DPCS: Dynamic Power/Capacity Scaling for SRAM Caches in the Nanoscale Era

MARK GOTTSCHO, University of California, Los Angeles
ABBAS BANAIYANMOFRAD, NIKIL DUTT, and ALEX NICOLAU,
University of California, Irvine
PUNEET GUPTA, University of California, Los Angeles

Fault-Tolerant Voltage-Scalable (FTVS) SRAM cache architectures are a promising approach to improve energy efficiency of memories in the presence of nanoscale process variation. Complex FTVS schemes are commonly proposed to achieve very low minimum supply voltages, but these can suffer from high overheads and thus do not always offer the best power/capacity trade-offs. We observe on our 45nm test chips that the "fault inclusion property" can enable lightweight fault maps that support multiple runtime supply voltages.

Based on this observation, we propose a simple and low-overhead FTVS cache architecture for power/capacity scaling. Our mechanism combines multilevel voltage scaling with optional architectural support for power gating of blocks as they become faulty at low voltages. A static (SPCS) policy sets the runtime cache VDD once such that a only a few cache blocks may be faulty in order to minimize the impact on performance. We describe a Static Power/Capacity Scaling (SPCS) policy and two alternate Dynamic Power/Capacity Scaling (DPCS) policies that opportunistically reduce the cache voltage even further for more energy savings.

This architecture achieves lower static power for all effective cache capacities than a recent more complex FTVS scheme. This is due to significantly lower overheads, despite the inability of our approach to match the min-VDD of the competing work at a fixed target yield. Over a set of SPEC CPU2006 benchmarks on two system configurations, the average total cache (system) energy saved by SPCS is 62% (22%), while the two DPCS policies achieve roughly similar energy reduction, around 79% (26%). On average, the DPCS approaches incur 2.24% performance and 6% area penalties.

Categories and Subject Descriptors: B.3.1 [**Memory Structures**]: Design Styles—*Cache memories*; B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault Tolerance; B.3.2 [**Memory Structures**]: Semiconductor Memories—*Static memory (SRAM)*

General Terms: Design, Performance, Reliability

Extension of Conference Paper. An earlier version of this work was published in *Proceedings of the 51st ACM/IEEE Design Automation Conference (DAC)* [Gottscho et al. 2014; Dutt et al. 2014]. Summary of new material: (1) study of the "fault inclusion property" using 45nm test chips; (2) discussion of the "power versus effective capacity" metric; (3) two alternate DPCS policies; (4) improved models, methodology, and evaluation; and (5) revised written content with more detail. For correspondence, please contact M. Gottscho at *mgottscho@ucla.edu*. This work was supported by the U.S. National Science Foundation, under grants CCF-1029783 and CCF-1029030.

Authors' addresses: M. Gottscho, NanoCAD Lab, 53-109 Engineering IV Building, University of California, Los Angeles, Los Angeles, CA 90095; email: mgottscho@ucla.edu; A. BanaiyanMofrad, 9920 Pacific Heights Blvd., Suite 150, San Diego, CA 92121; email: abbas@abreezio.com; N. Dutt, Department of Computer Science, Zot Code 3435, Donald Bren School of Information and Computer Science, University of California, Irvine, Irvine, CA 92697-3435; email: dutt@uci.edu; A. Nicolau, Center for Embedded Computing Systems, 944 Computer Science Building, University of California, Irvine, Irvine, CA 92697-3435; email: nicolau@ics.uci.edu; P. Gupta, 6730C Boelter Hall Building, University of California, Los Angeles, Los Angeles, CA 90095; email: puneet@ee.ucla.edu.

DOI: http://dx.doi.org/10.1145/2792982

Additional Key Words and Phrases: Process variation, variability-aware, resizable cache, energy proportionality, fault-tolerant voltage scaling, block disable, low power, nanoscale technology

ACM Reference Format:
Mark Gottscho, Abbas BanaiyanMofrad, Nikil Dutt, Alex Nicolau, and Puneet Gupta. 2015. DPCS: Dynamic power/capacity scaling for SRAM caches in the nanoscale era. ACM Trans. Architec. Code Optim. 12, 3, Article 27 (August 2015), 26 pages.
DOI: http://dx.doi.org/10.1145/2792982

1. INTRODUCTION

Moore's Law has been the primary driver behind the phenomenal advances in computing capability of the past several decades. Technology scaling has now reached the nanoscale era, where the smallest integrated circuit features are only a couple of orders of magnitude larger than individual atoms. In this regime, increased leakage power and overall power density has ended ideal Dennard scaling, leading to the rise of "dark silicon" [Esmaeilzadeh et al. 2011].

Owing to the extreme manufacturing control requirements imposed by nanoscale technology, the effects of process variation on reliability, yield, power consumption, and performance is a principal challenge [Gupta et al. 2013] and is partly responsible for the end of Dennard scaling. The typical solution is to leave design margins for the worst-case manufacturing outcomes and operating conditions. However, these methods naturally incur significant overheads.

Memory structures are especially sensitive to variability. This is because memories typically use the smallest transistors and feature dense layouts, comprise a significant fraction of chip area, and are sensitive to voltage and temperature noise. Memories are also major consumers of system power, are not very energy proportional [Barroso and Hölzle 2007, 2009], and are frequent points of failure in the field [Schroeder et al. 2011; Sridharan and Liberty 2012]. It is clear that the design of the memory hierarchy needs to be reconsidered with the challenges brought about in the nanoscale era.

One way to reduce the power consumption of memory is to reduce the supply voltage (VDD). Static power, dominated by subthreshold leakage current, is a major consumer of power in memories [Flautner et al. 2002] and has an exponential dependence on supply voltage [Weste and Harris 2011]. Since leakage often constitutes a major fraction of total system power in nanoscale processes [Kumar et al. 2009], even a minor reduction in memory supply voltage could have a significant impact on total chip power consumption.

Unfortunately, voltage-scaled SRAMs are susceptible to faulty behavior. Variability in SRAM cell noise margins, mostly due to the impact of random dopant fluctuation on threshold voltages, results in an exponential increase in probability of cell failure as the supply voltage is lowered [Wang and Calhoun 2011]. This has motivated research on Fault-Tolerant Voltage-Scalable (FTVS) SRAM caches for energy-efficient and reliable operation.

Many clever FTVS SRAM cache architectures use complex fault-tolerance methods to lower min-VDD while meeting manufacturing yield objectives. Unfortunately, these approaches are limited by a manifestation of Amdahl's Law [Amdahl 1967] for power savings. Large and flexible fault maps combined with intricate redundancy mechanisms cost considerable power, area, and performance. It is important that designers account for these overheads, as they ultimately limit the scalability of such approaches. In this work, we make several contributions:

(1) The SRAM memories on our custom 45nm SOI embedded systems on chip are characterized. We observe the *fault inclusion property*: particular bit cells that fail at a given supply voltage will still be faulty at all lower voltages.

(2) A new *static power versus effective capacity* metric is proposed for FTVS memory architectures, which is a good indicator of runtime power/performance scalability and overall energy efficiency.
(3) A simple and low-overhead multi-VDD fault-tolerance mechanism combines voltage scaling of the data array SRAM cells with optional power gating of faulty data blocks at low voltage.
(4) We propose Static Power/Capacity Scaling (SPCS) and Dynamic Power/Capacity Scaling (DPCS), two variants of a novel FTVS scheme that significantly reduces overall cache and system energy with minimal performance and area overheads.
(5) SPCS and DPCS achieve lower static power at all scaled cache capacities than a more complex FTVS cache architecture [BanaiyanMofrad et al. 2011] as well as per-cache way power gating.

2. RELATED WORK

There is a rich body of literature in circuit, architecture, and error correction coding techniques for FTVS deep-submicron memories. A summary of related work is provided here, with emphasis on architectural solutions. The reader may refer to Mittal [2014] for a more complete survey of architectural techniques for improving cache power efficiency.

2.1. Leakage Reduction

Two of the best-known architectural approaches are those generally based on Gated-VDD [Powell et al. 2000; Cheng et al. 2014] and Drowsy Cache [Flautner et al. 2002; Bardine et al. 2014]. The former dynamically resizes the instruction cache by turning off blocks that are not used by the application, exploiting variability in cache utilization within and across applications. The latter utilizes the alternative approach of voltage scaling idle cache lines, which yields good static power savings without losing memory state. Neither approach improves dynamic power nor accounts for the impact of process variation on noise margin-based faults, which are greatly exacerbated at low voltage [Kumar et al. 2009; Wang and Calhoun 2011]. This issue particularly limits the mechanism of Flautner et al. [2002].

2.2. Fault Tolerance

In the fault tolerance area, works targeting cache yield and/or min-VDD improvement include Error Correction Codes (ECCs) and architectural methods [Shirvani and McCluskey 1999; Agarwal et al. 2005; Ozdemir et al. 2006; Kim et al. 2007; Koh et al. 2009; Ansari et al. 2009; Rossi et al. 2011; Alameldeen et al. 2011a, 2011b; Qureshi and Chishti 2013], none of which explicitly address energy savings as an objective. A variety of low-voltage SRAM cells that improve read stability and/or writability have also been proposed, for example, 8T [Chang et al. 2005] and 10T [Calhoun and Chandrakasan 2006], but they have high area overheads compared to a 6T design.

Schemes that use fault tolerance to achieve lower voltage primarily for cache power savings include Hussain et al. [2008], Wilkerson et al. [2008], Abella et al. [2009], and Sasan et al. [2009], two very similar approaches from Ansari et al. [2011] and BanaiyanMofrad et al. [2011], and several others [Sasan et al. 2012; Zhang et al. 2012; Han et al. 2013; Chakraborty et al. 2014; Mahmood et al. 2014]. All of these approaches try to reduce the minimum-operable cache VDD with yield constraints by employing relatively sophisticated fault-tolerance mechanisms, such as address remapping, block and set-level replication, etc. They are also similar in that they either reduce the effective cache capacity by disabling faulty regions as VDD is reduced (e.g., FFT-Cache [BanaiyanMofrad et al. 2011]), or boost VDD in "weak" regions as necessary to maintain capacity (e.g., Sasan et al. [2012]). Han et al. [2013] and Kim and Guthaus [2013]

both utilize multiple memory supply voltages. Ghasemi et al. [2011] proposed a last-level cache with heterogeneous cell sizes for more graceful voltage/capacity trade-offs. Finally, variation-aware Non-uniform Cache Access (NUCA) architectures have been proposed [Wang et al. 2013; Hijaz and Khan 2014]. The approach by Hijaz and Khan [2014] mixes capacity trade-offs with latency penalties of error correction.

There are two recent papers that contain some similarities to the concepts in this article, yet were developed concurrently and independently of our [Mohammad et al. 2014; Ferreron et al. 2014]. Mohammad et al. [2014] discuss power/capacity trade-offs in FTVS caches and a similar architectural mechanism, primarily focusing on yield improvement and quality/power trade-offs. Ferreron et al. [2014] focuses on system-level impacts of block disabling techniques in a chip multiprocessor with shared caches and parallel workloads.

