Power / Capacity Scaling: Energy Savings With Simple Fault-Tolerant Caches

Mark Gottscho
UCLA Electrical Engineering
mgottscho@ucla.edu

Abbas BanaiyanMofrad
UC Irvine Computer Science
abanaiya@uci.edu

Nikil Dutt
UC Irvine Computer Science
dutt@uci.edu

Alex Nicolau
UC Irvine Computer Science
nicolau@ics.uci.edu

Puneet Gupta
UCLA Electrical Engineering
puneet@ee.ucla.edu

ABSTRACT

Complicated approaches to fault-tolerant voltage-scalable (FTVS) SRAM cache architectures can suffer from high overheads. We propose static (SPCS) and dynamic (DPCS) variants of power/capacity scaling, a simple and low-overhead fault-tolerant cache architecture that utilizes insights gained from our 45nm SOI test chip. Our mechanism combines multi-level voltage scaling with power gating of blocks that become faulty at each voltage level. The SPCS policy sets the runtime cache VDD statically such that almost all of the cache blocks are not faulty. The DPCS policy opportunistically reduces the voltage further to save more power than SPCS while limiting the impact on performance caused by additional faulty blocks. Through an analytical evaluation, we show that our approach can achieve lower static power for all effective cache capacities than a recent complex FTVS work. This is due to significantly lower overheads, despite the failure of our approach to match the min-VDD of the competing work at fixed yield. Through architectural simulations, we find that the average energy saved by SPCS is 55%, while DPCS saves an average of 69% of energy with respect to baseline caches at 1 V. Our approach incurs no more than 4% performance and 5% area penalties in the worst case cache configuration.

1. INTRODUCTION

As the semiconductor industry continues technology scaling, wafer-to-wafer, die-to-die, and intra-die variations are becoming major obstacles. The impact of variability on performance, power consumption, and yield is now a first-order concern [10]. The standard practice to dealing with variation is to guard-band designs, i.e., margin for the worst-case manufacturing outcomes and operating conditions. However, these methods incur significant overheads in power, performance, area, and cost [10].

One way to reduce the overall power consumption of a circuit is to reduce the supply voltage (VDD). However, voltage-scaled SRAMs are susceptible to faulty behavior. Variations in SRAM cell noise margins, mostly due to the impact of random dopant fluctuation on threshold voltages, result in an exponential increase in probability of cell failure as the supply voltage is lowered [23].

DAC '14, June 01 - 05 2014, San Francisco, CA, USA

http://dx.doi.org/10.1145/2593069.2593184

This has motivated research enabling low-voltage cache operation by tolerating hard and/or soft faults. Static power, dominated by subthreshold leakage current, is the primary consumer of power in memories [9] and has an exponential dependence on supply voltage [24]. Since leakage constitutes a major fraction of total system power [14], even a minor reduction in memory supply voltage can have a significant impact on total chip power consumption. Thus, it is critical that we continue to develop fault-tolerant voltage-scalable (FTVS) caches to allow both energy-efficient and reliable operation.

Most FTVS cache architectures use complicated fault tolerance methods to achieve very low min-VDD for a target expected yield. However, with respect to overall power and/or energy savings, these approaches are limited by Amdahl's Law. Large fault maps and flexible data redundancy mechanisms incur considerable power, area, and performance costs. For example, [5] reported area and power overheads of up to 13% and 16%, respectively, for their FTVS scheme in the L1 cache. These overheads ultimately bound the potential benefits of a given scheme. Moreover, min-VDD is not the only factor influencing power consumption; it is important that designers account for all overheads.

Thus, we believe that a different metric, *power vs. effective capacity*, should be used to evaluate and compare FTVS cache architectures. This metric accounts for the supply voltage as well as the effectiveness and overheads of the particular fault tolerance or capacity reduction mechanism (e.g., cache way-based power gating). With this in mind, simple FTVS schemes have the potential to fare similarly or even better than complex ones in terms of power, performance, and/or area, all with less design and verification effort.

Using ARM Cortex M3-based "Red Cooper" test chips manufactured in a 45nm SOI technology [15] as part of the NSF Variability Expedition (variability.org), we ran March SS tests [11] on the SRAM memory to characterize the nature of bit faults as VDD is reduced. We observed that in the presence of process variation, SRAM faults caused by voltage scaling obey the so-called *fault inclusion property*, i.e., bits that fail at some supply voltage level will also fail at all lower voltages. This observation suggested the use of a compressible fault map such that multiple VDDs can be used with little additional overhead compared to a single-VDD fault map.

