Improving Multi-Core Performance Using Mixed-Cell Cache Architecture

Samira M. Khan[1,2], Alaa R. Alameldeen[1], Chris Wilkerson[1], Jaydeep Kulkarni[1], Daniel A. Jiménez[3]
[1]Intel Labs [2]Carnegie Mellon University [3] Texas A&M University

Abstract

Many enterprise and mobile systems must operate within strict power constraints. These systems dynamically trade off performance and power to maximize performance while keeping power within specified limits. In multi-core systems, maximizing the number of active cores within a strict power budget requires minimizing the power per core. Lowering core voltage dramatically reduces power, but compromises cache reliability. Mixed-cell cache architectures, where part of the cache is designed with larger, more robust cells, enable caches to operate reliably at low voltage while minimizing the added cost of larger cells. But mixed-cell caches suffer from poor low-voltage scalability since caches can only use robust cells at low voltage, sacrificing up to 75% of cache capacity. Such capacity reduction strains shared cache resources, leading to significant performance losses.

In this paper, we propose a mixed-cell architecture that improves multi-core performance by allowing the use of both robust and non-robust cells. Our mechanisms store modified data only in robust lines by modifying the cache replacement policy and handling writes to non-robust lines. For a multi-core processor, our best mechanism improves performance by 17%, and reduces dynamic power in the L1 data cache by 50% over prior mixed-cell proposals.

1. Introduction

Power continues to be an important design constraint in modern microprocessors. In mobile systems, thermal design power (TDP) plays a key role in determining the form factor of the device. Likewise, data centers are built with fixed power and cooling capabilities. Improving performance within a given power constraint (MIPS/watt) yields direct economic benefits by increasing the compute capability supported by a fixed investment in datacenter infrastructure.

Designers have responded to continuing demand for performance within power constraints with traditional improvements in core performance and efficiency, but also by increasing the number of cores on a die. Today, state of the art server processors may contain tens of cores; and even mobile products, including tablets and smart phones, have more than one core. Increasing core counts, in the context of fixed power budgets, is a key challenge for future systems.

Today's performance oriented systems meet specific power targets by varying the voltage of active cores as the number of active cores changes [9]. As cores become inactive, the voltage of the remaining cores rises to maximize performance. As the number of active cores increases, the voltage of all cores will drop to avoid exceeding power limits. Changing the voltage in response to changes in core activity allows the power budget of these systems to remain constant regardless of the number of active cores. In fact, a given power budget can support more cores as long as the power consumed per core is reduced via microarchitectural improvements or by reduced voltage.

Voltage reduction in a power-limited system permits an increase in active cores, increasing performance for workloads that benefit from additional cores. However, reducing voltage leads to a dramatic loss of reliability for memory circuits operating at low voltages. To circumvent this problem, prior work has explored using separate voltages for the core logic and caches. This captures most of the power benefits by reducing the core voltage, while ensuring reliable cache operation at a higher voltage. However, separate voltage domains greatly increase design complexity [17]. Such complexity can be avoided by building the cache with cells better suited for low voltage operation. Improving a cell's reliability at low voltage involves upsizing existing transistors or adding new ones, both of which increase power and area.

The high overhead of cell upsizing has led architects to propose mixed (heterogeneous) cell cache architectures, consisting of traditional cells and robust cells [7], with the goal of minimizing the use of expensive, robust memory cells, while continuing to harvest their low voltage benefits. Mixed-cell cache architectures achieve this by implementing a small portion of the cache with robust cells, and the remainder with non-robust cells. When operating at a high voltage, both portions would be used to maximize cache capacity and performance. When operating at

978-1-4673-5587-2/13/$31.00 ©2013 IEEE

low voltage, the failure-prone non-robust cells would be turned off, reducing cache capacity by up to 75%.

This paper builds on previous work in mixed-cell cache architectures, focusing on improving their performance and efficiency benefits at near threshold voltage (NTV), 590mV in our case. We observe that the main value of reducing voltage in future multi-core systems is to increase the number of active cores that fit within a constrained power budget. For highly parallel workloads, we show that the ability to utilize more active cores makes the lowest voltage operating point also the highest in performance. While prior work argued that reducing cache size was acceptable at low voltage, we show that reducing cache size (especially for large L3 caches) leads to large performance losses in low voltage multi-core systems.

We propose a mixed-cell cache architecture that preserves both the performance benefits of large caches and the ability to operate at low voltage. In a mixed-cell cache, non-robust cache lines are more susceptible to failures at low voltage, while robust lines are resilient to such failures. To address this disparity, we treat modified and unmodified data differently. We use simple error-detection mechanisms (e.g., parity) on the non-robust lines, and use them only for unmodified (clean) lines. If an error is detected, the cache access is handled like a cache miss. Modified (dirty) data, which cannot be recovered from other caches or memory, are stored only in robust ways. We modify the cache replacement policy to ensure the allocation of modified lines to robust ways. We propose and evaluate three alternatives for dealing with writes to non-robust lines.

Some key contributions of this paper are:

1. We propose an enhanced mixed-cell cache architecture that ensures reliable NTV (590mV) operation through careful management of modified and unmodified data. When operating at NTV, our technique improves performance by 17% and reduces L1 data cache dynamic power by 50% over previous proposals.
2. We achieve the performance improvement at NTV while preserving the high-voltage performance benefits of previous mixed-cell cache designs.
3. In contrast to prior work, we show that when lowering voltage is used to improve performance through increased core count, losing cache capacity to reduce voltage is a poor tradeoff. The additional capacity demand of multi-core workloads outweighs power reductions at low voltage.

In the remainder of this paper, we discuss the impact of decreasing cache capacity and increasing cache latency on low-voltage multi-core performance, and summarize prior schemes for achieving reliability at lower voltages (Section 2). We explain our proposed techniques in detail in Section 3, and the circuit area and latency overheads in Section 4. We introduce the experimental methodology in Section 5, evaluate our design in Section 6, and conclude in Section 7.

2. Background

2.1 Multi-Core Performance at Low Voltage

As systems become more power-constrained, the challenge will be to maximize power-efficient performance across a broad operating range. Each core in a multi-core system, for example, may be operating at very low voltages when running highly-parallel workloads with sufficient work to utilize all cores. Seconds later, after other cores go idle, a single core may "turbo", i.e., operate at a higher voltage and frequency, and use all the shared cache and bandwidth resources to improve single-thread performance. In systems with higher core counts, the high voltage/high frequency operating points will not be the operating points where performance is most critical. Instead, the highest performance operating point will be where the maximum number of cores is active. In a power constrained system, this operating point will be characterized by all cores operating at the lowest voltage to ensure the system meets power constraints.

To illustrate this, we model a hypothetical processor based on Intel's SandyBridge (SNB) processor. When possible, we leverage published data from SNB to set the parameters of our experiments [8][20][26]. Intel's ultra-low-voltage (ULV) SNB has a TDP of 17W, divided between a GPU and two cores. At 700mV, 1.4GHz is the typical operating frequency for these cores. Each core in our hypothetical processor includes 32KB L1 instruction and data caches and a 256KB Mid-Level (L2) Cache. All cores share a 4MB Last-Level (L3) Cache. To evaluate the impact of different Vmin mechanisms on different operating points, we add two operating modes to our hypothetical 2-core system. Each operating mode is constrained to the same power envelope as the 2-core SNB operating at 700mV. Operating a single core at 850mV, for example, requires the same power as 2 cores operating at 700mV, likewise, 4 cores at 590mV. To project the frequency of the cores at 590 and 850mV, we rely on simulated logic delay vs. voltage data by Kulkarni and Roy [16], and apply it to our hypothetical 2-core processor with a 700mV, 1.4 GHz operating mode. Based on this analysis we expect the 4 cores operating at 590mV to be able to operate at 825 MHz, while a single core operating at 850mV would run at 2.1 GHz. In this paper we evaluate approaches to enable operation at near-threshold voltage (590mV) and their impact on performance at different operating modes.

