., and Xie, Y. 2009. Leveraging 3D PCRAM technologies to reduce checkpoint overhead for future exascale systems. In *Proc. of the Conference on High Performance Computing Networking, Storage and Analysis*.

[8] Fernandez-Pascual, R., Garcia, J., Acacio, M., and Duato, J. 2007. A Low Overhead Fault Tolerant Coherence Protocol for CMP Architectures. In *Proc. of the 13th IEEE Symp. on High-Performance Computer Architecture* (February 2007).

[9] Hagersten, E., and Koster, M. 1999. WildFire: A Scalable Path for SMPs. In *Proc. of the 5th IEEE Symp. on High-Performance Computer Architecture* (January 1999).

[10] International technology roadmap for semiconductors. Process integration, devices, and structures, 2009.

[11] Condit, J., Nightingale, E., Frost, C., Ipek, E., Lee, B., Burger, D., and Coetzee, D. 2009. Better I/O through byte-addressable, persistent memory. In *Proc. of the ACM SIGOPS 22nd symposium on Operating systems principles*.

[12] Ipek, E., Condit, J., Nightingale, E., Burger, D., and Moscibroda, T. 2010. Dynamically replicated memory: building reliable systems from nanoscale resistive memories. In *Proc. of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems*.

[13] Lee, B., Ipek, E., Mutlu, O., and Burger, D. 2009. Architecting phase change memory as a scalable dram alternative. In *Proc. of the 36th Annual Intnl. Symp. on Computer Architecture*.

[14] Lee, B., Zhou, P., Yang, J., Zhang, Y., Zhao, B., Ipek, E., Mutlu, O., and Burger, D. 2010. Phase-Change Technology and the Future of Main Memory. *IEEE Micro*.

[15] Qureshi, M., Karidis, J., Franceschini, M., Srinivasan, V., Lastras, L., and Abali, B. 2009. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *Proc. of Micro*.

[16] Qureshi, M., Srinivasan, V., and Rivers, J. 2009. Scalable high performance main memory system using phase-change memory technology. In *Proc. of the 36th Annual Intnl. Symp. on Computer Architecture*.

[17] Schechter, S., Loh, G., Straus, K., and Burger, D. 2010. Use ECP, not ECC, for hard failures in resistive memories. In *Proc. of the 37th Annual Intnl. Symp. on Computer Architecture*.

[18] Zhang, W., and Li, T. 2009. Characterizing and mitigating the impact of process variations on phase change based memory systems. In *Proc. of Micro*.

[19] Zhou, P., Zhao, B., Yang, J., and Zhang, Y. 2009. A durable and energy efficient main memory using phase change memory technology. In *Proc. of the 36th Annual Intnl. Symp. on Computer Architecture*.

[20] Zhou, P., Zhao, B., Yang, J., and Zhang, Y. 2009. Energy reduction for STT-RAM using early write termination. In

*Proc. of the 2009 International Conference on Computer-Aided Design*.

[21] Guo, X., Ipek, E., and Soyata, T. 2010. Resistive computation: avoiding the power wall with low-leakage, STT-MRAM based computing. In *Proc. of the 37th Annual Intnl. Symp. on Computer Architecture*.

*[22]* Zhu, J. 2008. Magnetoresistive Random AccessMemory: The Path to Competitiveness and Scalability. In *Proc. of the IEEE.*

[23] Mohan, C., and Bhattacharya, S. 2010. Implications of Storage Class Memories (SCM) on Software Architectures. In *HPCA 2010 Workshop on the use of Emerging Storage and memory Technologies (WEST)*.

[24] Prvulovic, M., Zhang, Z., and Torrellas. J. 2002. ReVive: Cost-Effective Architectural Support for Rollback Recovery in Shared-Memory Multiprocessors. *In Proc. of the 29th Annual Intnl. Symp. on Computer Architecture*.

[25] Sorin, D., Martin, M., Hill, M., and Wood, D. 2002. SafetyNet: Improving the Availability of Shared Memory Multiprocessors with Global Checkpoint/Recovery. In *Proc. of the 29th Annual Intnl. Symp. on Computer Architecture* (May 2002).

[26] Teodorescu, R., Nakano, J., and Torrellas, J. 2006. SWICH: A Prototype for Efficient Cache-Level Checkpointing and Rollback. *IEEE Micro*.

[27] Torrellas, J. 2009. Architectures for Extreme-Scale Computing. *Computer*.