In this work, we make three major contributions:

1. *With insight from the power vs. effective capacity metric, we propose a simple, low-overhead fault tolerance mechanism that supports multiple VDD levels. Our*

mechanism combines global voltage scaling of the data array SRAM cells with power gating of individual faulty data blocks to reduce cache power.

2. *We propose static (SPCS) and dynamic (DPCS) policy variants of power/capacity scaling, a novel FTVS scheme using our simple proposed mechanism that significantly reduces overall cache energy with minimal performance and area overheads while maintaining high yield.*

3. *We show analytically that our approach is better than a recent work using a sophisticated FTVS cache architecture [5] as well as simple way-based power gating by achieving lower total static power at all cache capacities. This is despite the inability of our scheme to achieve the lowest voltage or the largest capacity at each voltage compared to [5]. We also evaluate the power, performance, and energy impact of our scheme via architectural simulation.*

2. RELATED WORK

There is a rich body of literature in circuit and architectural techniques for leakage reduction as well as fault tolerance for deep-submicron memories. Two of the best-known leakage reduction works are Gated-Vdd [18] and Drowsy Cache [9]. The former dynamically resizes the instruction cache by turning off blocks which 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 [14, 23] (particularly limiting the mechanism of [9]).

In the fault-tolerance area, works targeting cache yield improvement include error correction codes (ECC) and architectural methods [22, 2, 17, 12, 13, 3, 19], none of which target power reduction as an objective. A variety of low-voltage SRAM cells that improve read stability and/or writability have also been proposed, e.g. 8T [8] and 10T [7], but they have inherently high area overheads compared to a traditional 6T design. Schemes that use fault-tolerance to achieve lower voltage and cache power savings include [25, 1, 20], two very similar approaches [4, 5], as well as [21]. 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., [5]), or boost VDD in "weak" regions as necessary to maintain capacity (e.g., [21]).

Our approach is different from all the related works as follows. We use similar circuit mechanisms as in [18, 9], but combine with fault tolerance to allow lower voltage operation using 6T SRAM cells, although our scheme could also be used with other cell designs. To the best of our knowledge, our scheme is the only one to use voltage scaling along with power gating blocks as they become faulty for additional energy savings. Unlike most recent fault-tolerance works, we use simple mechanisms with no address remapping, data redundancy, sub-block multiplexing, etc., simplifying design and verification while minimizing overheads. Also unlike previous work, we allow for multiple VDD levels that can be set statically (SPCS) or dynamically (DPCS) while tolerating faulty blocks to achieve a desired power/effective capacity point with little additional fault map overhead. We do not correct soft errors, but our scheme could be supplemented with related ECC methods for soft/transient fault tolerance. Finally, we also achieve dynamic power savings as we do not boost the data array SRAM VDD for accesses.

3. POWER/CAPACITY SCALING

Our scheme has two main components: the mechanism, described in Sec. 3.1, and the policy. We propose two different policies, described in Sec. 3.2 and Sec. 3.3, respectively. These policies both use the same underlying mechanism. Together, the mechanism and one of the policies (implemented in hardware or software) comprise the power/capacity scaling cache architecture.

3.1 Mechanism

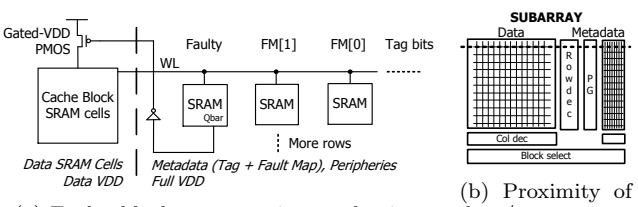
The mechanism consists of a lightweight fault map, populated by a built-in-self-test (BIST) routine, a circuit for globally adjusting the VDD of the data cells, and power gating transistors for each data subarray row, each of which corresponds to (part of) a single cache block. The data array periphery, tag array cells, and tag array periphery are all on a separate voltage domain at the nominal VDD, where they are assumed to be never faulty.

Our low-overhead fault map includes two entries for each data block that are maintained in the corresponding nearby tag subarray in addition to the conventional bits. The first entry consists of several fault map (FM) bits, which encode the lowest non-faulty VDD for the data block. The second entry is a single *Faulty* bit, which indicates whether the block is currently faulty. *Faulty* blocks do not contain valid data and must never allow an access to hit, making them unavailable for reading and writing. Furthermore, *Faulty* blocks must not be used for data placement after a cache miss (e.g., by LRU block replacement strategy).