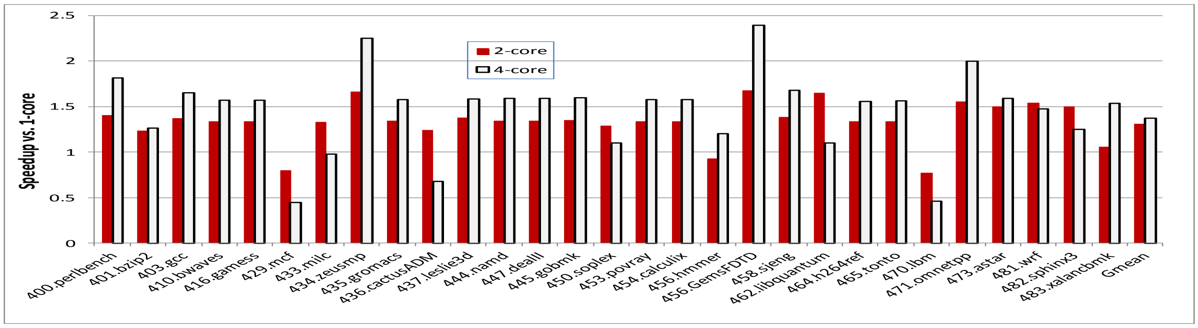

Figure 1. Speedup of 2-core and 4-core systems vs. single-core.

Figure 1 shows the speedup obtained by running multiple instances of the SPECCPU2006 benchmarks [22] on 2-core and 4-core processors vs. a single-core processor. We present more details on our evaluation methodology in Section 5. The figure shows that most benchmarks achieve significant performance improvements at the same power envelope as a single-core system, even though they run at a much lower frequency and share cache and bandwidth resources. Compared to a single-core system, a 4-core system has 37% better performance, and a 2-core system has 31% better performance on average (using the geometric mean). A few cache-sensitive applications (e.g., mcf) lose performance for the 4-core system compared to single-core due to sharing of the L3 and bandwidth resources. Other benchmarks have worse 4-core performance vs. 2-core for the same reason. However, most benchmarks see significant performance improvements (up to 139%). In 21 of the 29 benchmarks, we observe a speedup (with a geometric mean 21.3%) for a 4-core system over a 2-core system. This clearly illustrates the necessity of low-voltage operation to scale multi-core performance, and the need to preserve low-voltage cache capacity to avoid losing the multi-core performance benefits.

2.2 Related Work

In a given technology, SRAM bit cells generally employ minimum-geometry transistors which are susceptible to systematic as well as random process variations such as random dopant fluctuations (RDF) and line edge roughness (LER). Process variations produce V_T (threshold voltage) mismatch between neighboring transistors, resulting in asymmetric bit cell characteristics, and making bit cells susceptible to failure at low voltage. With bit cells susceptible to failure, large memory structures in the core, such as caches, become unreliable at low voltage. As a result, the core must operate at a minimum voltage (Vmin) to ensure reliable operation, and reducing core Vmin relies on reducing the Vmin of its caches. Reducing cache Vmin has become an area of active research. Previous work that addresses the challenge of operating caches at low voltage fits into two broad categories: circuit solutions and architectural solutions.

2.2.1 Circuit Solutions. In general, circuit techniques strive to reduce Vmin by improving the bit cell. One approach is to reduce the voltage for the core logic, but provide a separate higher voltage for caches. However, providing separate voltages complicates the design. A partitioned power supply increases power grid routing complexity, reduces on-die decoupling capacitance, increases susceptibility to voltage droops, and may require level shifters that add latency to signals that cross voltage domains [17]. Generating multiple voltages also increases complexity and inefficiency. Recent work has argued in favor of integrated regulators. Depending on the regulator design and target change in voltage, these incur power losses (~15-20%) [14].

Most commercial processors use multiple voltages generated off-chip by high-efficiency off-chip voltage regulators (~95% efficiency). Intel's Sandybridge processor, for example, has 6 separate power supplies for graphics, memory controller, analog, and IO. As the number of cores increases, providing multiple voltages for each core will become increasingly impractical. A four-core system with separate voltages for the core and its private L1/L2 caches would require 3 voltage domains per core (i.e., a total of 12 power supplies), in addition to those needed for other system components.

Another way to improve bit cell Vmin involves upsizing its constituent devices. Threshold voltage (V_T) variation depends inversely on the transistor gate area (σV_T α $1/\sqrt{W.L}$) [12], where W is the transistor channel width and L is the transistor channel Length. Consequently, upsizing devices can dramatically reduce variations and improve Vmin. However, as illustrated in Figure 2, the Vmin benefits of upsizing a typical 6T bit cell diminish as device size increases.

Figure 2 compares the Vmin for four different caches implemented in a 65nm technology[1]. Each cache is implemented using one of four different 6T

[1] In this paper, we use failure probability data from a recently published paper [16] rather than an earlier work [15]. The new data shows higher operating voltages as it uses a newer process technology (65 nm vs. 130 nm) where variations are significantly higher.

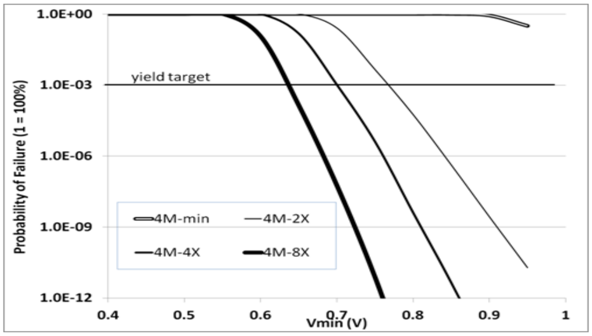

Figure 2. Vmin improvements with bit cell upsizing.

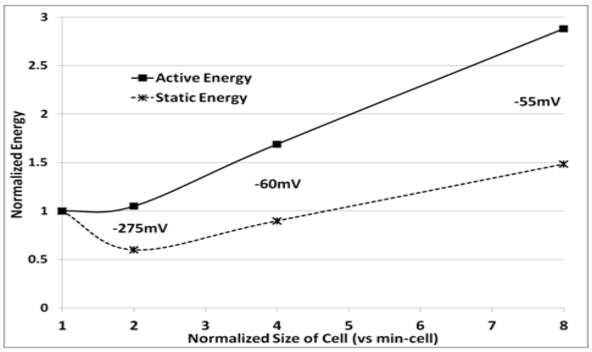

Figure 3. Active and static energy vs. cell size.

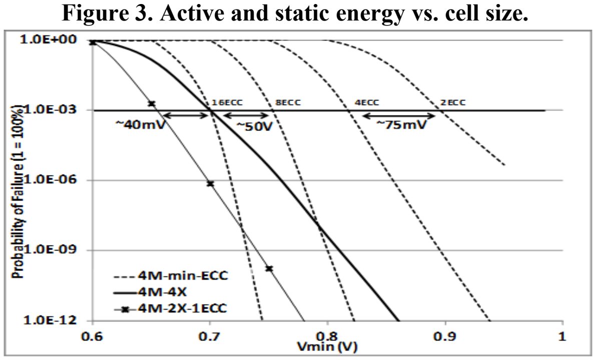

Figure 4. Improving Vmin with cell upsizing vs. ECC.

cells taken from [16]. Like earlier work [24], we set Vmin at the point when the cache failure probability is 1/1000. The figure depicts the probability (y-axis) that the cache will contain a single failing bit as a function of voltage (x-axis). A 4MB cache constructed with a minimum-sized six-transistor (6-T) cell, 4M-min, exhibits very high failure rates (~30%) even at high voltages (>900mV). The 4M-2X implementation of a 4MB cache doubles the device sizes in each memory cell, increasing cell area by 33%. 4M-4X quadruples the size of the devices, doubling the size of the cell. 4M-8X uses the most robust cell with devices that are eight times as large and a 233% larger cell size than 4M-min. Increasing cell sizes initially yields dramatic improvements vs. minimum-sized cells (note the 275mV improvement moving from 4M-min to 4M-2X). But further size increases yield smaller benefits, 60mV and 55mV for the 4M-4X and 4M-8X, respectively.