2.3. Memory Power/Performance Scaling

With energy proportionality becoming a topic of interest recent [Barroso and Hölzle 2007, 2009], there have been several works that target main memory. Fan et al. [2005] studied the coordination of processor DVFS and memory low-power modes. In MemScale [Deng et al. 2011], the authors were some of the first to propose dynamic memory frequency scaling as an active low-power mode. Independently of MemScale, David et al. [2011] also proposed memory DVFS. MemScale was succeeded by CoScale [Deng et al. 2012], which coordinated DVFS in the CPU and memory to deliver significantly improved system-level energy proportionality.

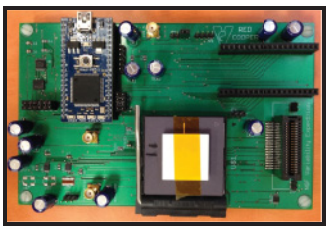
2.4. Novelty of This Work

Our approach is different from the related works as follows. The circuit mechanisms are similar to those of Gated-VDD [Powell et al. 2000] and Drowsy Cache [Flautner et al. 2002], but are combined with fault tolerance to allow lower voltage operation using 6T SRAM cells. However, the proposed scheme can also be used with any other cell design. To the best of our knowledge, our scheme is the first to use voltage scaling on data bit cells only, along with *optional* power gating blocks as they become faulty for additional energy savings. Due to architecture, floor planning, and layout considerations imposed by per-block power gating, we also discuss the advantages and disadvantages of this feature. Our approach emphasizes simplicity to get good energy savings using low-overhead multi-VDD fault maps that can be used for static (SPCS) or dynamic (DPCS) power/capacity scaling. SPCS and DPCS both achieve dynamic and static energy savings, as VDD is not boosted for cache accesses.

This work could be supplemented with other innovations. Although soft errors are not currently handled, this scheme can be supplemented with ECC. Circuit-level approaches to coping with aging are orthogonal to this work and could be incorporated as well. Our insights could be augmented with those of Mohammad et al. [2014] and Ferreron et al. [2014] to enable knobs for dynamic power/capacity/quality scaling. Although not explored in this article, we believe that DPCS can also be used to improve system-level energy proportionality by coordinating with existing CPU and main memory DVFS approaches. We leave these possibilities to future work.

3. THE SRAM FAULT INCLUSION PROPERTY

Once the noise margin of a memory cell collapses at low voltage, continuing to reduce voltage further will not restore its functionality. We refer to this behavior as the *fault inclusion property*: any faults that occur at a given supply voltage will strictly be a



(a) Custom "Red Cooper" 45nm SOI test chip and board

(b) 550mV

(d) 500mV

(c) 525mV

(e) 475mV

Fig. 1. Graphical depiction of faulty byte locations, represented as vertical orange rectangles, illustrating the fault inclusion property. Results are from a 45nm SOI test chip SRAM bank. Similar trends were observed on our other test chips.

subset of those at all lower voltages. Put more formally:

$$\begin{aligned}
&\text{Let } f_i(v) = \begin{cases} 1, & \text{if memory cell } i \text{ is faulty at supply voltage } v \\ 0, & \text{otherwise,} \end{cases} \\
&\text{where } 1 \le i \le n, \text{ and } n \text{ is the number of bit cells in the memory.} \\
&\text{Fault Inclusion Property: } f_i(v) \ge f_i(u) \text{ for } v \le u. \\
&\text{Let the fault map } \boldsymbol{F}(v) = (f_1(v), f_2(v), \ldots, f_i(v), \ldots, f_n(v)) \\
&\text{be a length-}n \text{ bit-vector. Then } w[\boldsymbol{F}(v)] \ge w[\boldsymbol{F}(u)] \\
&\text{for } v \le u, \text{ where } w[\cdot] \text{ denotes the Hamming weight.}
\end{aligned} \tag{1}$$

No prior work on FTVS SRAM cache architectures appears to take advantage of this property. With typical existing approaches, there is no way to operate the memory at additional intermediate voltage levels without including extra fault maps for each runtime voltage level. *A key insight in this work is that complete fault map duplication for each runtime supply voltage is unnecessary*.

To verify the previous formulation, tests were performed on four ARM Cortex M3-based "Red Cooper" test chips manufactured in a commercial 45nm SOI technology [Lai and Gupta 2014; Agarwal et al. 2014] in collaboration with colleagues in the NSF Variability Expedition. Each test chip had two 8kB SRAM scratchpad banks and no off-chip RAM. A board-level mbed LPC1768 microcontroller accessed the SRAM and controlled all chip voltage domains through JTAG. The board and a test chip are shown in Figure 1(a).

With each test chip's CPU disabled, March Simple-Static (SS) tests [Hamdioui et al. 2002] were run on both banks using the mbed to characterize the nature of faults as the array VDD was reduced. For each SRAM bank on each chip, a test suite was repeated five times. In each run, VDD scaled down one step from nominal 1V in 25mV increments. At each voltage level, faulty SRAM locations were logged at byte granularity.[1] No distinction was made between different underlying physical causes of failure (read stability, writability, retention failure, inter-cell interference, etc.). As dynamic failures are not the focus of this study, the test chip was operated at a 20MHz clock to minimize the chance of delay faults occurring in the SRAM periphery logic. This is because the test chip uses a single voltage rail for the memory cells and periphery.

As expected, faults caused by voltage scaling obeyed the fault inclusion property. This trend is depicted graphically in Figures 1(b)–(e) for one of the test chips. Faulty byte locations were consistent in each unique SRAM bank at each voltage level. The

[1]Due to runtime and storage limitations of our mbed-based testing, we manually verified the fault inclusion property at the bit level, while recording fault maps at the byte level.

patterns could be verified with repeated testing after compensating for noise, indicating that the faults were not caused by soft errors or permanent defects, but rather caused by variability in cell noise margins. These results suggested the use of a compact fault map such that multiple VDDs can be efficiently supported.

4. A CASE FOR THE "POWER VERSUS EFFECTIVE CAPACITY" METRIC

As previously mentioned, related works in FTVS cache architecture commonly use min-VDD as a primary metric of evaluation. While supply voltage is a very important control knob for power, it is only one factor that determines the static and dynamic energy consumption of a cache memory. Most FTVS schemes, including ours, require a portion of the memory to operate at full VDD or otherwise be resilient for guaranteed correctness.

4.1. Amdahl's Law Applied to Fault-Tolerant Voltage-Scalable Caches

When evaluating power reduction from voltage scaling, it is important to consider the proportion of cache components operating at high VDD. This is due to a manifestation of Amdahl's Law [Amdahl 1967] when interpreted for power and energy. Note the following relationship, which is similar to the traditional version of Amdahl's Law for speedup [Hennessy and Patterson 2012], but for power reduction instead via Fault-Tolerant (*FT*) Voltage Scaling (*VS*).

$$PowerReduction_{FTVS,\,overall} = \frac{1}{1 - Fraction_{VS} + Fraction_{FT\,overhead} + \frac{Fraction_{VS}}{PowerReduction_{VS}}}. \quad (2)$$

Note the additional $Fraction_{FT\,overhead}$ term in the denominator, which accounts for the additional fault-tolerance logic needed to scale voltage to the desired min-VDD on part of the memory ($Fraction_{VS}$). In fact, this modified formulation of Amdahl's Law also holds more generally whenever constant overheads are incurred for speedup or power reduction. This interpretation of Amdahl's Law might be overlooked when trying to achieve a lower min-VDD using complex fault-tolerance schemes. Thus, supply voltage alone should not receive too much emphasis in FTVS techniques.

For example, consider two competing FTVS approaches, *Scheme 1* and *Scheme 2*, where only the data array voltage can be scaled. Both *Scheme 1* and *Scheme 2* have the same cache size, block size, associativity, etc. Assume *Scheme 2*'s total tag array power overhead is 20% of the nominal data array power (full VDD) due to a large and complex fault map, and *Scheme 1*'s tag power overhead is 5%, thanks to a smaller and simpler fault map. The baseline cache has a tag array power overhead of only 3% because it has no fault map.

Let the data array voltage be scaled independently for *Scheme 1* and *Scheme 2* such that *Scheme 2*'s voltage is lower than *Scheme 1*'s due to better fault tolerance. Suppose the data array leakage power in *Scheme 1* is now 30% of its nominal value, and the data array leakage power in *Scheme 2* is now 20% of its nominal value. *Scheme 2* will save 61.1% of static power against the baseline cache, while *Scheme 1* will save 66.0%, despite operating at a higher data array voltage.

4.2. Problems With Yield Metric

Another common metric is memory yield in terms of functional reliability, performance, power, etc. Often, the min-VDD is determined for a particular cache configuration based on expected fault probabilities, the particular fault-tolerance mechanism, and a desired target yield such as 99%. While one can claim that a particular scheme is functional within the design envelope, definitions of functionality vary considerably. For example, one cache architecture may be considered functional if it simply operates in a "correct" manner, regardless of power consumption. Another architecture may define the cache

to be functional only if at least 50% of blocks are non-faulty. Thus, in general, min-VDD/yield should not be over-emphasized. The min-VDD metric can be more useful for caches that are on the same voltage domain as the processor core. However, in the future, as indicated by industry trends such as Intel's fine-grain voltage regulation (FIVR) design in its Haswell architecture [Bowhill et al. 2015], this is likely not to be the case.

4.3. Proposed Metric

We believe that a new metric, *static power versus effective capacity*, should be used to guide the design of FTVS cache architectures in addition to the other metrics. This metric accounts for the supply voltage as well as the efficacy and overheads of the particular fault tolerance or capacity reduction mechanism (e.g., power gating).

Simple FTVS schemes can fare similarly or even better than complex ones in terms of power, performance, and/or area, all with less design and verification effort. This is because the overall failure rate rises sharply in a small voltage range as shown earlier in Figure 1. The complexity required for tolerating such a high memory failure rate may not be worth the small array power savings from an incrementally lower voltage.

With this metric, we focus on static power to allow for analytical evaluation. This is a reasonable simplification in the case of large memories where static power constitutes a large fraction of memory energy. We apply this metric during our evaluation in Section 7.

Note that static power versus effective capacity is not necessarily a good metric for choosing the capacity of a cache at design-time and nominal voltage. Nor does the metric imply that smaller caches are more efficient in general. Rather, this metric is meant to capture the *runtime scalability* of the cache power and performance while operating below its designed maximum capacity. This is useful because the full cache capacity may not be needed at all times to deliver good performance, and hence, its capacity can be temporarily reduced to lower power and improve energy efficiency.

5. POWER/CAPACITY SCALING ARCHITECTURE

Our scheme has two main components: the architectural mechanism, described in Section 5.1, and the policy. We propose a static policy (SPCS) and two different dynamic policies (DPCS Policy 1 and DPCS Policy 2), described in Sections 5.2, 5.3.1, and 5.3.2, respectively. All three policies share the same underlying hardware mechanism presented next.

5.1. Mechanism

Industry trends point toward finer-grain voltage and frequency domains in future chips. For example, Intel has introduced fine-grain voltage regulation (FIVR) that is partly on-chip and on-package with its Haswell architecture [Bowhill et al. 2015]. This allows for many voltage domains on the chip on the level of per-core, per-L3 cache, etc. Thus, future chips might support separate voltage rails for each level of cache in order to further decouple logic V_{min} from memory V_{min}, especially if architectures can exploit this feature.