For N allowed data VDD levels, $\lceil log_2(N+1) \rceil$ FM bits are needed to encode the allowed VDD levels (assuming the *fault inclusion property*, described in Section 1). Fig. 1a shows the circuit concept for a $N = 3$ configuration, requiring two FM bits and one *Faulty* bit per block. For simplicity, we assume three VDD levels are allowed in the remainder of this paper, although our fault map approach should scale well for more voltage levels.

Fig. 1b depicts the higher-level architectural concept. The cache layout is constrained by one major requirement: the tag subarrays should be directly adjacent to their corresponding data subarrays. This is because the *Faulty* bit in the tag array should be physically close to the power gate mechanism for the matching data block. The layout also has the benefit that both subarrays could share one row decoder, although we do not explicitly model this scenario.

It is the responsibility of the power/capacity scaling policy, implemented in the cache controller hardware and/or low-level software, to ensure the *Faulty* and *Valid* bits are set correctly for the current data array cell VDD. The policy must do this before changing the data VDD by comparing each block's FM bits with an enumerated code for the in-



(a) Faulty block power gating mechanism

(b) Proximity of data/tag arrays

Figure 1: Power/Capacity Scaling Cache Architecture Concept

tended VDD level. If the VDD code is less than or equal to the block's *FM* value, then the *Faulty* bit needs to be set; otherwise, it should be cleared. The cache controller must ensure that any block that has *Faulty* set has *Valid* cleared. Otherwise, the cached data might be unreliable.

After all blocks' *Faulty* bits are properly set, the data array VDD transition can occur. When the *Faulty* bit is set, a downward level-shifting inverter, connected directly to the inverted SRAM cell node *Qbar*, controls the data block's power gating transistor. Thus, when *Faulty* is set, the block is immediately power-gated, reducing power further than the savings from pure voltage scaling.

We assume that the power gating implementation is the gated-PMOS configuration from [18], chosen because it has no impact on cell read performance, negligible area overhead, and good energy savings. We model a power-gated block as having zero leakage power, a reasonable approximation because it would likely be gated at a dramatically reduced voltage (compared to nominal VDD) that caused it to be faulty. For voltage scaling, we assume the presence of two voltage domains as described earlier, where the global data array voltage can be set externally from the cache module. The process by which the *Faulty* bit controls the power gate to a data block is also very similar to that of [9]. We verified the feasibility of our mechanism by running simple SPICE simulations to voltage-scale and power-gate SRAM cells.

Our mechanism requires that each set must have at least one non-faulty block at all allowed voltages, as we have no set-wise data redundancy. This is the constraint that limits our min-VDD at a fixed yield target. Higher associativity and/or smaller block sizes naturally result in lower min-VDD, but of course incur other design tradeoffs. Nevertheless, as we demonstrate in Sec. 4, using typical cache configurations, we can still achieve a good power/capacity tradeoff with the set yield constraint.

3.2 Static Policy: SPCS

In the static version of the architecture, we use the lowest VDD level that is likely to have at least 99% effective block capacity (also subject to the constraint mentioned in Sec. 3.1, that all sets must have at least one non-faulty block). This is set statically for the cache runtime, granting both static and dynamic power savings. The only performance overhead to this policy is due to any additional misses caused by the few faulty blocks, if any. This impact is shown to be very low in Sec. 4.2.

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 minimized 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 loss in performance. This phenomenon is described in more detail during the analytical evaluation in Sec. 4.2.

3.3 Dynamic Policy: DPCS

The primary motivation to consider a dynamic policy over SPCS is to exploit situations when the working set size is smaller than the size of the cache. This can occur due to varying behavior across different applications, or during the execution of a single application. For example, with SPCS, if only 40% of the cache is used in a window of execution, the cache is over-provisioned for that interval and costs additional power than is necessary, because it ensures at least 99% of blocks are available. DPCS tries to utilize variations in the working set to reduce power further than what is possible with SPCS while limiting performance degradation. DPCS consists of two parts: a policy and the voltage transition procedure which is tightly woven with the mechanisms described earlier. The proposed DPCS policy is only one of many possibilities.

The policy algorithm (see Listing 1) periodically samples the miss rate over an *Interval* number of accesses, which is used in conjunction with an estimate of the miss penalty to estimate the current average access time ($CAAT$) over that interval. Every *SuperInterval* number of intervals, the algorithm resets the data array VDD to that used by SPCS to measure the nominal average access time ($NAAT$). The most recent $CAAT$ is compared with the latest $NAAT$ using a simple high/low thresholding scheme to determine whether to increase or decrease the operating voltage for the next *Interval* number of accesses. This algorithm accounts for the performance penalty caused by updating the fault maps and changing the voltage in the cache (*DPCSTransitionPenalty* number of cycles).