Cell upsizing may come with additional power overheads as well. Since leakage varies linearly as a function of transistor dimensions, cell upsizing increases static power. Larger cells also add switching capacitance on the word lines (WL) and bit lines (BL) increasing dynamic power. Figure 3 plots normalized active and static energy (y-axis) vs. cell size (x-axis). We calculate the energy for each cell at the lowest operating voltage it can sustain. Upsizing from the minimum cell to the 2X cell yields a substantial benefit since the reduction in Vmin (275mV) more than compensates for the additional power introduced by larger devices. Further upsizing, however, increases power since the costs of larger devices outweigh the savings from voltage reductions (-60mV, -55mV).

2.2.2 Architectural Solutions. Another approach to reducing Vmin uses failure-prone cells with smaller devices, but augments the memory array with the capability to repair itself in the context of bit failures. Prior work introduced a number of different repair mechanisms which depend on memory tests to identify bad bits [1][3][19][21][24]. Relying on memory tests limits the applicability of these approaches when memory tests are expensive or failures are erratic [2]. Other repair mechanisms rely on a variety of coding techniques, such as Error-Correcting Codes (ECC), to autonomously identify and repair defective bits [5][13]. Prior work on this topic has focused on reducing the overhead of ECC; in some cases focusing on the cost of storing the code bits [25], and in other cases focusing on the cost of the coding logic [23].

Fundamentally, each of these approaches trades off the repair mechanism overhead for the ability to compensate for defective bits. For memory designs with very high failure rates, this tradeoff may be unattractive. To illustrate this, Figure 4 depicts a 4 MB cache implemented with a minimum sized cell (4M-min) and an ECC code applied at a cache line granularity. As the strength of the code increases, its Vmin benefit diminishes. Doubling the strength of the code from a 2-bit ECC (2ECC) to a code with the ability to correct 4-bit errors (4ECC) reduces Vmin by 75mV. On the other hand, increasing the strength from an 8-bit ECC (8ECC) to a 16-bit ECC (16ECC) reduces Vmin by 50mV. To operate at 700mV with the 4M-min, we would need to strengthen the ECC past the point of diminishing returns to 16ECC.

For comparison, Figure 4 includes the upsized 4M-4X from Figure 2, and 4M-2X-1ECC, a hybrid implementation that uses both 2X upsized cells and 1-bit ECC. In comparing 16ECC and 4M-4X we see that both produce acceptable yield loss of 0.1% at 700mV. Although the additional bits required for 16ECC introduce 34% overhead vs. 100% overhead for 4M-4X, the ECC checking logic (2-4million transistors) and latency makes 16ECC less attractive. 4M-2X-1ECC, with an overhead of about 40% for modest cell upsizing and a small single-bit error correcting code (SECDED) delivers the best overall tradeoff. Doubling the device size in the minimum cell allows 4M-2X-1ECC to capture most of the benefit available through cell upsizing, after which ECC can be used at a low cost to provide additional Vmin improvement.

To address the high overheads of operating at NTV, Wilkerson et al. [24] improve Vmin by storing error-correction patterns in cache resources, trading off cache capacity for low voltage operation. Chishti et al. [5] identify the limitations of testing-based implementations, and propose to provide error correction capability using Orthogonal Latin Square Codes. Chakraborty et al. [4] also trade off cache capacity for lower voltage. A multi-copy cache stores two copies of each clean datum and three copies of each dirty datum to allow detection and correction of corrupted bits, respectively.

In [6], Dreslinski et al. propose to combine the low voltage benefits of robust upsized cells and the cost benefits of smaller cells by building caches with a mixture of cell types. Cache lines consisting of robust cells operate at low voltage, while a separate power supply provides a higher voltage to less robust cells. By moving recently accessed data to the low voltage cache lines, Dreslinksi et al. service the majority of requests using low voltage cache lines, and reduce active power in the L1 cache.

Ghasemi et al. also propose an architecture that mixes cell types (i.e., sizes) [7], where both robust and non-robust cells are used to provide the full cache capacity at high voltage. As the voltage decreases, the cache portion with cells that are not robust for a given voltage is disabled, reducing the cache capacity (and static power) by 25%-75% depending on operating voltage. This approach impacts performance negligibly at low voltage when the number of operating cores is fixed across the operating voltages. Nonetheless, it may notably impact performance as more cores are activated at lower voltage.

In [6], the reduced voltage does little to improve leakage since most cells continue to operate at a less efficient higher voltage. To address this, Ghesemi et al. propose to power-gate the non-robust portions of the cache, reducing cache capacity by 50-75% [7].

This paper argues that the reduced voltage is often a vehicle to increase active cores in power-constrained systems, so performance at low voltage is critical and the loss of cache capacity is unacceptable. In the next section, we propose a mechanism that combines a circuit technique (cell upsizing) with architecture techniques to improve low-voltage performance with little overhead.

3. Mixed-Cell (MC) Cache Design

The key idea behind our mixed-cell cache is to protect modified lines by storing them in robust cells, while using the remainder of the cache for clean lines. We use simple error detection and correction mechanisms to detect errors in clean lines. We allocate write misses to robust lines, and read misses to clean lines. On a subsequent write to a clean line, we investigate three alternatives to ensure modified data is not lost. In the remainder of this section, we describe the details of our implementation.

3.1 Cache Architecture

Figure 5 shows all three levels of our cache hierarchy with support for robust cells. Our baseline cache hierarchy is typical of what's found in modern processors [11] with a 32KB 8-way L1 cache, 256KB 8-way L2 cache, and a 4MB 16-way L3 cache. For each level in the cache hierarchy, we implement two ways with robust cells, while the remaining ways use standard (non-robust) cells. This adds an area overhead of 25% (L1 and L2) and 12.5% (L3) for the cache data array. We add a status bit associated with each tag indicating whether the associated line is a robust way or a non-robust way. We don't necessarily need this extra bit if the robust ways are fixed to two specific ways (Way 0 and Way 1 in Figure 5). We also add an extra LRU bit since we implement a different replacement algorithm in the low-voltage mode.

The right level of error detection capability varies for different cache levels. Since the L1 cache is byte-accessible and extremely latency sensitive, we use a parity bit for each byte in the L1 [27], similar to Intel's Atom and Core processors. We use simple SECDED ECC for each line in the L2 and L3 caches. We provide this protection for both robust and non-robust lines to account for soft errors as well as voltage-dependent failures. In general, detectable errors in clean data are recoverable, but those in dirty data are not. To minimize DUE (detectable unrecoverable errors), we handle modified data differently from unmodified data.

If an error is detected in a clean line, it is treated like a cache miss and is obtained from the next level in the cache hierarchy. For modified data, however, we must ensure a very low probability of failure, which we achieve through the use of robust (upsized) cells. This is particularly true in the L1, where parity is unable to correct bit errors and the increased robustness of the cell allows us to minimize the likelihood of bit errors. To simplify our L1 cache implementation, we handle all accesses to failing lines as cache misses. Since the number of such lines is small, this has a very little impact on performance. For the L2 and L3 caches, SECDED ECC corrects most errors. Errors that are detected but not corrected (e.g., lines with two errors) are handled as cache misses and obtained from the next cache level or from memory. L2 and L3 lines that incur double-bit errors can be disabled to avoid undetectable errors, i.e., silent data corruption (SDC), when soft errors hit the same line.

Although a detailed analysis of the pros and cons of disabling cache lines is outside the scope of this paper, our analysis shows that the probability of failures in robust cells is extremely low at the voltages we consider. For example, we find that 99.9% of the L3 caches will suffer failures in less than 1% of all lines at low voltage (Section 4). It's worth noting, however, that despite the minimal loss of capacity in the average set, specific sets may suffer significant capacity loss. These sets may cause performance outliers when running workloads that exercise them heavily. Due to this performance variability some designs may prefer alternative approaches to mitigate defects.