The power/capacity scaling mechanism primarily consists of a lightweight fault map and a circuit for globally adjusting the VDD of the data cells. The data array periphery, metadata (*Valid*, *Dirty*, *Tag*, etc.) array cells, metadata array periphery, and the processor core are all on a separate voltage domain running at nominal VDD, where they are assumed to be never faulty. Data and metadata arrays otherwise use identical cell designs. Voltage is not boosted for data access, granting both dynamic and static energy savings. To bridge the voltage domains of the data array with its periphery, the final stage of the row decoder is also used as a downward level-shifting gate to drive the wordline. Similarly, column write drivers also are downward level shifting, while sense amplifiers used in read operations restore voltage levels to the full nominal VDD

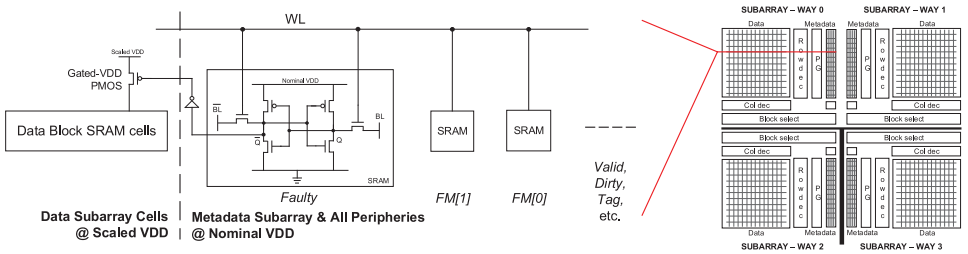


Fig. 2. Power/capacity scaling cache architecture concept with optional power gating of faulty blocks. If this feature is omitted by the designer, the *Faulty* bit SRAM cell does not drive a Gated-VDD transistor for the data block and the cache has relaxed floor plan and layout constraints.

swing. This circuit approach is validated by a very recent SoC design from Samsung [Pyo et al. 2015], which independently arrived at the decision to employ a dual-rail design to decouple logic and bitcell supply voltage.

Overall access time may be affected by up to 10% in the worst case at low voltage, as found later in Section 6.3. This is because the impact of reduced cell voltage is only one part of the overall cache access time, and near-threshold operation is avoided. Note that voltage boosting during cell access could be utilized to allow even lower array voltages than used in this work, as the proposed design is limited by SRAM read stability.

If the number of voltage domains remains constrained, a version of the architecture with a single voltage rail for the core and all peripheries and a shared voltage rail for all cache data arrays is also feasible. However, this can restrict DPCS capabilities for energy savings by coupling cache voltages together while limiting voltage scaling policies. The overall architecture is depicted in Figure 2.

5.1.1. Fault Map. The low-overhead fault map includes two fields for each data block that are maintained in the corresponding nearby metadata subarray in addition to the conventional bits (*Valid*, *Dirty*, *Tag*, etc.), as shown in Figure 2. The first entry is a single *Faulty* bit, which indicates whether the corresponding data block is presently faulty. Blocks marked as *Faulty* can never contain valid or dirty data and are unavailable for reading and writing. The second field consists of several fault map (*FM*) bits, which encode the lowest nonfaulty VDD for the data block. For V allowed data VDD levels, $K = \lceil log_2(V + 1) \rceil$ *FM* bits are needed to encode the allowed VDD levels (assuming the fault inclusion property). Figure 2 depicts the $V = 3$ configuration for a small four-way cache, requiring one *Faulty* and two *FM* bits per block.

5.1.2. Power Gating of Faulty Blocks. Power gating transistors can be used to attain additional power savings for faulty data blocks that occur at reduced voltage. Blocks may span one or more subarray rows. When a block's *Faulty* bit is set in the metadata array, a downward level-shifting inverter power gates the block's data portion. We assume that the power gating implementation is the gated-PMOS configuration from Powell et al. [2000], chosen because it has no impact on cell read performance, negligible area overhead [Powell et al. 2000; Cheng et al. 2014], and good energy savings. A power-gated block at low VDD is modeled as having zero leakage power. This is a reasonable assumption because the block would likely be power gated at a low voltage that caused it to be faulty in the first place. We verified the functional feasibility of this mechanism via SPICE simulations.

Designers might choose to omit power gating of faulty blocks for two major reasons. First, if power gating is used, the metadata subarrays should be directly adjacent to their corresponding data subarrays as shown by Figure 2. This is so the *Faulty* bit can control the power gate mechanism and the row decoder can be shared between

subarrays, although the latter benefit is not explicitly modeled. However, this constrains the cache floor plan.

Second, *by allowing for per-block power gating, both power rails must be routed in the wordline direction.* Unfortunately, this is not a common approach in industry, where a thin-cell SRAM cell is preferred for above-threshold operation (e.g., as seen by layouts in Sinangil et al. [2011] and Ebrahimi et al. [2015]). However, others have used wordline-oriented rails successfully in a variety of low-voltage design scenarios [Calhoun and Chandrakasan 2006; Verma and Chandrakasan 2008; Singh et al. 2008; Cheng et al. 2014; Chang et al. 2015]. Nevertheless, there are other important factors to consider when deciding on power rail orientation.

Without advocating for either approach, we believe that fine-grain power gating of faulty blocks could be viewed as an additional benefit from wordline-oriented rails. Without loss of generality, in the rest of the article we assume the presence of per-block power gating, except in Sections 7.2 and 8.2, where we directly compare the two approaches.

5.1.3. Transitioning Data Array Voltage. It is the responsibility of the power/capacity scaling policy, implemented in the cache controller, to ensure the *Faulty* and *Valid* bits are set correctly for the current data array VDD. After all blocks' *Faulty* bits are properly set for the target voltage, the data array VDD transition can occur. The controller must compare each block's *FM* bits with an enumerated code for the intended VDD level. Equation (3) captures this logical relationship for all V. We assume that $V = 3$ levels are allowed throughout this article, corresponding to $K = 2$ *FM* bits. Note that our fault map approach scales well for more intermediate voltage levels if needed.

$$Block_i.Faulty = TRUE \text{ iff } (VDD_{data,\,global}[K-1{:}0] \leq Block_i.FM[K-1{:}0]). \quad (3)$$

The procedure to scale array VDD is described by Algorithm 1. A voltage transition has a delay penalty to update the *Faulty* bits and then set the voltage. The cache controller must read out the entire fault map set by set, with each way in parallel. Next, it compares the *FM* bits for the set with the target array *VDD* bits. Finally, it writes back the correct *Faulty* bits for each block in the set. We assume that it takes two cycles to do this for each set. After all *Faulty* bits are updated, the voltage regulator can adjust the data array VDD, which takes *VoltageRegulatorDelay* clock cycles. The total *VoltageTransitionPenalty* in Algorithm 1 is then equal to $2*NumSets+$ *VoltageRegulatorDelay* clock cycles. Note that *VoltageTransitionPenalty* in Algorithm 1 does not include the time to write back any cache blocks. The performance and energy impact of any such writebacks are captured accurately in our modified gem5 [Binkert et al. 2011] cache model and simulation framework leveraged in Section 8.

5.1.4. Cache Operation In Presence of Faulty Blocks. Since some blocks may be faulty at any given supply voltage, the cache controller must consult the fault information during an access. Neither a hit nor a block fill are allowed to occur on a block that is marked as *Faulty*. Thus, all blocks that are currently *Faulty* must be marked as not *Valid* and not *Dirty*, and their *Tag* bits should be zeroed. When fulfilling a cache miss, the traditional LRU scheme is applied on each set, with the special condition that any blocks marked as *Faulty* are omitted from LRU consideration. This can change the effective associativity on each set.

5.1.5. 3C+F: Compulsory, Capacity, Conflict, and Faulty Misses. Cache misses can be categorized into three buckets, commonly known as the "3 C's": compulsory, capacity, and

ALGORITHM 1: Voltage Transition Procedure

Input: *NextVDD* (integer index), *VoltageTransitionPenalty* (integer, in cycles)
Output: SRAM data array supply voltage scaled corresponding to *NextVDD* after delay of *VoltageTransitionPenalty* clock cycles

```
Halt cache accesses (by buffering or stalling);
for each Set in CacheSets do
    for PARALLEL[Assoc] Block in Set do
        Read metadata bits for Block;
        if NextVDD[1:0] ≤ Block.FM[1:0] then
            if Block.Valid and Block.Dirty then
                Write back Block;
            end
            Invalidate Block;
            Block.Faulty = TRUE;
        else
            if Block.Faulty then
                Block.Faulty = FALSE;
            end
        end
    end
end
CurrVDD = NextVDD;
Scale cache data array SRAM cells to voltage corresponding to CurrVDD;
```

conflict misses [Hennessy and Patterson 2012].[2] Note that in practice, it is not feasible to classify misses in this fashion at runtime, as the full access history for all referenced data must be maintained.

When disabling faulty blocks, it is not clear how resulting misses should be classified. One pitfall would be to simply attribute misses caused by faulty blocks as capacity misses. However, faulty blocks also reduce local associativity. Moreover, consider blocks that are only temporarily faulty during execution with DPCS. An evicted block from a faulty set might be referenced later, causing a miss, even though the set may no longer contain faulty blocks.

Thus, another category of misses can be defined for theoretical purposes. A *faulty miss* is one that occurs because the referenced data was evicted or replaced earlier from a set containing faulty blocks. To be considered a faulty miss, the reference should otherwise have hit in a cache where faulty blocks never previously occurred in the set. Note that disabling faulty blocks in the L1 cache can affect miss behavior in the L2 cache. For simplicity, we ignore intercache dependencies in the 3C+F classification.

5.1.6. Variation-Aware Selection of Allowed Runtime Voltage Levels. The proposed mechanism requires that each set must have at least one non-faulty block at all allowed voltages, as there is no set-wise data redundancy. This is the constraint that limits the min-VDD.[3] Higher associativity and/or smaller block sizes naturally result in lower min-VDD as shown later in Section 8.1, but they incur other design trade-offs. Nevertheless, as demonstrated later in the evaluation, a good power/capacity trade-off can be achieved even for four-way caches with blocks of 64 bytes.

The allowed runtime voltage levels for the SPCS and DPCS policies may be decided at design time by margining for yield, at test time to opportunistically remove some yield constraints, or at runtime, which can also account for aging effects. However, the

[2]In this work, for simplicity, we do not consider multiprocessor machines, so the "fourth C" for coherence misses is not included.
[3]If uncacheable blocks are permitted, then the voltage can be lowered further.

design-time approach falls short of the other two, as it cannot exploit the individualized outcome of each chip. The test-time approach might considerably increase manufacturing cost. In this work, runtime choice of VDD levels is used (as opposed to the design-time choice from our prior work [Gottscho et al. 2014]). This eliminates yield margins by customizing the desired voltage levels based on the manifested fault patterns on each SRAM array. Note that voltage margins may still be left for noise resilience.

5.1.7. Populating and Maintaining Fault Maps with BIST. To populate the cache fault maps, one can use any standard Built-In Self-Test (BIST) routine that can detect faults with minimum granularity of a single block (e.g., the march tests from Section 3). The BIST routine can then apply the fault inclusion property to compress the representation of the fault maps by only encoding the minimum nonfaulty voltage for each block.