The transition procedure (see Listing 2), when invoked by the policy, iterates through all cache sets, handling each cache way in parallel. For each block in a set, the algorithm examines the *Faulty* and *FM* bits to determine whether the block will change from faulty to non-faulty or vice versa. Based on this comparison, the block's *Valid* and *Dirty* bits may be used to determine whether the block needs to be written back before invalidation. After these special cases are handled, the block state is cleared (except for *FM*), and the *Faulty* bit is set accordingly. Thus, *DPCSTransitionPenalty* is conservatively estimated as two cycles per set (to read, process, and write metadata using the tag array) plus an additional *Penalty* number of cycles to actually adjust the data supply voltage after the blocks have been handled.

```
every Interval accesses:                                    1
  if IntervalCount == 0:                                    2
    sample NAAT over last Interval accesses                 3
    IntervalCount++                                         4
  else if IntervalCount == SuperInterval - 1:               5
    do DPCSTransition(SPCS_VDD)                             6
    IntervalCount <- 0                                      7
  else:                                                     8
    sample CAAT over last Interval accesses                 9
    if CAAT > (1 + HT) * (NAAT +                            10
          DPCSTransitionPenalty):
      do DPCSTransition(CurrVDD + 1)                        11
    else if CAAT < (1 + LT) * (NAAT +                       12
          DPCSTransitionPenalty):
      do DPCSTransition(CurrVDD - 1)                        13
    IntervalCount++                                         14
```

Listing 1: DPCS Transition Policy

```
routine DPCSTransition(NextVDD):                            1
  delay DPCSTransitionPenalty cycles                        2
  loop Set in CacheSets:                                    3
    parallel[Assoc] Block in Set:                           4
      read metadata bits for Block                          5
      if NextVDD <= Block.FaultMap:                         6
        if Block.Valid and Block.Dirty:                     7
          do Writeback(Block)                               8
        do Invalidate(Block)                                9
        Block.Faulty <- TRUE                                10
      else:                                                 11
        if Block.Faulty:                                    12
          Block.Faulty <- FALSE                             13
  CurrVDD <- NextVDD                                        14
```

Listing 2: DPCS Transition Procedure

4. EVALUATION

4.1 Experimental Methodology

We assessed SPCS and DPCS using a combination of analytical and simulated evaluation (results in Sec. 4.2 and

Table 1: Common gem5 Parameters

Parameter	Value	Parameter	Value
ISA	Alpha	Simulation Mode	Syscall Emul.
CPU Model	Detailed (OoO)	Blk. Repl. Policy	LRU
No. Cores	1	Cache Config.	L1 (Split), L2
No. Mem. Chan.	1	Cache Blk. / Sblk. Size	64 B / 2 B
Memory Model	DDR3-1600 x64	Phys. Mem Size	2048 MB
Fast-forward	1 B inst.	Simulate	2 B inst.
Benchm. Compile Opt.	Base	Input	First ref.

Table 2: System Configurations

Parameter	Config. A	Config. B
Clock Freq.	2 GHz	3 GHz
L1$ Size, Assoc., Hit Lat.	64 KB by 4, 2 cycles	256 KB by 8, 3 cycles
L2$ Size, Assoc., Hit Lat.	2 MB by 8, 4 cycles	8 MB by 16, 8 cycles
No. Data VDDs, FM Bits/Blk.	3, 3	3, 3
L1$ VDD3 (baseline)	1 V	1 V
L1$ VDD2 (SPCS and DPCS)	0.7 V	0.67 V
L1$ VDD1 (DPCS only)	0.6 V	0.54 V
L2$ VDD3 (baseline)	1 V	1 V
L2$ VDD2 (SPCS and DPCS)	0.67 V	0.67 V
L2$ VDD1 (DPCS only)	0.56 V	0.53 V
L1 Interval (accesses)	100,000	100,000
L2 Interval (accesses)	10,000	10,000
SuperInterval (Intervals)	20	20
DPCSTransitionPenalty (cycles)	2 * No. Sets + 20	2 * No. Sets + 40
Threshold (Low/High)	0.05 / 0.10	0.05 / 0.10