Our Mixed-Cell cache handles writes differently from reads. The main condition we need to satisfy is to store modified data only in robust ways. To achieve this goal, we modify the cache replacement policy to handle write misses differently from read misses (Section 3.2). We also need to handle subsequent writes to non-robust lines (Section 3.3).

3.2 Changes to Cache Replacement Policy

We assume the baseline caches implement a Least-Recently Used (LRU) replacement policy. While the proposed mechanism could be applied to other replacement policies, we chose LRU to simplify our description. In our mixed-cell cache architecture, we allocate write misses only to robust ways, and read misses to non-robust ways. On a read miss, we choose a replacement victim, NR_LRU, only from non-robust ways based on LRU bits. On a write miss, we choose a victim, GLOBAL_LRU that is the LRU line among all ways of the set (both robust and non-robust). If the victim line is robust, we trigger a writeback for modified data, and allocate the new line in its place. If the chosen victim is in a non-robust way, we choose the LRU line from the two robust ways (RB_LRU), trigger a writeback for modified data to convert the RB_LRU line to a clean line, move the RB_LRU line to use the GLOBAL_LRU line's storage, and allocate the new line to the RB_LRU line.

3.3 Handling Writes to Non-Robust Lines

Our mixed-cell cache architecture needs to prevent DUE and SDC for modified data. For lines allocated on a write miss, the cache replacement algorithm guarantees that modified data will only be stored in robust cells. However, for lines that were allocated to non-robust ways on a read miss, we explore different alternatives to prevent failures:

1. Writeback. We handle the write to a non-robust line like we would for a write-through cache. We store modified data in the same non-robust line, but convert it to a clean line by writing back the data immediately to the next cache level. This writeback traffic causes additional network congestion, as well as power and latency overheads. A write to the L1 cache can trigger

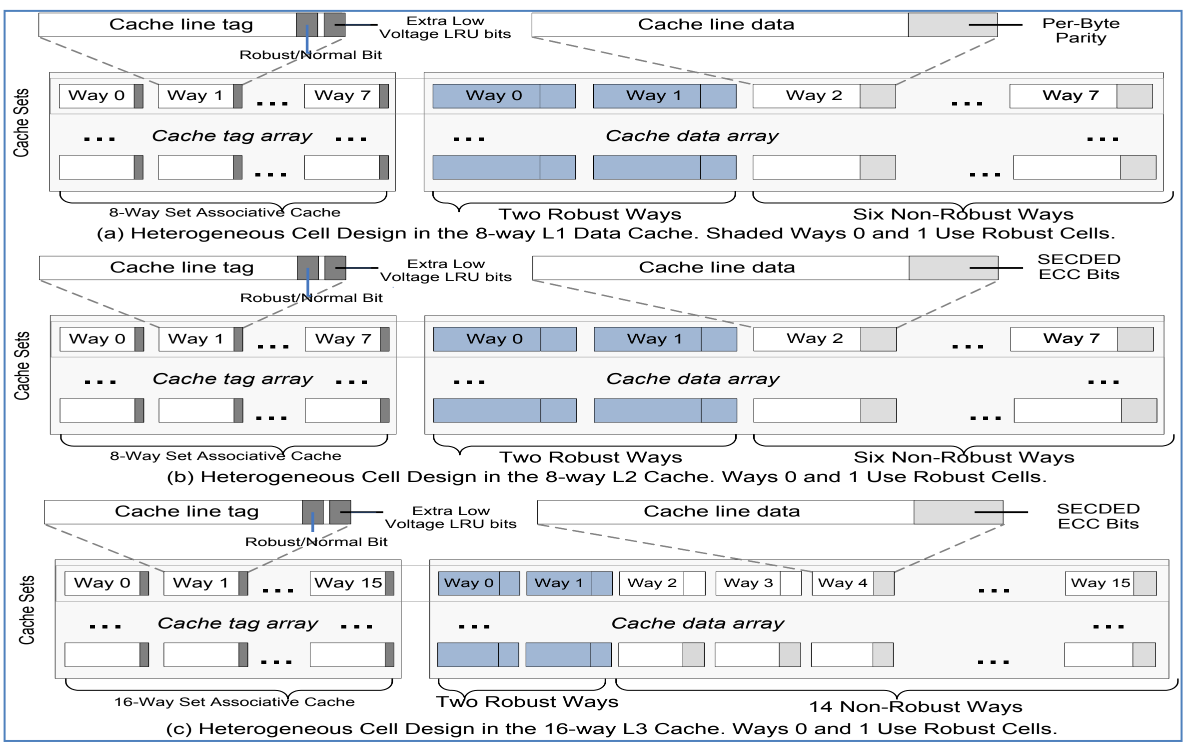

Figure 5. Mixed-cell cache architecture. L1 cache uses byte parity. L2 and L3 use SECDED ECC.

cascading writes all the way to memory if the L2 and L3 caches allocated the same line to non-robust ways.

2. Swap. For many benchmarks, we observe that a write to a cache line is usually followed by more writes to the same cache line. To reduce writeback traffic, we handle a write to a non-robust line by swapping with the LRU way of robust lines in the set, RB_LRU. The RB_LRU line triggers a writeback to convert to a clean line. The RB_LRU line is then swapped with the written line. The status and LRU bits are also swapped between the two cache tags. This approach reduces traffic as it is more likely to write to the most recently-written line than it is to write to the LRU robust line. We model this mechanism's overhead by blocking access to the cache for 3 cycles (L1) or 6 cycles (L2 and L3 that have 32-byte accesses) to account for using the cache read and write ports to perform the swap.

3. Duplication. To avoid writeback traffic and the additional swap latency, we explore trading off capacity to save this overhead. In this mechanism, we assign each two consecutive non-robust lines as "partner lines" similar to [27]. For example, in Figure 5's L1 cache, the line in way 2 is a partner line to that in way 3, the line in way 4 is a partner line to that in way 5, and the line in way 6 is a partner line to that of way 7. When a write occurs to a non-robust line, we evict its partner line and write the data to both lines, using two extra cycles. We modify the replacement algorithm so that the partner line is always invalid and is not a candidate for replacement. This duplication causes losing some cache capacity, but avoids writeback traffic and swap overhead. When writing to a duplicate line, we perform the write to both the original line and its partner. When reading from a duplicate line, we check parity (L1) or ECC (L2/L3), and trigger a read from the partner line if an error is detected.

In Section 6, we compare the relative performance for these three implementations. We next explore mixed-cell alternative designs and their overheads.

4. Mixed-Cell Vmin Analysis

In mixing cell types, we hope to enable reliable operating modes as low as 590mV while minimizing overheads when operating at higher voltages. Using this baseline, we construct three hypothetical designs capable of operating at 590mV, creating the additional power headroom to run four cores. Due to the dramatic benefits the 2X cell produces relative to the minimum cell, we use the 2X upsized cells in our baseline cache designs. The dotted lines in Figure 6 show the Vmin of the L1, L2 and L3, the three largest Vmin limiting structures in a typical CPU. Each cache consists of 2X upsized cells; the L2 and L3 are augmented with a SECDED code. Due to high sensitivity to additional latency in the L1 and high cost of per-byte ECC, the L1 implements byte parity which allows error detection but cannot repair failing bits. As depicted in Figure 6, the lack of ECC causes the Vmin of the L1 cache (700mV) to exceed that of the L2 and L3 caches, each of which can operate comfortably below 700mV. However, none of the arrays can operate below 600mV.

As proposed in [7], we replace two ways in both the L2 and L3 caches with 4X upsized cells. By leveraging the 4X larger cells to store irreplaceable data and the ability of the SECDED code to detect (but not correct) 2 bit errors in non-critical data, we can improve the Vmin of both caches by 60mV, meeting our 590mV target. In the L1, we adopt a similar mechanism to [6], replacing 2-ways in the L1 with upsized cells. Since the 4X upsized cells fail to operate reliably at 590mV without ECC protection, we use the larger 8X upsized cells as our robust cells instead. Figure 6 shows the impact of adding more robust cells on Vmin, improving the L1 cache by 125mV and the L2 and L3 by 60mV each, enough to permit 590mV operation.