[28] Martin, M., Sorin, D., Beckmann, B., Marty, M., Xu, M., Alameldeen, A., Moore, K., Hill, M., and Wood, D. 2005. Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset. *Computer Architecture News*.

[29] Brooks, D., Tiwari, V., and Martonosi, M. 2000. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proc. of the 27th Annual Intnl. Symp. on Computer Architecture* (June 2000).

[30] Shyamkumar, T., Muralimanohar, N., Ahn, J., and Jouppi, N. 2008. *CACTI 5.1*. Technical Report HPL-2008-20, Hewlett Packard Labs.

[31] Lee, D., and Katz, R. 1991. Using Cache Mechanisms to Exploit Nonrefreshing DRAM's for On-Chip Memories. *IEEE Journal of Solid-State Circuits* (April 1991).

[32] Xie, Y., Loh, G., Black, B., Bernstein, K. 2006. Design space exploration for 3D architectures. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*.

[33] Meixner, A., Bauer, M., and Sorin, D. 2007. Argus: Low-Cost, Comprehensive Detection of Errors in Simple Cores. In *Proc. of Micro*.

[34] Meisner, D., Gold, B., and Wenisch, T. 2009. PowerNap: Eliminating Server Idle Power. In *Proc. of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (Mar. 2009).

[35] Barroso, L., and Holzle, U. 2007. The case for energy-proportional computing. *IEEE Computer* (Jan 2007).

[36] Dennard, R., Gaensslen, F., Rideout, V., Bassous, E., and LeBlanc, A. 1974. Design of Ion-Implanted MOSFET's with Very Small Physical Dimensions. *IEEE Journal of Solid-State Circuits* (Oct. 1974).

[37] Alameldeenand A., Wood, D. 2003. Variability in architectural simulations of multi-threaded workloads. In *HPCA*.

[38] Degalahal, V., Lin, L., Narayanan, V., Kandemir, M., and Irwin, M. 2005. Soft errors issues in low-power caches. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*.

[39] Das, S., Pant, S., Rao, R., Pham, T., Ziesler, C., Blaauw, D., Austin, T., Flautner, K., and Mudge, T. 2004. Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation, *IEEE MICRO* (Dec. 2004).

[40] Gomaa, M., Scarbrough, C., and Vijaykumar, T. 2003. Transient-Fault Recovery for Chip Multiprocessors. In *Proceedings of ISCA-30* (June 2003).

[41] Miller, T., Surapaneni, N., Teodorescu, R., and Degroat, J. 2009. Flexible Redundancy in Robust Processor Architecture, *Workshop on Energy-Efficient Design (WEED), in conjunction with ISCA* (June 2009).

[42] Srinivasan, J., Adve, S., Pradip, B., Rivers, J. 2005. Lifetime reliability: toward an architectural solution. *IEEE Micro*.

[43] Mukherjee, S. 2008. Architecture Design for Soft Errors. *Elsevier Inc*.

[44] Nakano, J., Montesinos, P., Gharachorloo, K., and Torrellas, J. 2006. ReViveI/O: Efficient handling of I/O in highly-available rollback-recovery servers. *Intl. Symp. on High-Perf. Com, Arch.*

[45] Agarwal, R., Garg, P., and Torrellas, J. 2011. Rebound: Scalable Checkpointing for Coherent Shared Memory, *ISCA* (June 2011).

[46] Ghasemi, H., Draper, S., and Kim, N. 2011. Low-Voltage On-Chip Cache Architecture using Heterogeneous Cell Sizes for Multi-Core Processors. In *HPCA*.

[47] Chishti, Z., Alameldeen, A., Wilkerson, C., Wu, W., and Lu, S. 2009. Improving Cache Lifetime Reliability at Ultra-Low Voltages, In *Proc. of Micro* (December 2009).

[48] Wilkerson, C., Gao, H., Alameldeen, A., Chishti, Z., Khellah, M., and Lu, S. 2008. Trading Off Cache Capacity for Reliability to Enable Low Voltage Operation, *35th Annual International Symposium on Computer Architecture* (June 2008).

[49] Flautner, K., Kim, N., Martin, S., Blaauw, D., and Mudge, T. 2002. Drowsy Caches: Simple Techniques for Reducing Leakage Power. *Proc. IEEE/ACM 29th Intl. Symposium on Computer Architecture* (May 2002).

[50] Li, S., Ahn, J., Strong, R., Brockman, J., Tullsen, D., Joupp, N. 2009. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *Proc. of Micro.*