Sec. 4.3, respectively). For the analytical portion, we modified CACTI 6.5 [16] to calculate delays, static power, dynamic access energy, and area of each cache configuration for the baseline (which lacks fault tolerance and voltage scalability) and proposed architectures. NFET and PFET on/off current parameters in CACTI were obtained directly from SPICE data. We used MOSFET models from an industrial 45nm SOI process which is the same as that used in the production of our test chip mentioned in Sec. 1. We used the default ITRS 45nm parameters present in CACTI for other technology information such as wire parasitics. SRAM bit cells used regular threshold voltage FETs, while the peripheral logic used low threshold FETs. CACTI generated optimized cache architectures at the nominal voltage of 1 V (specified by the process technology guidelines) using an energy-delay metric. Throughout our analytical and simulated evaluations, all delay, power, and energy figures were based on the CACTI results in 10 mV supply voltage increments.

Our bit error rates (BER) are shown in Fig. 2. These were computed using the data and models from [23], which performed a detailed analysis of SRAM noise margins and error rates for an industrial 45nm technology. Since our proposed mechanism allows for SRAM access at reduced voltage, we adopted the worst case of read, write, and hold static noise margins, namely, the read operation, for the BER. Otherwise, in our evaluation, we did not distinguish between causes of cell failure. Using these BER, we found the probabilities of block failure for each data array voltage, thus allowing us to compute the expected cache capacity and yield. The minimum VDD for all caches was chosen at "design time" such that the expected yield was at least 99%.

To simulate the baseline, SPCS, and DPCS caches, we modified the cache architecture in the gem5 [6] framework to implement our scheme in detail as well as instrument it for power estimation. We used sixteen SPEC CPU2006 benchmarks (integer and floating point) that we were able to compile for Alpha using base-level optimization and run successfully in the simulator. The benchmarks were fast-forwarded for one billion instructions, then simulated in maximum detail for two billion instructions, using the first reference data input. For simplicity, we simulated a single-core system to run the SPEC benchmarks, which are all single-threaded. We ran each benchmark for DPCS multiple times, finding that the impact of random faulty block locations on power and performance was negligible. For example, over five independent runs of *bzip2* for DPCS using fault maps created randomly each time, performance and energy results varied less than 1%. The common gem5 parameters used in all our simulations are summarized in Table 1.

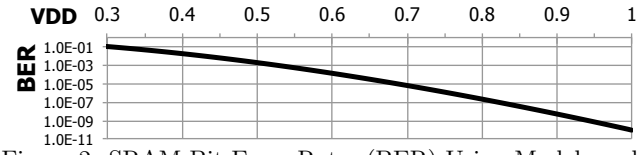


Figure 2: SRAM Bit Error Rates (BER) Using Models and Data From [23]

We had two alternate configurations of the system, which are summarized in Table 2. In Config. A, we matched cache details as closely as possible to that of the reference work, FFT-Cache [5], to allow close analytical comparisons (see Sec. 4.2 for the results). Although we had access to the analytical models used in FFT-Cache, we could not make direct simulated comparisons due to various differences in technology data, power models, and simulator implementations. Unfortunately, because we lacked implementations of other related works, we only compare against their reported results.

Our CACTI results for each cache configuration indicated that within the data array voltage range of interest (above near-threshold), reducing the data cell VDD impacted the overall cache access time by roughly 15% in the worst case. In our simulated evaluations in Sec. 4.3, we make the simplifying assumption that the system clock period is not affected. This is reasonable if the cache access is not the critical path which dictates cycle time. Nevertheless, our hit times in Table 2 reflect the reported worst-case delays from CACTI for each cache at the lowest voltage. We assume that each cache level can be independently voltage scaled from each other and the CPU.

To explore the impact of our policies on a faster system, we quadrupled the size of both the L1 and L2 caches and doubled associativities in Config. B. This was done to see if overprovisioned caches have a significant impact in the results between SPCS and DPCS. Furthermore, our scheme can reach lower voltages using higher associativities for the same cache size, due to the set yield constraint. For DPCS, the *Interval*, *SuperInterval*, *LowThreshold*, and *HighThreshold* variables were set to "reasonable" values to reduce the huge design space and to manage the impact on performance. We assumed that the *Penalty* cycles to scale the data array VDD is enough to minimize noise and allow a stable transition in both cache configurations.

4.2 Analytical Results

Using the derived static power numbers from CACTI and our analytical fault models, we were able to compare our proposed mechanism with that of FFT-Cache [5] (using their original fault tolerance model) and a generic way-granularity power gating scheme using the power vs. effective capacity metric described in Sec. 1.