In Figure 7, we compare three reduced Vmin designs to a baseline that uses more robust cells. Rather than plotting the Vmin of each cache separately, Figure 7 shows the maximum Vmin of all caches for each mechanism. We describe the compared mechanisms in more detail in the next section. ROBUST builds all caches using only robust cells. MC_DISABLE applies the mixed-cell approach proposed in [6] and [7] to all caches in the hierarchy. Lines implemented with robust cells replace two ways out of each 8-way set in the both the L1 and L2 caches, and four out of sixteen ways in the L3. Since only robust ways operate at low voltage, the Vmin of the robust portion of the cache determines the overall cache Vmin. The smaller cache capacity when operating at Vmin accounts for the slight improvement the MC_DISABLE shows when compared to ROBUST. MC_SWP refers to our proposed mechanism in which both the robust portions of the cache and the non-robust portions of the cache combine to determine Vmin. In general, our ability to detect errors in the non-robust portions of the cache result in 10X reduction in failure rate relative to the robust portions. As a result the robust portions of the cache determine the Vmin for both MC_DISABLE and MC_SWP, causing them to exhibit similar failure rates at 590mV.

Circuit/Design Overheads. Our mixed-cell cache design uses typical 6-T bit cells and upsized 6-T bit cells. Mixing these two cell types within a sub-array (bank) can reduce area efficiency and introduce manufacturing complications. Matching the poly-pitch of the different cell types and growing the cell in only one dimension (Figure 8) mitigates design and manufacturing complexity of a mixed-cell cache.

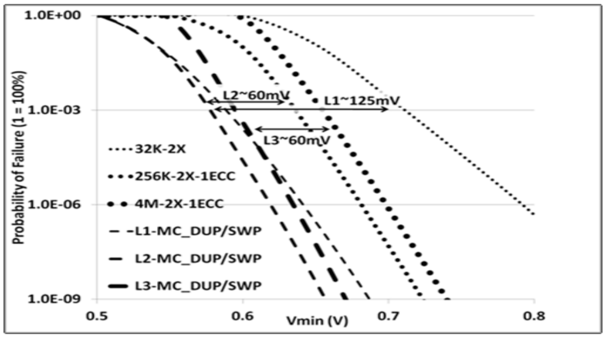

Figure 6. Vmin at different cache levels in baseline and mixed-cell caches.

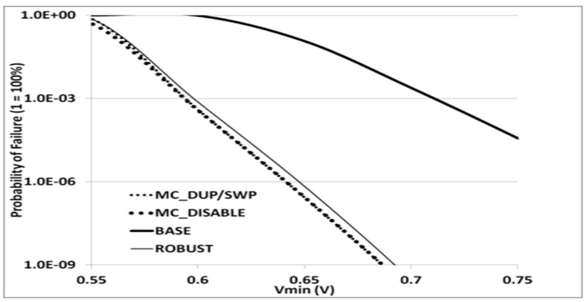

Figure 7. Vmin for all caches in baseline, all robust, and mixed-cell caches.

Mixing cell types at a sub-array level (such that each sub-array consists of only one cell type and different sub-arrays consist of different cells) avoids this problem. In fact, many of today's CPUs use different cell types for tag and data arrays. Our mixed-cell cache organizes cache ways as banks (sub-arrays). Some sub-arrays can be implemented with robust cells, and the remaining sub-arrays with smaller (non-robust) cells.

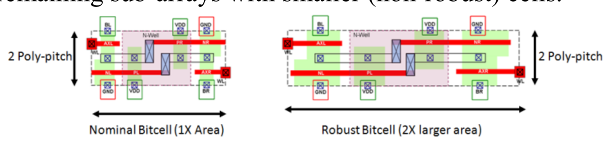

Figure 8: Upsized cell pitch-matched with nominal cell.

As mentioned earlier, larger cells come with area and power overheads. Kulkarni et al. [15][16] analyzed the impact of device upsizing on the bit cell area. When each transistor is upsized by 4X, the cell area doubles, resulting in 4X increase in bit cell leakage as well as WL/BL capacitances. In this work, we model 6-T bit cells upsized to improve Vmin. In addition to bit cell upsizing, several read/write assist techniques have been proposed to achieve low Vmin operation. Read-write assist techniques control the magnitude and the duration of different node biases (such as word-lines, bit lines, bit cell VSS node, and bit cell VCC node) [12]. Assist techniques can lower Vmin at the expense of the higher switching capacitance (C_{DYN}). A "robust cell" can be achieved by the optimal combination of bit cell size and/or assist techniques to enable lower Vmin.

5. Evaluation Methodology

5.1 Baseline Configuration

We use CMP$im [10], a Pin-based x86 simulator. Our baseline processor is 4-wide out-of order with 128-entry reorder buffer and a three-level cache hierarchy, similar to the baseline of the Cache Replacement Championship [11]. Each core has L1 split instruction and data caches, a unified L2 cache, and all cores in our 2-core and 4-core processor share the L3 cache.

Cache Configuration. We use a 32KB, 4-way L1 instruction cache, a 32KB, 8-way 32KB L1 data cache, a 256KB 8-way L2 cache (per core), and a 4MB, 16-way shared L3 cache. The L3 cache is a non-inclusive/non-exclusive cache. All caches use 64-byte lines, and implement LRU as the default replacement policy. The load-to-use latencies for the L1, L2, and L3 caches are 3, 10, and 25 cycles, respectively. We keep the shared L3 size constant between our single-core, 2-core, and 4-core configurations. This translates into larger cache capacity per core for single-core (4MB/core) and 2-core (2MB/core) configurations compared to the 4-core system (1MB/core). We assume on-die interconnects can transfer 32-bytes per cycle between the L1 and L2, and between the L2 and L3 caches. This limits our on-die bandwidth to one 64-byte line every two cycles between the L1 and L2, and between each of the L2 caches and the L3 cache. This increases network congestion and latency when writeback traffic contends with cache misses for the shared interconnect. We also model the extra latency and cache port utilization due to swaps and duplications. For each swap in MC_SWP, we assume the L1 cache is inaccessible for three cycles, and the L2 and L3 are inaccessible for six cycles. We assume a duplication in MC_DUP makes the cache inaccessible for an extra cycle (L1) or two cycles (L2 and L3).

Memory Configuration. We model a 200-cycle latency to memory at the high frequency (2.1GHz for a single processor). The memory latency (in cycles) decreases when we use lower frequencies for the 2-core and 4-core systems. For the 2-core system running at 1.4GHz, memory latency decreases to 130 cycles as the cycle time is longer. For the 4-core processor running at 825MHz, memory latency becomes 80 cycles. We support a maximum of 32 outstanding misses to memory. We implement four memory controllers per chip, each with a 6GB/sec bandwidth to memory, for a total of 24GB/sec memory bandwidth. This bandwidth is shared between all cores in the system, so the bandwidth per core is higher for our single-core (24GB/sec/core) and 2-core (12 GB/sec/core) compared to our 4-core system (6GB/sec/core).

Reliability. We assume our baseline system implements per-byte parity for the L1 cache, and

SECDED ECC for the L2 and L3 caches to guard against soft errors. Since the on-die network uses 32-byte transfers, we implement SECDED ECC on 32-bytes in the L2 and L3 caches to protect against cache and network failures. Our single-core baseline runs at 850 mV with a 2.1 GHz frequency. The 2-core system runs at 700 mV with a 1.4 GHz frequency. The 4-core system runs at 590 mV with an 825 MHz frequency.