A drawback of fault map approaches in general is the BIST runtime and storage overhead. The proposed approach incurs longer testing time than typical single-voltage solutions. The overall testing time is $O(N * V)$, where N is the size of the cache, and V is the number of runtime voltage levels. However, the actual runtime complexity is likely less than this, as fewer blocks must be tested at successively lower voltages due to the fault inclusion property.

One approach is to run BIST at every system startup, which requires no permanent fault map storage and allows aging effects such as Bias Temperature Instability (BTI) to be considered. Particular data patterns stored in the SRAM cells are not expected to have a major impact on aging caused by BTI. This is because data tends to only briefly "live" in the cache and is assumed to be random across many workloads and over time. If data dependence does result in a noticeable impact on aging, the DPCS architecture can compensate with more frequent fault map characterizations.

Note that the fault inclusion property is still valid under aging conditions that affect threshold voltage (such as BTI or hot carrier injection). This is because static noise margins are strongly dependent on threshold voltages and decrease monotonically with VDD [Wang and Calhoun 2011].

If testing time is an issue, the impact could be partly mitigated by proceeding concurrently with other system startup procedures, for example, DRAM fault testing and initialization of peripheral devices. If the system is rarely or never power cycled, then BIST can be run periodically during runtime using a temporary way-disabling approach resembling that in Alameldeen et al. [2011b]. Owing to the relatively low frequency and duration of such testing, the impact of BIST on overall system performance is expected to be small.

If on-chip or off-chip non-volatile memory is available, the low-overhead fault maps could be characterized at test time or first power-up and stored permanently (similar to online self-test methods from Li et al. [2008]). This has the advantage of eliminating runtime BIST and its associated performance overheads, but forgoes any compensation for aging aside from design guardbanding.

Idle low-power states are another consideration for the fault map design. In the event of a complete core shutdown without state retention, the fault maps must be (1) maintained in the tag arrays, costing idle power; (2) written to non-volatile storage; or (3) sacrificed, in which case BIST must be run again when the core is brought online. In the first case, tag array static power can limit the effectiveness of long-term core shutdowns. In the latter two cases, there is a significant energy and performance cost to shut down a core for a short duration. We leave the coordination of DPCS and CPU/memory power states to future work.

5.2. Static Policy: SPCS

In the static policy, the lowest VDD level is used on a per-chip basis that has at least 99% effective capacity (also subject to the yield constraint described in Section 5.1.6).

This is set once for the entire system runtime. The only performance overhead to this policy is due to any additional misses that may be caused by the few faulty blocks.

The primary benefit of using SPCS is that voltage guardbands could easily be reduced to suit the unique manufactured outcome of each cache, while only minor modifications to the cache controller and/or software layers are needed. Power can be reduced with negligible performance impact. Because the SRAM cell and block failure rates rise exponentially as the supply voltage is lowered, additional voltage reduction beyond the 99% capacity point brings more power savings, but potentially a significant loss in performance. This phenomenon is described in more detail during the analytical evaluation in Section 7, and is one reason why more sophisticated fault-tolerance methods get diminishing improvements from voltage scaling.

5.3. Dynamic Policies: DPCS

A basic guiding principle behind cache operation is that programs tend to access data with spatial and temporal locality. If a program accesses a large working set of data in a short period of time, then large caches are likely to improve performance. Conversely, if a smaller working set is accessed during an equal length of time, then the cache capacity is less important. Note that in both cases, it is important that data is reused for the cache to be effective.

For these reasons, the SPCS policy described in Section 5.2 can be too conservative. This is because the full cache capacity may be overkill for a given application in a particular phase of execution. The primary motivation to consider a DPCS policy over SPCS is to exploit these situations when the whole cache is unnecessary to deliver acceptable performance.

The proposed DPCS policies described next in Sections 5.3.1 and 5.3.2 are only two possibilities among many. In both cases, we emphasize simple policies that can be easily implemented.

5.3.1. DPCS Policy 1: Access Diversity Based. This DPCS policy (described by Algorithm 2) tries to balance energy efficiency with performance by using spatial locality as the primary guiding principle. At the end of every time interval, if the fraction of *available cache capacity that was touched* falls below some fixed *LowThreshold* (*LT*), the DPCS policy reduces voltage and sacrifices cache capacity for the next interval. Conversely, if the cache pressure is high, then DPCS boosts voltage for the next interval and reenables previously faulty blocks, which increases the capacity once again.

ALGORITHM 2: DPCS Policy 1 (Access Diversity Based)

Input: *CyclesPerInterval* (integer, in cycles), *HighThreshold* (*HT*: high policy threshold, arbitrary units), *LowThreshold* (*LT*: low policy threshold, arbitrary units)
if *ClockCycleNum* mod *CyclesPerInterval* == 0 **then**
  Compute *IntervalBlockTouchedRate* as fraction of blocks that were touched at least once during previous interval;
  Compute *IntervalCapacityRate* as fraction of full cache capacity that was available during previous interval;
  if *IntervalBlockTouchedRate* $\geq HT *$ *IntervalCapacityRate* **then**
    DPCSTransition(min(*CurrVDD* + 1, *SPCS_VDD*));
  else
    if *IntervalBlockTouchedRate* $\leq LT *$ *IntervalCapacityRate* **then**
      DPCSTransition(max(*CurrVDD* − 1, 1));
    end
  end
end

This policy benefits workloads that are relatively localized over discrete time intervals (small working sets), or those workloads that access memory infrequently. In both cases, the cache capacity sacrifice can yield significant energy savings with little visible performance impact at the system level.

However, the downside of this policy is that it is oblivious to the actual impact of faulty blocks. For example, a piece of code may frequently access a region of memory that maps to only a few cache sets, which may happen to contain some faulty blocks. As a result, miss rates and average access time will dramatically increase, hampering performance. The policy is incapable of reacting to this scenario, as it might only see 5% of the available cache capacity being used for this piece of code. It will not transition to a higher VDD, even though the decreased capacity on a few select sets matters a great deal.

5.3.2. Dynamic Policy 2: Access Time Based. In contrast to *DPCS Policy 1*, this policy (described by Algorithm 3 tries to trade off energy efficiency and performance using relative average access time as the main metric. The policy is similar to that of Section 5.3.1, but it uses different performance counters to guide DPCS transitions.

ALGORITHM 3: DPCS Policy 2 (Access Time Based)

Input: $CyclesPerInterval$ (integer, in cycles), $HitLatency$ (integer, in cycles) $HighThreshold$ (HT: high policy threshold, arbitrary units), $LowThreshold$ (LT: low policy threshold, arbitrary units)

```
if ClockCycleNum mod CyclesPerInterval == 0 then
    Compute AverageAccessTime during previous interval;
    if AverageAccessTime ≥ HT * HitLatency then
        DPCSTransition(min(CurrVDD + 1, SPCS_VDD));
    else
        if AverageAccessTime ≤ LT * HitLatency then
            DPCSTransition(max(CurrVDD − 1, 1));
        end
    end
end
```

The benefit of this policy is that it focuses on the "actual" cache performance as measured by average access time. Thus, the real impact of faulty blocks is captured. Revisiting the example from Section 5.3.1, this approach will scale up VDD to relieve the performance bottleneck. Furthermore, because this policy does its best to bound access time, it may be easier to estimate the impact of DPCS on system-level application performance.

However, this policy also has a significant drawback. Cache miss rates are strongly dependent on application behavior. Miss rates and average access time might increase simply due to the application referencing a previously unused data structure, causing compulsive misses. In such a case, the access time-based policy will increase the supply voltage in order to reduce the miss rate, but it would have little to no effect on performance. Thus, neither policy is superior in all cases.

6. MODELING AND EVALUATION METHODOLOGY

We assessed SPCS and DPCS using a combination of analytical and simulation-based evaluations. The overall framework is depicted in Figure 3. We now describe the models and procedures used in the evaluation.

6.1. Modeling Cache Fault Behavior

Probabilistic failure models were used to analytically compare power/capacity tradeoffs, predict yield, and guide the generation of random fault map instances for the simulation part of our evaluations. Equation (4) summarizes the essential fault models,

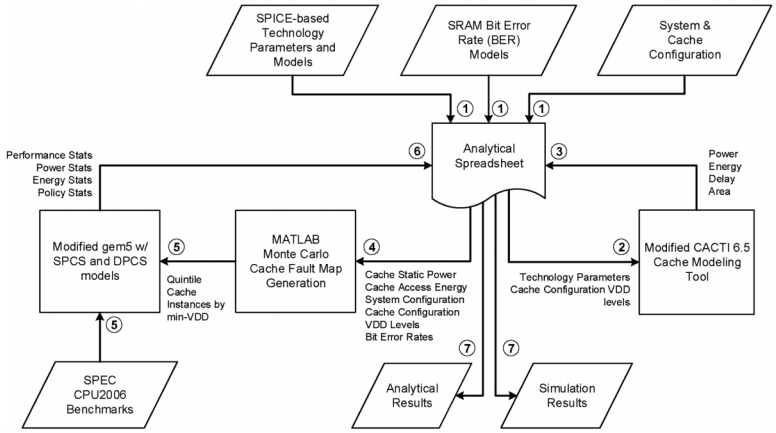


Fig. 3. Modeling and evaluation framework. Circled numbers indicate the order of tasks performed.

Table I. System and Cache Configurations. Some Parameters only Pertain to the Architectural Simulations

Parameter	Config. A	Config. B
Clock Freq.	2GHz	3GHz
L1$ Size, Assoc., Hit Lat.	64KB by 4, two cycles	256KB by 8, four cycles
L2$ Size, Assoc., Hit Lat.	2MB by 8, eight cycles	8 MB by 16, 16 cycles
L1 Interval (cycles)	61,200	122,400
L2 Interval (cycles)	859,200	1,838,400
L1 VoltageTransitionPenalty (cycles)	2 * No. Sets + 100	2 * No. Sets + 200
L2 VoltageTransitionPenalty (cycles)	2 * No. Sets + 400	2 * No. Sets + 2000
L1 DPCS Policy 1 Threshold Low, High (A.U.)	0.75, 0.85	0.75, 0.85
L2 DPCS Policy 1 Threshold Low, High (A.U.)	0.05, 0.15	0.05, 0.15
L1 DPCS Policy 2 Threshold Low, High (A.U.)	1.10, 1.30	1.10, 1.30
L2 DPCS Policy 2 Threshold Low, High (A.U.)	1.10, 1.30	1.10, 1.30

where $p_f(v)$ is the probability of cell failure (Bit Error Rate (BER)) at supply voltage v.