The analytical results for the L1 Config. A are shown in Fig. 3 (similar results were obtained for the L2 cache as well as both Config. B caches). Our mechanism achieved lower total static power than that of FFT-Cache and generic way-based power gating (which has a linear power/capacity tradeoff) at all effective capacities. This was despite the ability of FFT-Cache to achieve higher effective capacities at all voltages. We estimated that for the Config. A L1 cache, our mechanism achieves 28.2% lower static power than FFT-Cache at the 99% effective capacity level. This is due to the lower overheads of our mechanism compared to the com-

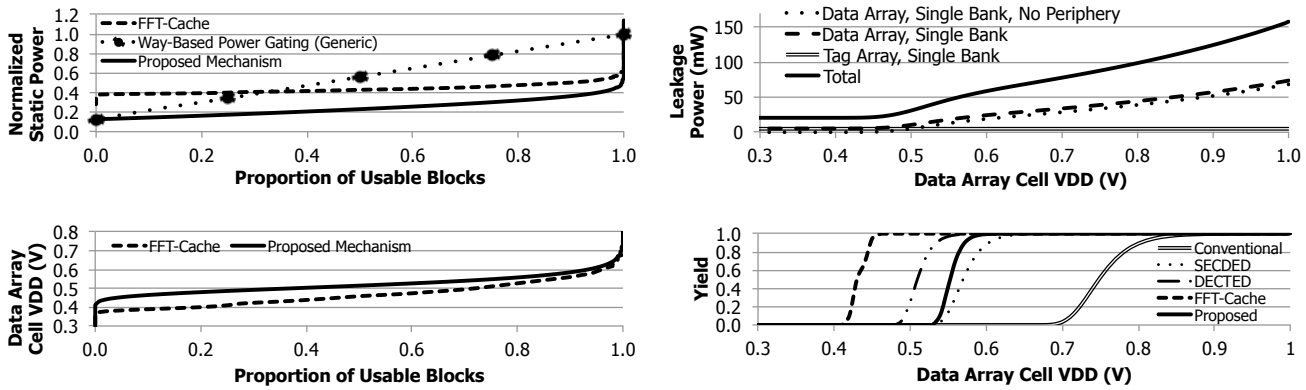


Figure 3: Analytical Results for L1 Config. A

plex fault tolerance of FFT-Cache. The difference arises from a significantly smaller fault map (only three bits per block), as well as application of the fault inclusion property to compress multi-VDD fault maps. In contrast, FFT-Cache needs two entire fault maps for each of the lower VDDs, compounding the existing high overheads. If the number of voltage levels is reduced to two (high/low toggle), the gap between the two schemes shrinks to 17.8% at 99% effective capacity. However, as voltage reduces, the gap increases.

Fig. 3 also shows the yield of the proposed scheme and those of a baseline cache lacking fault tolerance, SECDED and DECTED variants of ECC, and FFT-Cache for the L1 Config. A. Note that SECDED and DECTED were applied at the subblock level of two bytes (see Table 1). Note that our proposed mechanism did not achieve the best min-VDD at fixed yield, but it did better than SECDED in all cache configurations. In the depicted configuration, DECTED achieved slightly better min-VDD than the proposed mechanism due to low associativity, but DECTED typically incurs much higher area, power, and performance overheads to achieve the yields shown [12, 19]. 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 transient faults. Nevertheless, these ECC schemes could be combined with our approach to handle both voltage-induced faults as well as transient soft errors.

Our CACTI results indicated that from the fault map alone, our area overheads compared to a baseline cache lacking fault tolerance did not exceed 4% in the worst case of all configurations. The additional area overheads from the power gating transistor [18] plus small inverter was estimated to be less than 1%. The DPCS policy implementation was 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 5% in the worst case, while in the best case, the area overhead was only 2%. These area overheads are a significant improvement compared to the reported overheads of other FTVS schemes [5] such as 10T SRAM cells (66%), ZerehCache (16%), Wilkerson08 (15%), Ansari (14%), and FFT-Cache (13%).