5.2 Benchmarks

We simulate benchmarks from the SPECCPU2006 suite [22]. We use SimPoint [18] to identify a single characteristic interval (i.e., simpoint) of each benchmark. Each benchmark is run with the first reference input. We first run 50 million instructions to warm up the internal structures and then run the simulation for 100 million instructions. For multi-core simulations, each benchmark runs simultaneously with the others, restarting after 100 million instructions, until all of the benchmarks have executed at least 100 million instructions after the warmup. We use instructions-per-cycle (IPC) to measure single-core performance. For multi-core workloads, we use the weighted speedup normalized to the baseline. That is, we compute IPC_i for each application i sharing the L3 cache, then compute $IsolatedIPC_i$ for application i running in isolation with the same L3 cache size. We compute the weighted IPC of a workload as sum of its components' ($IPC_i/IsolatedIPC_i$).

For most of our results in Section 6, we use a cache-sensitive subset of SPECCPU2006 benchmarks. We selected eighteen benchmarks that see more than a 3% slowdown when either the L1 or L3 cache is reduced to a quarter of its original size. The 4-core workload mixes are randomly chosen from these eighteen benchmarks. We have ten homogeneous workloads (running four copies of the same application), and ten heterogeneous workloads (running a mix of four different benchmarks). Table 1 lists the benchmarks included in each of 4-core workloads.

Table 1. Four-Core Workloads.

Num.	Mix	Num.	Mix
1	4X 400.perlbench	2	4X 403.gcc
3	4X 416.gamess	4	4X 444.namd
5	4X 445.gobmk	6	4X 450.soplex
7	4X 453.povray	8	4X 471.omnetpp
9	4X 482.sphinx3	10	4X 483.xalancbmk
11	bzip2, gcc, soplex, xalancbmk	12	perlbench, omnetpp, sphinx3, xalancbmk
13	bzip2, astar, sphinx3, xalancbmk	14	gcc, mcf, soplex, sphinx3
15	gcc, astar, sphinx3, xalancbmk	16	perlbench, bzip2, soplex, xalancbmk
17	perlbench, gcc, mcf, namd	18	perlbench, mcf, gobmk, astar
19	perlbench, gobmk, soplex, sphinx3	20	perlbench, gobmk, omnetpp, xalancbmk

5.3 Simulated Configurations

We compared the following configurations:

Baseline (BASE). Our baseline configuration is explained in detail in Section 5.1. To compare the same area as our heterogeneous-cell designs, we increase the cache capacity in the L1 and L2 caches by 25%, and the L3 cache by 12.5%. We use a 40KB, 10-way L1 data cache, a 320KB, 10-way L2 cache, and an 18-way, 4.5 MB L3 cache. This configuration could not operate at low voltage, but we include its results for reference.

ROBUST. This configuration uses robust cells for the whole cache with no changes to the cache replacement policy. We use a 20KB, 5-way L1 data cache, a 160KB, 5-way L2 cache, and a 9-way, 2.25 MB L3 cache.

Write-through (WT). This configuration uses standard cells as the baseline, and protects modified data by propagating it down the cache hierarchy. A processor store instruction to the L1 data cache triggers a write to the L2, and a write to the L2 triggers a write to the L3 cache. This configuration has significant slowdowns due to contention for on-die and memory bandwidth.

MC_DISABLE. This configuration uses a mix of robust and non-robust cells like the proposal by Ghasemi, et. al. [7]. At low voltage, only the robust cells are used, resulting in an 8KB, 2-way L1 data cache, a 64KB 2-way L2 cache, and a 1MB 4-way L3 cache (larger area than our proposals which use only two robust L3 ways).

MC_WB. This configuration implements the mixed-cell writeback mechanism (Section 3.3). We allocate write misses to robust ways and read misses to non-robust ways. Subsequent writes to non-robust ways trigger writebacks to the next cache level or to memory.

MC_SWP. MC_SWP implements the mixed-cell swap mechanism (Section 3.3). On a write to a non-robust line, the line is swapped with a robust line, triggering a writeback from the robust line.

MC_DUP. This configuration implements the mixed-cell duplication mechanism (Section 3.3). On a write to a non-robust line, the partner non-robust line is evicted and the write goes to both the original line and its partner line. This mechanism reduces cache size when a large percentage of writes goes to non-robust lines.

6. Results

In this section, we present the experimental results for our proposal. Section 6.1 evaluates low-voltage multi-core performance and compares it to single-core performance. Section 6.2 analyzes the power and energy-efficiency of our proposals vs. prior techniques.

6.1 Performance

We evaluate the performance of single-core and multi-core systems using the same power budget (Section 2.1). A single-core runs at a higher voltage and frequency, and uses the whole L3 cache, on-die and

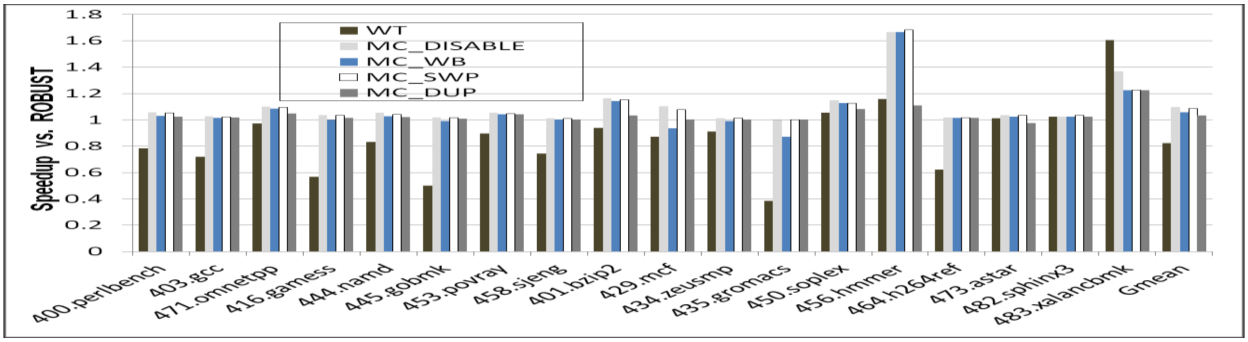

Figure 9. Single-core speedup vs. ROBUST.

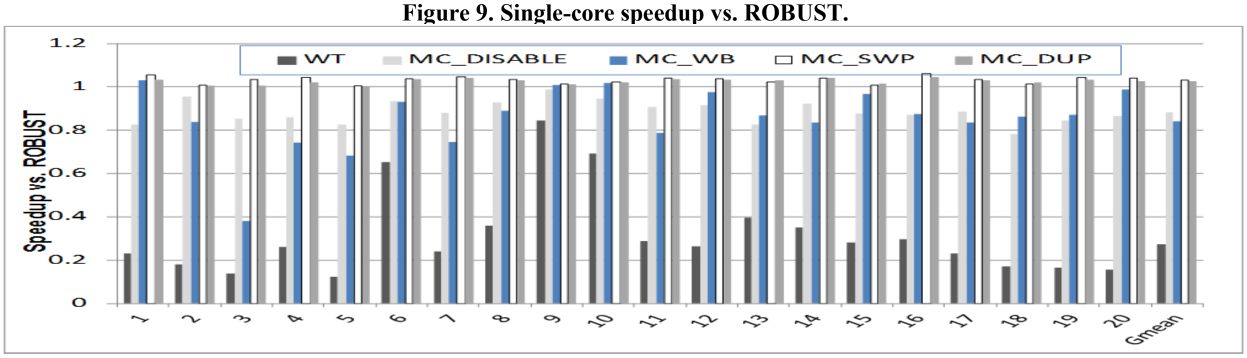

Figure 10. 4-Core Speedup vs. ROBUST for workload mixes in Table 1.

memory bandwidth. For the same power budget, 2- and 4-core configurations run at lower voltages and frequencies, and share the L3 and bandwidth resources.