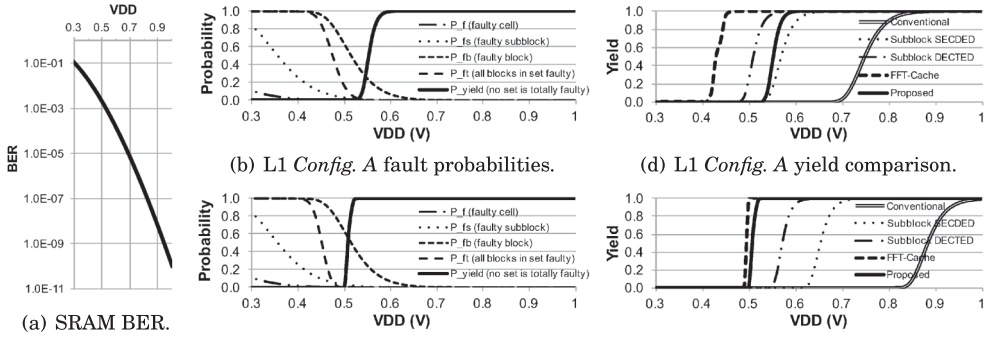
$$\begin{aligned} &p_{fs}(v) = 1 - (1 - p_f(v))^k \text{ is } Pr[\text{faulty subblock of length } k \text{ bits}].\\ &p_{fb}(v) = 1 - (1 - p_{fs}(v))^m \text{ is } Pr[\text{faulty block of length } m \text{ subblocks}].\\ &p_{ft}(v) = p_{fb}(v)^a \text{ is } Pr[\text{completely faulty set of associativity } a].\\ &p_{yield}(v) = (1 - p_{ft}(v))^s \text{ is the yield of an } a\text{-way DPCS cache with } s \text{ sets}. \end{aligned} \tag{4}$$

For our yield comparisons with SECDED and DECTED ECC, we assumed each method maintained ECC at the subblock level, which is significantly stronger than conventional block-level application. The number of ECC bits $n - k$ required per subblock of length k bits with total subblock codeword of length n bits, as well as ECC yield, is given by Equation (5):

$$\begin{aligned} &n_{SECDED} - k = log_2(n_{SECDED}) + 1,\\ &n_{DECTED} - k = log_2(n_{DECTED}) + 2,\\ &p_{yield,SECDED} = Pr[\text{All subblocks have } \leq 1 \text{ faulty bit}],\\ &p_{yield,DECTED} = Pr[\text{All subblocks have } \leq 2 \text{ faulty bits}]. \end{aligned} \tag{5}$$

6.2. System and Cache Configurations

The proposed mechanism as well as the SPCS and DPCS policies were evaluated for two system configurations: *Config. A* and *Config. B*. These are described by Table I. Each system configuration had a split L1 cache and an L2 cache. SPCS and DPCS



(a) SRAM BER.

(b) L1 *Config. A* fault probabilities.

(c) L2 *Config. B* fault probabilities.

(d) L1 *Config. A* yield comparison.

(e) L2 *Config. B* yield comparison.

Fig. 4. Analytical failure characteristics for each configuration.

policies were only applied to the L1D and L2 caches. The same policy was used at both levels. The policies' decisions were not coordinated between cache levels.

6.3. Technology Parameters and Modeling Cache Architecture

To maintain relevance to the test chip experiments from Section 3, MOSFET saturation and leakage current were drawn from the corresponding industrial 45nm SOI technology library. All other parameters are based on International Technology Roadmap for Semiconductors (ITRS) data. BERs were computed using models from Wang and Calhoun [2011] and are shown in Figure 4(a). Since the proposed architectural mechanism allows for SRAM access at reduced voltage, we adopted read stability for the BER calculations, which was the worst case for Static Noise Margin (SNM). Otherwise, we did not distinguish between causes of cell failure.

We calculated delay, static power, dynamic energy per access, and area of the modified and baseline cache architectures using CACTI 6.5 [Balasubramonian et al. 2009]. CACTI generated the optimal design for the nominal VDD of 1V using the energy-delay product as the metric. We modified CACTI to reevaluate the energy and delay as the data array VDD was scaled down without reoptimizing the design. The L2 cache used slower and less leaky transistors than the L1 cache to keep power in check.

Our CACTI results indicated that as long as the data array VDD is above near-threshold levels, reducing the data cell VDD impacts the total cache access time by approximately 10% in the worst case of four cache configurations at min-VDD. This performance impact is modeled as a one-cycle hit time penalty when the cache operates at the lowest DPCS voltage level in our simulations. This is based on a conservative assumption that the cache access critical path dictates overall cycle time.

6.4. Simulation Approach

To simulate the *baseline* (no VDD scaling and no faulty blocks), *SPCS*, *DPCS Policy 1*, and *DPCS Policy 2* caches, we carefully modified the cache architecture in the gem5 [Binkert et al. 2011] framework to implement SPCS and DPCS in detail as well as instrument it for cache, CPU, and DRAM power and energy. These used simple first-order energy models, with constants for static power and dynamic energy/operation. This enabled the overall energy impact of extra CPU stall cycles and DRAM accesses to be captured in addition to the direct cache energy savings from SPCS and DPCS.

We used 16 SPEC CPU2006 benchmarks cross-compiled for the Alpha ISA using base-level optimization using gcc. The benchmarks were fast-forwarded for two billion instructions, then simulated in maximum detail for two billion instructions using the first SPEC reference data inputs. We modeled a single-core system to run the

Table II. Common Parameters Used in All Gem5 Simulations

Parameter	Value	Parameter	Value
ISA	Alpha	Simulation Mode	Syscall Emulation
CPU Model	Detailed (OoO)	Block Replacement Policy	LRU
Number of Cores	1	Cache Configuration	L1 (Split), L2
Number of Memory Channels	1	Cache Block Size	64B
Memory Model	DDR3-1600 × 64	Physical Memory Size	2048MB
Fast-forward	2 billion instructions	Simulate	2 billion instructions
Benchmark Compile Optimizations	Base	Input	First reference
Full VDD (Baseline)	1V	Number of Data VDDs, Added Bits/Block	3, 3

single-threaded benchmarks. The gem5 settings common to all system configurations are summarized in Table II.

For each of the nonbaseline policies, 16 SPEC benchmarks, and two system configurations, gem5 was run five times for five unique fault map inputs. Thus, in total, we performed 512 gem5 simulations to provide some insight on the impact of manifested process variations.

6.4.1. Fault Map Generation and Selection of Runtime VDDs. For accurate simulation in the presence of faulty blocks at low voltage, probabilistic power/capacity scaling data based on CACTI results were input to a MATLAB model that generated 10,000 random fault map instances for each cache configuration. A script chose five fault maps corresponding to each quintile in the distribution as sorted by the min-VDD of each cache instance. For each of these five selected fault map instances, three runtime VDDs were selected based on the criteria described earlier in Section 5.1.6. The total static power of each cache instance was then computed based on the number of power-gated faulty blocks. The particular locations of faulty blocks were assumed to not matter, because eventually all blocks will be referenced roughly uniformly [Sánchez et al. 2013].

The SPCS voltage for each fault map was minimized such that the capacity was at least 99%. The lower DPCS voltage was set such that capacity was at least 75% and no set was totally faulty (recall that we require each set to have at least one nonfaulty block at all voltages). In practice, out of 40,000 total generated fault map instances, the lower DPCS voltage was always limited by the latter constraint.

6.4.2. Parameter Selection for DPCS Policies. The DPCS policy parameters were set to reasonable values to reduce the huge design space, based on our analysis of policy behaviors on different workloads (examples are given later in Section 8.2.3). We assumed that the *VoltageRegulatorDelay* to scale the data array VDD is enough to minimize noise-induced SRAM errors and allow for a stable transition. For each cache configuration, the *Interval* parameter was set to be roughly 100× the respective *VoltageTransitionPenalty* in clock cycles. This is done so that the worst-case system-level performance impact of DPCS transitions would be bounded to 1% of overall runtime. Our choices of *Interval* are not guaranteed to be optimal across all applications, but they do provide some guarantees on the high-level performance impact of DPCS.

DPCS *HighThreshold* (*HT*) and *LowThreshold* (*LT*) for both policies were set to provide sufficient hysteresis, avoiding voltage-level oscillation for workloads in general over different phases of execution. For DPCS Policy 1 (access diversity based), the thresholds were set such that mild cache utilization (as measured by the working set size over an *Interval* [Dhodapkar and Smith 2002]) would result in a voltage reduction. For DPCS Policy 2 (access time based), we ensured that the cache average access time would not suffer too greatly in relation to its hit time.

The choice of policy parameters might be chosen in other ways, for example, by leveraging other techniques for detecting phases of execution [Dhodapkar and Smith 2003; Dani et al. 2014]. The DPCS policy parameters could also be dynamically adjusted by online monitoring of workload behaviors and/or OS-based management. We leave these possibilities to future work, where we aim to adapt DPCS to the context of multicore systems.

7. ANALYTICAL RESULTS

In this section, results pertaining to yield, static power versus effective capacity, and area overheads are discussed without the use of architectural simulation.

7.1. Fault Probabilities and Yield

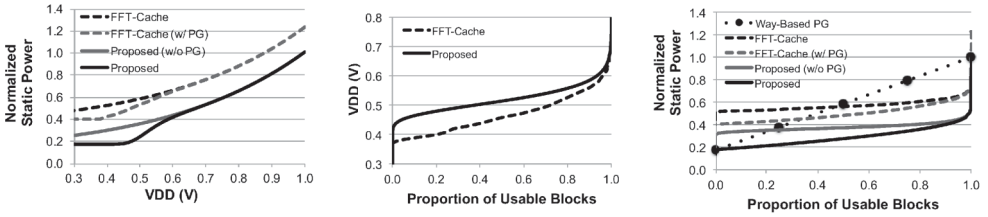
Using the derived static power from our technology data integrated into CACTI as well as the analytical fault models from Section 6.1, we were able to compare our proposed mechanism analytically with alternative approaches. The results for L1 *Config. A* and L2 *Config. B* are shown in Figure 4. Similar trends are observed for the L1 *Config. B* and L2 *Config. A* caches, whose results are not shown for brevity. For yield versus design-time fixed VDD, we compare with SECDED/DECTED ECC and FFT-Cache [BanaiyanMofrad et al. 2011], which is a recent FTVS approach achieving one of the lowest min-VDDs. For FFT-Cache and both ECC schemes, fault tolerance is applied at the subblock level of eight bytes (where the full block size is 64 bytes) for additional resilience. For the static power versus effective capacity metric described in Section 4, we compare with FFT-Cache and a generic way-granularity (associativity reduction) power gating scheme.

The probability of faulty bits (BER) is shown in Figure 4(a), while the probabilities of faulty subblocks, blocks, sets, and DPCS yield are depicted in Figures 4(b) and 4(c). It is clear that in the region of 500–600mV, a steep drop-off in yield is encountered. This is due to exponential fault rates at the bit level, which becomes critical in this region. This trend agrees with our test chip measurements in Section 3 that showed that SRAM failures experience “avalanche-like” behavior.

In Figures 4(d) and 4(e), the yield of DPCS is compared with alternative approaches. Although our approach does not achieve the best min-VDD for constant yield, it did better than SECDED in all cache configurations, even though ECC is applied at eight-byte subblock granularity. In the L1 *Config. A* configuration, DECTED achieves moderately better min-VDD than our mechanism due to low associativity, which impacts yield of our approach. The 16-way set associativity of L2 *Config. B* results in lower min-VDD than both ECC schemes and nearly matches FFT-Cache. Note that DECTED typically incurs high area, power, and performance overheads to achieve the yields shown [Kim et al. 2007; Rossi et al. 2011]. Furthermore, SECDED/DECTED may be overkill for sparse voltage-induced faults, and as voltage is reduced, tolerating bit cell failures reduces the ability of these ECC schemes to tolerate soft errors. Nevertheless, these ECC schemes could be combined with our approach to handle both voltage-induced faults as well as transient soft errors. We leave this to future work.