4.3 Simulation Results

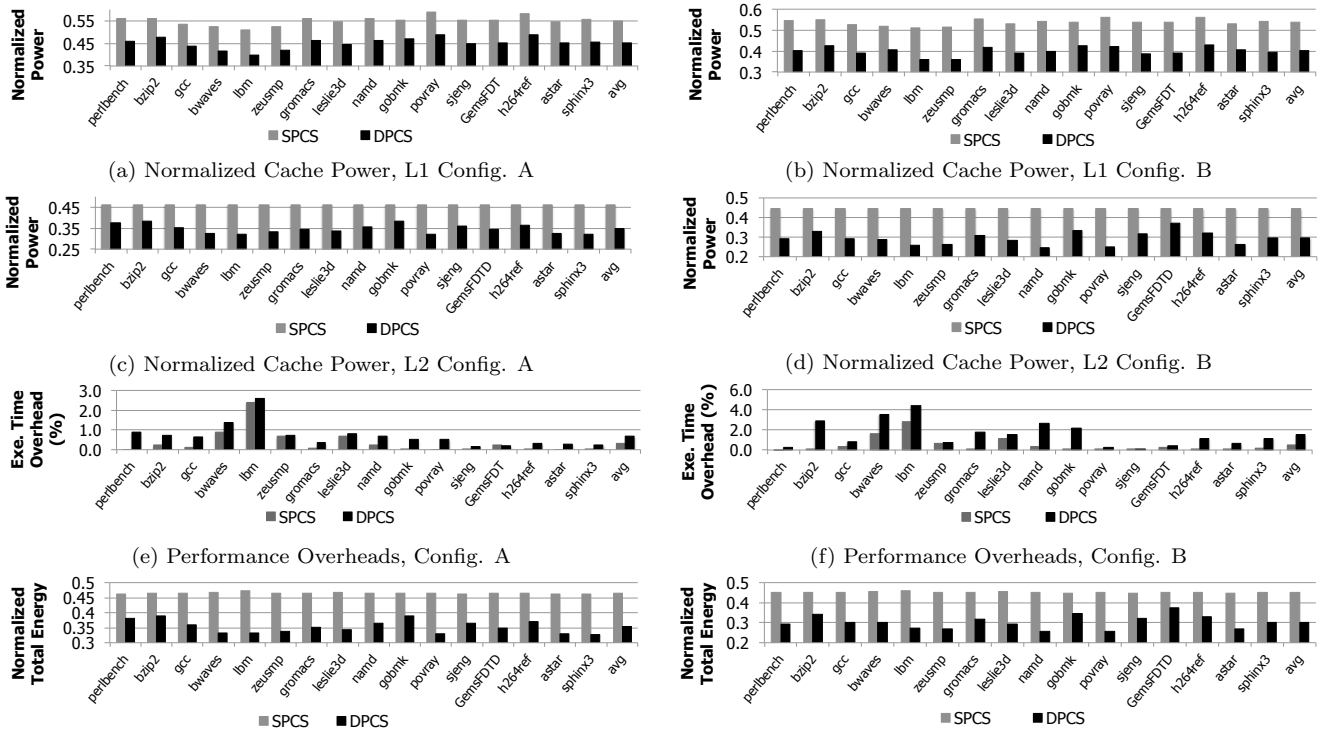
The gem5 simulation results for power, performance, and energy are depicted in Fig. 4. No benchmark suffered more than 2.3% (SPCS) and 2.6% (DPCS) performance degradation for Config. A, and no more than 2.8% (SPCS) and 4.4% (DPCS) for Config. B, compared with the baseline cache architecture at the full VDD of 1 V. The execution time overheads from DPCS were low due to its performance-aware opportunistic policy. The performance impact for SPCS was even lower, as there were no voltage transitions and at least 99% of the nominal capacity was likely to be available. The higher performance hits from Config. B can be attributed to two primary factors: (a) higher DPCS transition latencies due to more sets, and (b) higher configured CPU performance combined with larger miss penalties, amplifying the impact of misses caused by disabled faulty blocks.

SPCS achieved predictable power savings all around, because cache behavior was not significantly impacted at the 99% capacity point, and no voltage transitions occurred. Power savings were almost always better for DPCS, particularly for the larger caches. This was in line with our expectations, as larger caches can afford to sacrifice more blocks at lower VDD. SPCS reduced total energy use by roughly 54% for Configs. A and B on average compared to the nominal cache architecture running at full VDD of 1 V. The majority of the energy savings came from static power reduction thanks to aggressive voltage scaling with power gating of faulty blocks.

In nearly all cases, DPCS yielded significant energy savings compared to SPCS: 23.9% for Config. A and 33.2% for Config. B on average. The corresponding performance overhead of DPCS compared to SPCS was modest. Overall, DPCS achieved lower energy than SPCS because it never used a higher voltage than SPCS, as it would not yield any improvement in cache performance (recall that at least 99% of blocks are non-faulty at the SPCS voltage, i.e., *VDD2*), but it could still drop to *VDD1* when it would not impact performance too much. Reducing voltage further than *VDD1* is not likely to be useful, as the yield quickly drops off and the power savings have diminishing returns with respect to the baseline cache.