Single-Core Performance for Mixed-Cell Designs. Figure 9 shows the speedup of different alternatives described in Section 5.3 compared to the ROBUST mechanism (where all the cells are robust, and the cache size is smaller than the baseline). The write-through (WT) mechanism performs 18% worse, on average, due to additional writeback traffic that increases congestion in the on-die interconnect and memory bus. The MC_DISABLE proposal in [7] has the whole cache operational at the single-processor, high-voltage configuration, so it performs 9.5% better than ROBUST. Our MC mechanisms perform similar to MC_DISABLE if we use LRU cache replacement at high voltage. However, if we use the same replacement policy as low-voltage (Section 3.2), they perform better by 5.6% (MC_WB), 8.4% (MC_SWP), and 3.4% (MC_DUP) on average.

We performed simulations to evaluate the applicability of ECC-based mechanisms in the L1 data cache. Our experiments show that increasing the L1 access latency by one cycle, from 3 to 4 cycles, degrades performance by 5%. This demonstrates the need to implement a non-ECC mechanism in the L1 to avoid latency increases and performance losses.

Multi-Core Performance for Mixed-Cell Designs. Figure 10 shows the speedup of different alternatives for a 4-core system running at low voltage, relative to the ROBUST mechanism (Section 5.3). Since multi-core performance is more sensitive to cache parameters, other mechanisms show significant slowdowns compared to just upsizing all cache lines while losing half the cache capacity. Write-through (WT) introduces a significant amount of network traffic leading to an average 73% slowdown vs. ROBUST. MC_DISABLE only operates a quarter of the cache capacity at low-voltage leading to an average 12% slowdown. While MC_WB slows down performance by 16% on average due to the extra writeback traffic, the other two mechanisms improve performance by 3.5% (MC_SWP) and 2.6% (MC_DUP). Compared to MC_DISABLE, MC_SWP improves performance by 17% while MC_DUP improves performance by 16%, on average.

Network Traffic. Figure 11 shows increases in on-die bandwidth for the alternatives discussed in Section 5.3. Figure 12 illustrates increases in memory bandwidth demand. These figures demonstrate why WT degrades performance. On average, WT increases on-die traffic and memory bandwidth (off-die) by factors of 18x and 62x. MC_WB is also bandwidth hungry, increasing on-die traffic by a factor of 5, and memory traffic by almost a factor of 9. MC_DISABLE increases traffic because of the smaller cache sizes, by 152% and 45% for on-die and memory bandwidth, respectively. In contrast, MC_SWP and MC_DUP increase on-die bandwidth by an average of 47% and 44% respectively, and memory bandwidth by 14% and 4.8%. Consider the top 4 workloads (6, 14, 17, 18) each of which make more than 20 million requests from memory, 5 times more than other workloads. For these, MC_SWP and

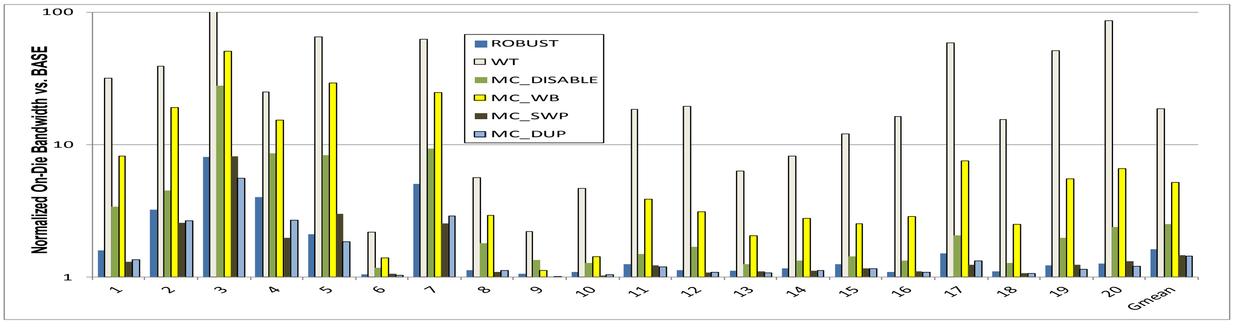

Figure 11. 4-Core On-Die Bandwidth vs. BASE (log scale) for workload mixes in Table 1.

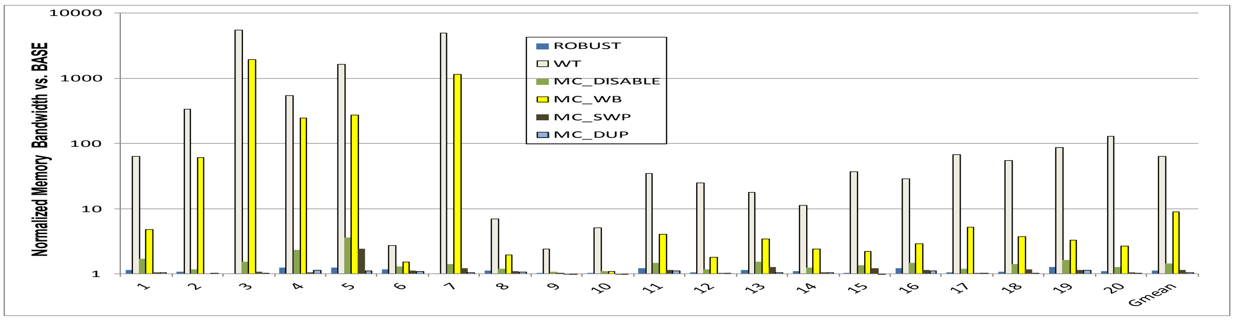

Figure 12. 4-Core Memory Bandwidth vs. BASE (log scale) for workload mixes in Table 1.

MC_DUP increase memory bandwidth by 9% and 3%, compared to 30% for MC_DISABLE. MC_WB and WT increase memory traffic by factors of 3x and 18x. When focusing on only the worst on-die traffic offenders, we observe similar trends. Based on these comparisons, MC_SWP and MC_DUP clearly improve over alternatives.

6.2 Energy Efficiency

Although our approach allows operation at a reduced voltage, the reduction in Vmin must compensate for the dynamic and static power added by the larger robust cells. In designs where one or more of the caches share a power supply with the core, reductions in cache Vmin enable reductions in core Vmin. Our mixed-cell cache reduces cache Vmin by 125mV, leading to a 50% reduction in core power, and therefore allowing us to double the number of active cores at the same power budget. However, cell upsizing adds both static power (a problem in large L3 caches), and dynamic power (a problem in L1 caches).

Static Power in L3 Cache. Figure 13 compares the static power for different alternatives in the L3 cache where static power dominates total power. We compare MC_DUP/SWP, MC_DISABLE, ROBUST, and a fourth option (separate Vcc) in which the cache has a separate power supply. The static power of the caches, normalized to the separate Vcc configuration at high voltage, is shown on the Y axis as a function of voltage (X axis). MC_DISABLE sees a significant reduction in static power at 715mV, the Vmin of the non-robust portion of the cache, due to power gating of the non-robust portion and its associated loss of cache capacity. Likewise, the reduced capacity of ROBUST compensates for increased cell size. Static power for MC_DUP and MC_SWP are equivalent and are plotted together. At Vmin, MC_DUP and MC_SWP reduce power by 10% vs. separate Vcc, without the cost of the additional power supply, or its additional inefficiencies.

Dynamic Power in L1 Cache. Figure 14 compares the dynamic power, normalized to separate Vcc at high voltage, for different alternatives in the L1 cache (with much higher dynamic power than static power). In general, both mixed-cell approaches and the ROBUST approach incur significant penalties relative to the baseline. MC_DISABLE and ROBUST both suffer from the need to service all cache requests with robust cells. In contrast, MC_SWP benefits from splitting L1 cache access between the non-robust and the power hungry robust cells. Although robust ways represent only 25% of L1 cache capacity, our experiments show that forcing writes to robust ways causes 65% of the L1 access to be handled by the robust ways. MC_DUP, with the ability to handle writes in both robust and non-robust ways handles only 35% of the L1 accesses in robust ways and consumes 30% less power than MC_SWP as a result. Overall, the dynamic power of MC_DUP is 50% better than MC_DISABLE (similar to [7]) and within 30% of the configuration with a separate power supply without the additional costs and inefficiencies of a separate supply.