7.2. Static Power versus Effective Capacity

Despite some yield weaknesses for low-associativity cache configurations, the proposed mechanism still achieves lower total static power at all voltages compared to FFT-Cache [BanaiyanMofrad et al. 2011]. This is due to the lower overheads of DPCS’ fault tolerance compared to the complex approach of FFT-Cache. The difference arises from a significantly smaller fault map with only three extra bits per 64-byte block. In contrast, FFT-Cache needs two entire fault maps for each of the lower VDDs, compounding its existing high overheads and extra necessary logic on the critical path.



(a) Static power vs. VDD. (b) VDD vs. effective capacity. (c) Static power vs. effective capacity comparison.

Fig. 5. Analytical power/capacity results for the L1 *Config. A* cache with and without power gating of faulty blocks. Similar normalized trends were observed for the other three cache configurations.

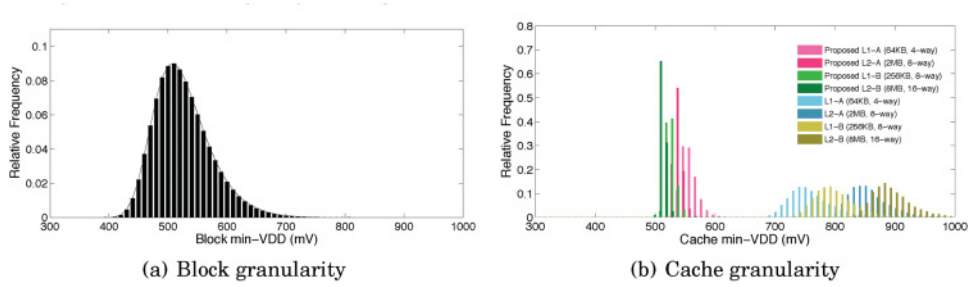
Furthermore, under certain floor plan and layout conditions described earlier in Section 5.1.2, our approach can power gate blocks as they become faulty at low voltage, providing additional power reduction at low capacities compared to pure voltage scaling. This result is depicted in Figure 5(a) along with a hypothetical version of FFT-Cache with power gating of lost capacity (gray dashed line). If the power gating circuitry described in Section 5.1.2 is omitted, then power savings will be moderately worse at capacities less than 100%, as shown by Figure 5(a). However, overall area overheads would decrease by roughly 2%, as described next in Section 7.3. This simpler architecture may be better for SPCS, where power gating logic is only used for up to 1% of blocks that are allowed to be faulty. While the per-block power gating feature is useful for an improved power versus capacity curve, it is not absolutely essential to DPCS. This makes our architecture compelling even for traditional thin-cell SRAM layouts with almost no modifications to the macro floor plan.

Although our approach achieves lower power than FFT-Cache, it also has a lower effective capacity at all voltage levels than FFT-Cache due to a much simpler but weaker fault-tolerance mechanism, as shown by Figure 5(b). This reaffirms the better yield of FFT-Cache. These results clearly illustrate a trade-off in power and capacity at each voltage level between the two approaches.

In the end, the superior power savings of the proposed approach make up for the capacity deficit with respect to FFT-Cache, as depicted by Figure 5(c). Moreover, our approach with power gating of faulty blocks does better than generic associativity reduction via per-way power gating at all cache capacities. This is unlike FFT-Cache, which does worse than power gating below 50% cache capacity because of its high overheads. We found that for the L1 *Config. A* cache, our mechanism achieves 31.1% lower static power than FFT-Cache at the same 99% effective capacity. In addition, as cache capacity is scaled down with power, the gap increases in favor of our approach.

7.3. Area Overhead

Our CACTI results indicate that from the fault map alone, our area overheads compared to a baseline cache lacking fault tolerance do not exceed 4% in the worst case of all configurations. The additional area overheads from the power gating transistor [Powell et al. 2000] plus small inverter are estimated to be no more than 2%. The DPCS policy implementation is assumed to have negligible area overhead, due to its simplicity and that it could be implemented in software, as the cache controller typically includes the necessary performance counters. Furthermore, the fault map comparison logic is only a few gates per cache way. Thus, we estimated the total area overhead to be up to 6% among all tested cache configurations. These area overheads are a significant improvement compared to the reported overheads of other FTVS schemes such as 10T



(a) Block granularity (b) Cache granularity

Fig. 6. Manifested distribution of min-VDD at the block and cache granularities.

SRAM cells (66%), ZerehCache (16%), Wilkerson et al. (15%), Ansari et al. (14%), and FFT-Cache (13%) [BanaiyanMofrad et al. 2011].

8. SIMULATION RESULTS

In this section, we discuss aggregate energy savings and performance overheads across a portion of the SPEC CPU2006 benchmark suite on both system configurations for several different fault map instances.

8.1. Fault Map Distributions

Using the fault models described in Section 6.1 and the methodology from Section 6.4.1, we generated 10,000 complete fault map instances for each of the four cache configurations, that is, *L1-A*, *L2-A*, *L1-B*, and *L2-B*. Each fault map contained the min-VDD for each cache block at a resolution of 10mV. Figure 6(a) depicts the aggregate faulty block distribution across all 40,000 randomly generated fault maps.

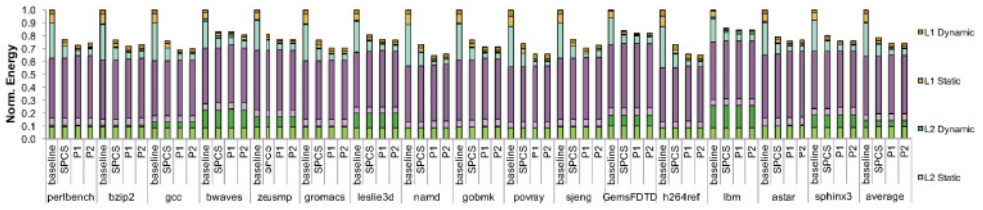
Figure 6(b) depicts the histogram of the minimum global VDD that can be used across a cache instance. For the baseline case, any faulty block at a given voltage will limit the overall min-VDD. Clearly, the long tail from Figure 6(a) hampers the traditional design approach (these numbers agree with our 45nm process guidelines for min-VDD as well). In the baseline caches, as nominal cache capacity increases, so does the min-VDD, regardless of associativity.

However, the cache min-VDD distributions for the proposed architecture follow a very different trend, as indicated by Figure 6(b). Unlike traditional caches, the benefits of increased associativity outweigh increased nominal capacity for reducing VDD. This is because the voltage scalability of our approach is only limited by the constraint that no set may be totally faulty (recall the min-VDD requirement for our approach from Sections 5.1.6 and 6.1). As associativity increases, the likelihood of any set being completely faulty rapidly decreases.

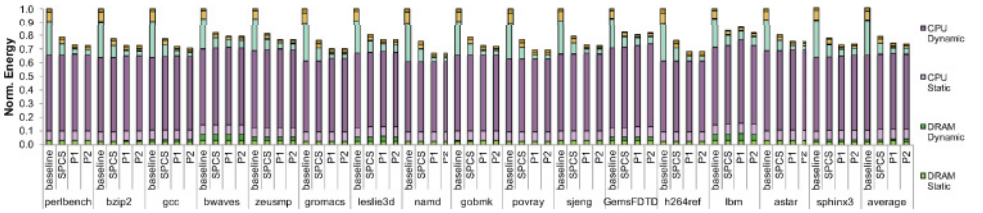
Thus, our simple fault-tolerance scheme exhibits good scalability as cache memories increase in size and associativity. Moreover, the variance in min-VDD across many cache instances is significantly less than conventionally designed caches, meaning that even design-time choice of VDD can be done with narrower guardbands while maintaining high yield.

8.2. Architectural Simulation

We now discuss the impact of SPCS and DPCS on energy and performance. All the depicted results include the architectural design choice to power gate faulty blocks. We also conducted the same set of experiments without per-block power gating; the results are not shown for brevity. For the worst-case fault maps at the lowest voltages, the total cache static energy difference between the two approaches did not exceed 5%. At the



(a) *Config. A*. CPU power was normalized to 50% and DRAM energy to 15% of total average baseline energy.



(b) *Config. B*. CPU dynamic power was scaled up by $3\times$, and static power by $5\times$ compared to *Config. A* in accordance with 50% higher clock frequency. DRAM energy was unchanged.

Fig. 7. Breakdown of averaged total system energy for *baseline*, *SPCS*, and DPCS policies *P1* and *P2*, all normalized to the respective benchmarks' *baseline* system total energy. Note that because DPCS has little impact on overall runtime and DRAM accesses, the relative energy breakdown across CPU and DRAM can be renormalized to other systems in a straightforward manner.

system level, the overall energy gap between the two design choices was less than 1%. Henceforth, we only discuss the architecture with the power gating feature included.

8.2.1. Breakdown of Energy Savings. Both SPCS and DPCS deliver static and dynamic power savings due to low-voltage operation even during cache accesses. The breakdown of total system energy is depicted in Figure 7 for the *baseline*, *SPCS*, *DPCS Policy 1* (*P1*), and *DPCS Policy 2* (*P2*) caches using the mean of five unique fault map runs. The variations in energy breakdowns across benchmarks in total *baseline* and *SPCS* energy are due to workload dependencies, that is, the extent to which they are CPU or memory bound.

SPCS achieved good and consistent energy savings across all benchmarks, achieving an average of 62% total cache and 22% total system energy savings with respect to *baseline* in both *Configs. A* and *B*. This is because cache efficacy is not significantly impacted at the 99% capacity point, and no voltage transitions occur to other power/capacity points.

Energy savings were almost always better for both DPCS policies *P1* and *P2* compared to *SPCS* and the *baseline*. This was in line with our expectations, as both policies never go above the *SPCS* voltage level because little performance would be gained. However, note that DPCS has some system-level energy overhead due to extra time spent in DPCS transitions and potentially more cache misses and CPU stall cycles compared to *SPCS* or *baseline*. For example, reduced power and capacity in L1 due to DPCS can cause more L2 accesses, resulting in a trade-off between L1 static power and L2 dynamic energy.

In nearly all cases, the access diversity-based *DPCS Policy 1* (*P1* in Figure 7) achieved equal or slightly lower total cache energy than the access time-based *DPCS Policy 2* (*P2*). On average, *P1* reduced total cache (system) energy by 78% (27%) compared to *baseline* on *Config. A*, and by 80% (26%), on *Config. B*. On average across all benchmarks and configurations, *P2* saved nearly identical system-level energy as *P1*. However, in

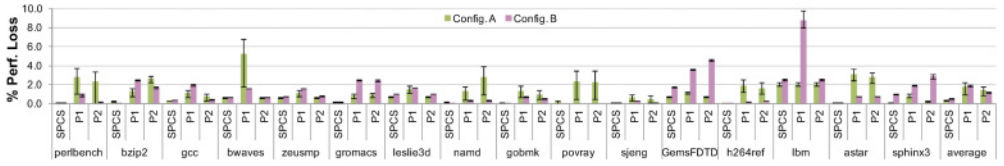


Fig. 8. Average performance degradation of *SPCS* and DPCS policies *P1* and *P2*. Error bars indicate the range of performance impact across the five quintile fault maps.

some benchmarks, such as *perlbench* on *Config. A* or *lbm* on *Config. B*, the gap between policies was noticeable.