7. Conclusions

Mixed-cell cache architectures enable low-voltage operation for a fraction of the cache composed of robust cells. This tradeoff allows using the entire cache when

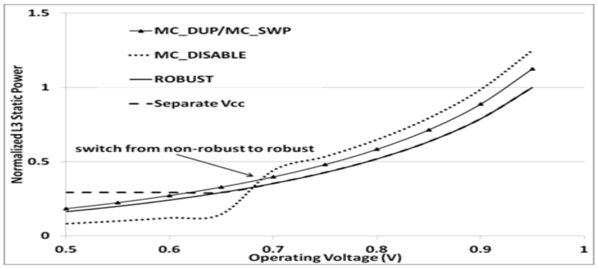

Figure 13. L3 static power comparison.

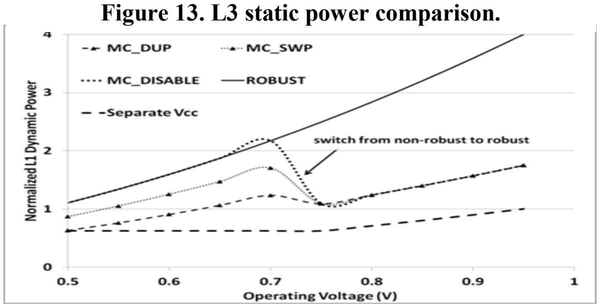

Figure 14. L1 dynamic power comparison.

operating at high voltage, but loses capacity at low voltage. In power constrained multi-core systems, the lowest voltage mode maximizes the number of active cores and therefore needs the most cache capacity.

We evaluated a power-constrained system with the ability to operate 1, 2 and 4 cores within the same power budget. To support 4 active cores, our system uses a mixed-cell cache architecture to operate at 590mV (NTV), where the 75% capacity loss experienced by our baseline mixed-cell cache architecture resulted in a 12% performance loss. We describe a novel mechanism that avoids losing cache capacity by managing cache data to allow the use of both robust and non-robust portions. Consistent with prior work, our proposal delivers a 9.5% performance benefit relative to a non-mixed cell baseline using only robust cells. However, in contrast to prior work, our design avoids significant cache size reductions at low voltage, improving multi-core performance an average of 17% and saving 50% of the L1 dynamic power.

Acknowledgements

We are very grateful to Aamer Jaleel who helped us with his simulator, CMP$im. We thank the anonymous reviewers for their helpful feedback.

References

[1] Jaume Abellà, et al., "Low Vccmin Fault-Tolerant Cache with Highly Predictable Performance", International Symposium on Microarchitecture, pp. 111 - 121, Dec. 2009.

[2] M. Agostinelli, et al., "Erratic fluctuations of SRAM cache Vmin at the 90nm process technology node," IEDM Technical Digest, pp. 655-658, Dec 2005.

[3] Amin Ansari, et al., "ZerehCache: Armoring Cache Architectures in High Defect Density Technologies", International Symposium on Microarchitecture, pp. 100 - 110, Dec. 2009.

[4] Arup Chakraborty, et al., "E < MC^2: Less Energy through Multi-Copy Cache," Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES), pp. 237-246, Oct. 2010.

[5] Zeshan Chishti, et al., "Improving Cache Lifetime Reliability at Ultra-low Voltages", International Symposium on Microarchitecture, pp. 89 - 99, Dec. 2009.

[6] Ronald G. Dreslinski, et al., "Reconfigurable Energy Efficient Near Threshold Cache Architectures," International Symposium on Microarchitecture, pp. 459-470, Dec. 2008.

[7] Hamid Reza Ghasemi, Stark Draper, and Nam Sung Kim, "Low-Voltage On-Chip Cache Architecture using Heterogeneous Cell Sizes for Multi-Core Processors," International Symposium on High-Performance Computer Architecture, pp. 38-49, Feb 2011.

[8] Intel Corp., "2nd Generation Intel® Core™Processor Family Mobile and Intel®Celeron® Processor Family Mobile," Datasheet, Vol. 1, 2011.

[9] Intel Corp., "Intel® Turbo Boost Technology in Intel® Core™ Microarchitecture (Nehalem) Based Processors," White Paper, 2008.

[10] Aamer Jaleel, et al., "CMP$im: A Pin-Based On-The-Fly Multi-Core Cache Simulator," 4th Workshop on Modeling, Benchmarking and Simulation, Beijing, China, June 2008.

[11] JILP Workshop on Computer Architecture Competitions: Cache Replacement Championship, http://www.jilp.org/jwac-1/, 2010.

[12] Muhammad M. Khellah, et al., "Read and Write Circuit Assist Techniques for Improving Vccmin of Dense 6T SRAM Cell," Conf. on Int. Circuit Design and Technology, pp. 185–189, Jun. 2008.

[13] Jangwoo Kim, et al., "Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding", International Symposium on Microarchitecture, pp. 197 - 209, Dec. 2007.

[14]Wonyoung Kim, David Brooks, and Gu-Yeon Wei, "A Fully-Integrated 3-Level DC/DC Converter for Nanosecond-Scale DVFS," IEEE Journal of Solid-State Circuits, 47(1), pp.206-219, Jan. 2012.

[15] Jaydeep Kulkarni, Keejong Kim and Kaushik Roy, "A 160 mV Robust Schmitt Trigger Based Subthreshold SRAM", IEEE Journal of Solid-State Circuits, vol. 42, no. 10, pp. 2303 - 2313, Oct. 2007.

[16] Jaydeep Kulkarni and Kaushik Roy, "Ultra-low Voltage Process Variation Tolerant Schmitt Trigger based SRAM Design", IEEE Transactions on VLSI Systems, 2011.

[17] Wai-Kei Mak and Jr-Wei Chen, "Voltage Island Generation under Performance Requirement for SoC Designs," Asia & South Pacific Design Automation Conference, Jan. 2007.

[18] Erez Perelman, Greg Hamerly, and Brad Calder, "Picking Statistically Valid and Early Simulation Points," International Conference on Parallel Architectures and Compilation Techniques, pp. 244-255, Sep. 2003.

[19] David Roberts, Nam Sung Kim and Trevor Mudge, "On-Chip Cache Device Scaling Limits and Effective Fault Repair Techniques in Future Nanoscale Technology", Digital System Design Architectures, Methods and Tools, pp. 570 - 578, Aug. 2007.

[20] Efi Rotem, et al., "Power Management Architecture of the 2nd Generation Intel® Core™ Microarchitecture, Formerly Codenamed Sandy Bridge", HotChips, Aug. 2011.

[21] Stanley Schuster, "Multiple Word/Bit Line Redundancy for Semiconductor Memories", IEEE Journal of Solid-State Circuits, vol. 13, no. 5, pp. 698 - 703, Oct. 1978.

[22] SPEC CPU2006 Benchmarks, http://www.spec.org/cpu2006/

[23] Chris Wilkerson, et al., "Reducing Cache Power with Low Cost, Multi-bit Error-Correcting Codes", International Symposium on Computer Architecture, pp. 83-93, Jun. 2010.

[24] Chris Wilkerson, et al., "Trading off Cache Capacity for Reliability to Enable Low Voltage Operation", International Symposium on Computer Architecture, pp. 203 - 214, Jun. 2008.

[25] Doe Hyun Yoon and Mattan Erez, "Memory Mapped ECC: Low-Cost Error Protection for Last Level Caches", International Symposium on Computer Architecture, pp. 116-127, Jun., 2009.

[26] Marcelo Yuffe, et al., "A Fully Integrated Multi-CPU, GPU and Memory Controller 32nm Processor", International Solid-State Circuits Conference, pp. 264-266, Feb, 2011.

[27] Wei Zhang, et al., "ICR: In-Cache Replication for Enhancing Data Cache Reliability," International Conference on Dependable Systems and Networks, pp. 291-300, Jun. 2003.