Discrepancies in energy savings between *DPCS Policy 1* and *DPCS Policy 2* could be explained by a combination of threshold parameters and the method of inferring performance impact at low voltage. For example, the cache capacity utilization measured by *DPCS Policy 1* loosely relates to performance through working set analysis. In contrast, *DPCS Policy 2* measures performance more directly through average access time. However, owing to the huge design space already presented, we do not have the resources to exhaustively test other policy threshold parameters. We leave static and dynamic policy optimization to future work.

Regardless of differences in policies, SPCS and DPCS demonstrate reduced total cache and system energy overall with little impact on the CPU or DRAM. As we will see in the next section, this can be attributed to low performance overheads while still reducing static cache power and some dynamic access energy.

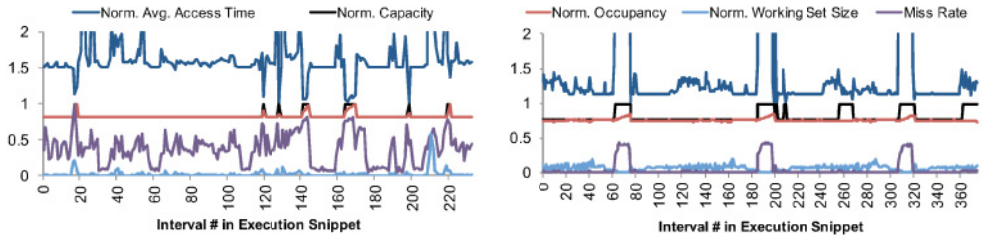
8.2.2. Performance Overheads. The energy savings presented previously come at a mild performance cost as measured by execution time. Figure 8 depicts the average performance penalty for *SPCS*, *DPCS Policy 1*, and *DPCS Policy 2* for each benchmark with respect to the *baseline* system.

Across all benchmarks on *Config. A*, the slowdown of *SPCS* is only $0.32 \pm 0.06\%$ with respect to *baseline*. For *Config. B*, the slowdown is $0.50 \pm 0.02\%$. This shows that a 1% loss of cache capacity has a negligible impact on performance. Thus, even with the worst-case fault maps, the per-chip *SPCS* approach is effective at achieving energy savings with very little overhead. Thanks to the low variance in min-VDD with SPCS (see Section 8.1), an aggressive design-time choice of SRAM VDD (as opposed to our runtime approach) could also result in negligible performance impact while maintaining high yield and good energy savings.

Figure 8 also shows how DPCS performance degrades compared to SPCS. On average, the access time-based *DPCS Policy 2* (0.77%–1.76%) does equal or is better than the access diversity-based *DPCS Policy 1* (0.94%–2.24%) on *Config. A*. This small performance advantage of *P2* holds for *Config. B*. For both SPCS and DPCS, higher cache associativity improves performance consistency in addition to min-VDD across a variety of fault maps.

Interestingly, the per-benchmark performance of both policies appears to be dependent on the system configuration. For example, in *Config. A*, *P1* does poorly on *bwaves* and is also very sensitive to fault map variation. In contrast, *lbm* performed very consistently for *SPCS*, *P1*, and *P2* under *Config. A*, but suffered relatively more under *P1* for *Config. B*. We believe these sensitivities are due to a combination of different CPU frequencies, cache hit times, and possibly fault map instances.

8.2.3. DPCS Policy Behavior. Finally, we illustrate some characteristics of each DPCS policy by examining short snippets of their behavior during execution. Figure 9 shows two trace fragments for different workloads, system configurations, cache levels, and DPCS policies.



(a) *gobmk*, *DPCS Policy 1*, L1 *Config. A*. Each *Interval* corresponds to 61,200 cycles at 2GHz.

(b) *bzip2*, *DPCS Policy 2*, L2 *Config. B*. Each *Interval* corresponds to 1,838,400 cycles at 3GHz.

Fig. 9. Selected snippets of DPCS traces in mid-execution using median fault map instances.

For most of the execution trace depicted in Figure 9(a), the cache capacity is roughly 80%. This indicates that DPCS is in the lowest voltage mode. *DPCS Policy 1* does not boost voltage unless the working set size exceeds 90% of the available capacity. This occurs several times around *Interval* 150. The cache occupancy then increases rapidly, indicating that some performance was probably recovered. However, higher miss rates and average access times do not necessarily cause high block touch rates. This can be seen around *Interval* 210, where the miss rate jumps to approximately 60%, but only 65% of the nominal capacity is referenced at that time. In this case, a DPCS boost would probably not help performance noticeably.

A very different trend is shown in Figure 9(b). Since the cache is relatively large compared to the needs of the application, the working set size always remains below 20% of nominal capacity. However, note that the application regularly has a miss rate of up to 40%. *DPCS Policy 2* adapts by transitioning to a higher cache capacity in an attempt to recover performance. However, the downside is that such an action is not guaranteed to affect the miss rate or recover performance.

9. CONCLUSION AND FUTURE WORK

In this work, we proposed SPCS and DPCS, a novel FTVS SRAM architecture for energy-efficient operation. Our proposed mechanism and policies leverage several important observations. First, using our 45nm SOI test chips, we observed the *fault inclusion property*, which states that any block that fails at some supply voltage will also be faulty at all lower voltages. To the best of our knowledge, no other FTVS approaches exploit this behavior. This allows for efficient fault map representation for multiple runtime cache VDD levels. Our approach also has the benefit of a simple and low-overhead implementation, which allows the caches to achieve better power/capacity trade-offs than some competing approaches that primarily focus on achieving low min-VDD at fixed yield.

We believe that DPCS has the potential to complement conventional DVFS for improved system-level energy proportionality. The application of DPCS in a real system would bring interesting challenges and opportunities to the software stack. Our future work seeks to develop coordinated DVFS and DPCS policies at the system level, taking into account heterogeneous system architectures, multicore processors, cache coherence, aging effects, and implications for power management of other system components. This approach to variation-aware design could also be extended to allow for approximate computing.

REFERENCES

Jaume Abella, Javier Carretero, Pedro Chaparro, Xavier Vera, and Antonio González. 2009. Low Vccmin fault-tolerant cache with highly predictable performance. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*.

Amit Agarwal, Bipul C. Paul, Hamid Mahmoodi, Animesh Datta, and Kaushik Roy. 2005. A process-tolerant cache architecture for improved yield in nanoscale technologies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 13, 1 (2005), 27–38.

Yuvraj Agarwal, Alex Bishop, Tuck-Boon Chan, Matt Fotjik, Puneet Gupta, Andrew B. Kahng, Liangzhen Lai, Paul Martin, Mani Srivastava, Dennis Sylvester, Lucas Wanner, and Bing Zhang. 2014. *RedCooper: Hardware Sensor Enabled Variability Software Testbed for Lifetime Energy Constrained Application*. Technical Report. University of California, Los Angeles. DOI:http://www.escholarship.org/uc/item/1c21g217.

Alaa R. Alameldeen, Zeshan Chishti, Chris Wilkerson, Wei Wu, and Shih-Lien Lu. 2011a. Adaptive cache design to enable reliable low-voltage operation. *IEEE Transactions on Computers (TC)* 60, 1 (2011), 50–63.

Alaa R. Alameldeen, Ilya Wagner, Zeshan Chishti, Wei Wu, Chris Wilkerson, and Shih-Lien Lu. 2011b. Energy-efficient cache design using variable-strength error-correcting codes. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*. 461–471.

Gene M. Amdahl. 1967. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the Spring Joint Computer Conference*. 483.

Amin Ansari, Shuguang Feng, Shantanu Gupta, and Scott Mahlke. 2011. Archipelago: A polymorphic cache design for enabling robust near-threshold operation. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*. 539–550.

Amin Ansari, Shantanu Gupta, Shuguang Feng, and Scott Mahlke. 2009. ZerehCache: Armoring cache architectures in high defect density technologies. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*. 100–110.

Naveen Balasubramonian, Rajeev Muralimanohar, and Norman P. Jouppi. 2009. *CACTI 6.0: A Tool to Model Large Caches*. Technical Report. HP Laboratories.

Abbas BanaiyanMofrad, Houman Homayoun, and Nikil Dutt. 2011. FFT-Cache: A flexible fault-tolerant cache architecture for ultra low voltage operation. In *Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*. 95–104.

Alessandro Bardine, Manuel Comparetti, Pierfrancesco Foglia, and Cosimo Antonio Prete. 2014. Evaluation of leakage reduction alternatives for deep submicron dynamic nonuniform cache architecture caches. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)* 22, 1 (2014), 185–190.

Luiz André Barroso and Urs Hölzle. 2007. The case for energy-proportional computing. *IEEE Computer* 40, 12 (2007), 33–37.

Luiz André Barroso and Urs Hölzle. 2009. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Vol. 4. Morgan & Claypool Publishers.

Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The gem5 simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (2011), 1–7.

Bill Bowhill, Blaine Stackhouse, Nevine Nassif, Zibing Yang, Arvind Raghavan, Charles Morganti, Chris Houghton, Dan Krueger, Olivier Franza, Jayen Desai, Jason Crop, Dave Bradley, Chris Bostak, Sal Bhimji, and Matt Becker. 2015. The Xeon Processor E5-2600 v3: A 22nm 18-Core Product Family. In *International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers*.

Benton H. Calhoun and Anantha Chandrakasan. 2006. A 256kb sub-threshold SRAM in 65nm CMOS. In *IEEE International Solid State Circuits Conference (ISSCC) Digest of Technical Papers*. 2592–2601.

Arup Chakraborty, Houman Homayoun, Amin Khajeh, Nikil Dutt, Ahmed Eltawil, and Fadi Kurdahi. 2014. Multicopy cache: A highly energy-efficient cache architecture. *ACM Transactions on Embedded Computing Systems (TECS)* 13, 5s, Article 150 (Nov. 2014).

Leland Chang, David M. Fried, Jack Hergenrother, Jeffrey W. Sleight, Robert H. Dennard, Robert K. Montoye, Lidija Sekaric, Sharee J. McNab, Anna W. Topol, Charlotte D. Adams, Kathryn W. Guarini, and Wilfried Haensch. 2005. Stable SRAM cell design for the 32 nm node and beyond. In *Symposium on VLSI Technology Digest of Technical Papers*. 128–129.

Meng-Fan Chang, Chien-Fu Chen, Ting-Hao Chang, Chi-Chang Shuai, Yen-Yao Wang, and Hiroyuki Yamauchi. 2015. A 28nm 256kb 6T-SRAM with 280mV improvement in Vmin using a dual-split-control assist scheme. In *International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers*.

Hsiang-Yun Cheng, Matt Poremba, Narges Shahidi, Ivan Stalev, Mary Jane Irwin, Mahmut Kandemir, Jack Sampson, and Yuan Xie. 2014. EECache: Exploiting design choices in energy-efficient last-level caches for chip multiprocessors. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*. 303–306.

Aparna Mandke Dani, Bharadwaj Amrutur, and Y. N. Srikant. 2014. Toward a scalable working set size estimation method and its application for chip multiprocessors. *IEEE Transactions on Computers* 63, 6 (June 2014), 1567–1579. DOI:http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6375705

Howard David, Chris Fallin, Eugene Gorbatov, Ulf R. Hanebutte, and Onur Mutlu. 2011. Memory power management via dynamic voltage/frequency scaling. In *Proceedings of the International Conference on Autonomic Computing (ICAC)*. 31.

Qingyuan Deng, David Meisner, Abhishek Bhattacharjee, Thomas F. Wenisch, and Ricardo Bianchini. 2012. CoScale: Coordinating CPU and memory system DVFS in server systems. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*. 143–154.

Qingyuan Deng, David Meisner, Luiz Ramos, Thomas F. Wenisch, and Ricardo Bianchini. 2011. MemScale: Active low-power modes for main memory. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Vol. 46. ACM, 225–238.

Ashutosh S. Dhodapkar and James E. Smith. 2002. Managing multi-configuration hardware via dynamic working set analysis. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*. 233–244.

Ashutosh S. Dhodapkar and James E. Smith. 2003. Comparing program phase detection techniques. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*. 217.

Nikil Dutt, Puneet Gupta, Alex Nicolau, Abbas BanaiyanMofrad, Mark Gottscho, and Majid Shoushtari. 2014. Multi-layer memory resiliency. In *Proceedings of the Design Automation Conference (DAC)*.

Behzad Ebrahimi, Reza Asadpour, Ali Afzali-Kusha, and Massoud Pedram. 2015. A FinFET SRAM cell design with BTI robustness at high supply voltages and high yield at low supply voltages. *International Journal of Circuit Theory and Applications* (Jan. 2015), DOI:10.1002/cta.2057.

Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. 2011. Dark silicon and the end of multicore scaling. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*. ACM, 365–376.

Xiaobo Fan, Carla S. Ellis, and Alvin R. Lebeck. 2005. The synergy between power-aware memory systems and processor voltage scaling. *Lecture Notes in Computer Science* 3164 (2005), 164–179.

Alexandra Ferreron, Dario Suarez-Gracia, Jesus Alastruey, Teresa Monreal, and Victor Vinals. 2014. Block disabling characterization and improvements in CMPs operating at ultra-low voltages. In *Proceedings of the International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*.

Krisztián Flautner, Nam Sung Kim, Steve Martin, David Blaauw, and Trevor Mudge. 2002. Drowsy caches: Simple techniques for reducing leakage power. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*. 148–157.

Hamid Reza Ghasemi, Stark C. Draper, and Nam Sung Kim. 2011. Low-voltage on-chip cache architecture using heterogeneous cell sizes for high-performance processors. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*. 38–49.

Mark Gottscho, Abbas BanaiyanMofrad, Nikil Dutt, Alex Nicolau, and Puneet Gupta. 2014. Power/capacity scaling: Energy savings with simple fault-tolerant caches. In *Proceedings of the Design Automation Conference (DAC)*.

Puneet Gupta, Yuvraj Agarwal, Lara Dolecek, Nikil Dutt, Rajesh K. Gupta, Rakesh Kumar, Subhasish Mitra, Alexandru Nicolau, Tajana Simunic Rosing, Mani B. Srivastava, Steven Swanson, and Dennis Sylvester. 2013. Underdesigned and opportunistic computing in presence of hardware variability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 32, 1 (2013), 8–23.

Said Hamdioui, Ad J. van de Goor, and Mike Rodgers. 2002. March SS: A test for all static simple RAM faults. In *Proceedings of the International Workshop on Memory Technology, Design, and Testing*. 95–100.

Yinhe Han, Ying Wang, Huawei Li, and Xiaowei Li. 2013. Enabling near-threshold voltage (NTV) operation in multi-VDD cache for power reduction. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*. 337–340.

John L. Hennessy and David A. Patterson. 2012. *Computer Architecture: A Quantitative Approach* (5th ed.). Morgan Kaufmann.

Farrukh Hijaz and Omer Khan. 2014. NUCA-L1: A non-uniform access latency level-1 cache architecture for multicores operating at near-threshold voltages. *ACM Transactions on Architecture and Code Optimization (TACO)* 11, 3, Article 29 (Oct. 2014), 1–28.

Mohammed Abid Hussain and Madhu Mutyam. 2008. Block remap with turnoff: A variation-tolerant cache design technique. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*. 783–788.

Jangwoo Kim, Nikos Hardavellas, Ken Mai, Babak Falsafi, and James C. Hoe. 2007. Multi-bit error tolerant caches using two-dimensional error coding. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*. 197–209.

Seokjoong Kim and Matthew R. Guthaus. 2013. SEU-aware low-power memories using a multiple supply voltage array architecture. In *Proceedings of the International Conference on Very Large Scale Integration (VLSI-SoC) (IFIP Advances in Information and Communication Technology)*, Andreas Burg, Aye Cokun, Matthew Guthaus, Srinivas Katkoori, and Ricardo Reis (Eds.), Vol. 418. Springer, Berlin, 181–195.

Cheng-Kok Koh, Weng-Fai Wong, Yiran Chen, and Hai Li. 2009. The salvage cache: A fault-tolerant cache architecture for next-generation memory technologies. In *Proceedings of the International Conference on Computer Design (ICCD)*. 268–274.

Animesh Kumar, Jan Rabaey, and Kannan Ramchandran. 2009. SRAM supply voltage scaling: A reliability perspective. In *Proceedings of the International Symposium on Quality Electronic Design (ISQED)*. 782–787.

Liangzhen Lai and Puneet Gupta. 2014. Accurate and inexpensive performance monitoring for variability-aware systems. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*. 467–473.

Yanjing Li, Samy Makar, and Subhasish Mitra. 2008. CASP: Concurrent autonomous chip self-test using stored test patterns. In *Design, Automation, and Test in Europe (DATE)*. 885–890.

Tayyeb Mahmood, Seokin Hong, and Soontae Kim. 2014. Ensuring cache reliability and energy scaling at near-threshold voltage with macho. *IEEE Transactions on Computers (TC)* 64, 6 (2014), 1694–1706.

Sparsh Mittal. 2014. A survey of architectural techniques for improving cache power efficiency. *Sustainable Computing: Informatics and Systems* 4, 1 (2014), 33–43.

Baker S. Mohammad, Hani Saleh, and Mohammed Ismail. 2014. Design methodologies for yield enhancement and power efficiency in SRAM-based SoCs. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)* PP, 99 (2014), 1.

Serkan Ozdemir, Debjit Sinha, Gokhan Memik, Jonathan Adams, and Hai Zhou. 2006. Yield-aware cache architectures. In *Proceedings of the International Symposium on Microarchitecture (MICRO)*. 15–25.

Michael Powell, Se-Hyun Yang, Babak Falsafi, Kaushik Roy, and T. N. Vijaykumar. 2000. Gated-Vdd: A circuit technique to reduce leakage in deep-submicron cache memories. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*. 90–95.

Jungyul Pyo, Youngmin Shin, Hoi-Jin Lee, Sung-il Bae, Min-su Kim, Kwangil Kim, Ken Shin, Yohan Kwon, Heungchul Oh, Jaeyoung Lim, Dong-wook Lee, Jongho Lee, Inpyo Hong, Kyungkuk Chae, Heon-Hee Lee, Sung-Wook Lee, Seongho Song, Chung-Hee Kim, Jin-Soo Park, Heesoo Kim, Sunghee Yun, Uk-Rae Cho, Jae Cheol Son, and Sungho Park. 2015. 20nm high-k metal-gate heterogeneous 64b quad-core CPUs and hexa-core GPU for high-performance and energy-efficient mobile application processor. In *International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers*.

Moinuddin K. Qureshi and Zeshan Chishti. 2013. Operating SECDED-based caches at ultra-low voltage with FLAIR. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*. 1–11.

Daniele Rossi, Nicola Timoncini, Michael Spica, and Cecilia Metra. 2011. Error correcting code analysis for cache memory high reliability and performance. In *Design, Automation, and Test in Europe (DATE)*. 1–6.

Daniel Sánchez, Yiannakis Sazeides, Juan M. Cebrián, José M. García, and Juan L. Aragón. 2013. Modeling the impact of permanent faults in caches. *ACM Transactions on Architecture and Code Optimization (TACO)* 10, 4, Article 29 (Dec. 2013), 1–23.

Avesta Sasan, Houman Homayoun, Kiarash Amiri, Ahmed Eltawil, and Fadi Kudahi. 2012. History and variation trained cache (HVT-cache): A process variation aware and fine grain voltage scalable cache with active access history monitoring. In *Proceedings of the International Symposium on Quality Electronic Design (ISQED)*. 498–505.

Avesta Sasan, Houman Homayoun, Ahmed Eltawil, and Fadi Kurdahi. 2009. A fault tolerant cache architecture for sub 500mV operation: Resizable data composer cache (RDC-Cache). In *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'09)*. ACM, 251–260.

Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. 2011. DRAM errors in the wild: A large-scale field study. *Communications of the ACM* 54, 2 (2011), 100.

Philip P. Shirvani and Edward J. McCluskey. 1999. PADded cache: A new fault-tolerance technique for cache memories. In *Proceedings of the VLSI Test Symposium*. 440–445.

Mahmut E. Sinangil, Hugh Mair, and Anantha P. Chandrakasan. 2011. A 28nm high-density 6T SRAM with optimized peripheral-assist circuits for operation down to 0.6V. In *International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers*. 260–262.

Jawar Singh, Dhiraj K. Pradhan, Simon Hollis, and Saraju P. Mohanty. 2008. A single ended 6T SRAM cell design for ultra-low-voltage applications. *IEICE Electronics Express* 5, 18 (2008), 750–755.

Vilas Sridharan and Dean Liberty. 2012. *A Field Study of DRAM Errors*. Technical Report. AMD.

Naveen Verma and Anantha P. Chandrakasan. 2008. A 256 kb 65 nm 8T subthreshold SRAM employing sense-amplifier redundancy. *IEEE Journal of Solid-State Circuits (JSSC)* 43, 1 (2008), 141–149.

Jiajing Wang and Benton H. Calhoun. 2011. Minimum supply voltage and yield estimation for large SRAMs under parametric variations. *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)* 19, 11 (2011), 2120–2125.

Po-Hao Wang, Wei-Chung Cheng, Yung-Hui Yu, Tang-Chieh Kao, Chi-Lun Tsai, Pei-Yao Chang, Tay-Jyi Lin, Jinn-Shyan Wang, and Tien-Fu Chen. 2013. Variation-aware and adaptive-latency accesses for reliable low voltage caches. In *Proceedings of the International Conference on Very Large Scale Integration (VLSI-SoC)*. 358–363.

Neil H. E. Weste and David M. Harris. 2011. *CMOS VLSI Design: A Circuits and Systems Perspective* (4th ed.). Addison-Wesley.

Chris Wilkerson, Hongliang Gao, Alaa R. Alameldeen, Zeshan Chishti, Muhammad Khellah, and Shih-Lien Lu. 2008. Trading off cache capacity for reliability to enable low voltage operation. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*. 203–214.

Meilin Zhang, Vladimir M. Stojanovic, and Paul Ampadu. 2012. Reliable ultra-low-voltage cache design for many-core systems. *IEEE Transactions on Circuits and Systems II: Express Briefs* 59, 12 (2012), 858–862.

Received December 2014; revised June 2015; accepted June 2015