5. CONCLUSION AND FUTURE WORK

In this work, we proposed two policy variants of power/capacity scaling caches: static (SPCS) and dynamic (DPCS). The underlying mechanism, used by both policies, achieved better static power/capacity tradeoffs than a recent complex fault-tolerant voltage-scalable (FTVS) approach due to its simple low-overhead implementation. Our mechanism also allows for multiple runtime VDD levels without a significant increase in fault map overhead. An architectural simulation-based evaluation of our scheme showed up to an average of 54.9% (SPCS) and 69.6% (DPCS) total energy savings with performance overheads of 0.6-4.4% for DPCS compared to a baseline cache lacking fault tolerance and running at the nominal 1 V. Our mechanisms incurred an estimated 2-5% total area overhead. We achieved these results with a design-



(a) Normalized Cache Power, L1 Config. A

(b) Normalized Cache Power, L1 Config. B

(c) Normalized Cache Power, L2 Config. A

(d) Normalized Cache Power, L2 Config. B

(e) Performance Overheads, Config. A

(f) Performance Overheads, Config. B

(g) Normalized Total Cache Energy, Config. A

(h) Normalized Total Cache Energy, Config. B

Figure 4: Simulation Results for SPCS and DPCS

time 99% cache yield requirement, even though we do not claim the lowest min-VDD compared to previous work. Potential directions for future work include an investigation of more sophisticated DPCS policies, an evaluation of system-wide power and energy impacts, a broader design space exploration involving multi-core systems with consideration of cache coherence, and a study of the implications for a software stack running on a DPCS-enabled system.

6. ACKNOWLEDGMENT

This work was supported jointly by NSF Variability Expedition grant numbers CCF-1029783 and CCF-1029030.

7. REFERENCES

[1] J. Abella et al. Low Vccmin Fault-Tolerant Cache with Highly Predictable Performance. In *IEEE/ACM MICRO*, 2009.

[2] A. Agarwal et al. A Process-Tolerant Cache Architecture for Improved Yield in Nanoscale Technologies. *IEEE TVLSI*, 2005.

[3] A. Ansari et al. ZerehCache: Armoring Cache Architectures in High Defect Density Technologies. In *IEEE/ACM MICRO*, 2009.

[4] A. Ansari et al. Archipelago: A Polymorphic Cache Design for Enabling Robust Near-Threshold Operation. In *IEEE HPCA*, 2011.

[5] A. BanaiyanMofrad et al. FFT-Cache: A Flexible Fault-Tolerant Cache Architecture for Ultra Low Voltage Operation. In *IEEE/ACM CASES*, 2011.

[6] N. Binkert et al. The gem5 Simulator. *SIGARCH Comp. Arch. News*, 2011.

[7] B. Calhoun and A. Chandrakasan. A 256kb Sub-threshold SRAM in 65nm CMOS. In *IEEE ISSCC*, 2006.

[8] L. Chang et al. Stable SRAM Cell Design for the 32nm Node and Beyond. In *IEEE Symp. on VLSI Tech.*, 2005.

[9] K. Flautner et al. Drowsy Caches: Simple Techniques for Reducing Leakage Power. In *ACM/IEEE ISCA*, 2002.

[10] P. Gupta et al. Underdesigned and Opportunistic Computing in Presence of Hardware Variability. *IEEE TCAD*, 2013.

[11] S. Hamdioui et al. March SS: A Test for All Static Simple RAM Faults. In *IEEE MTDT*, 2002.

[12] J. Kim et al. Multi-Bit Error Tolerant Caches Using Two-Dimensional Error Coding. In *IEEE/ACM MICRO*, 2007.

[13] C.-K. Koh et al. The Salvage Cache: A Fault-Tolerant Cache Architecture for Next-Generation Memory Technologies. In *IEEE ICCD*, 2009.

[14] A. Kumar et al. SRAM Supply Voltage Scaling: A Reliability Perspective. In *IEEE ISQED*, 2009.

[15] L. Lai and P. Gupta. Accurate and Inexpensive Performance Monitoring for Variability-Aware Systems. In *ACM/IEEE ASP-DAC*, 2014.

[16] N. Muralimanohar et al. CACTI 6.0: A Tool to Model Large Caches. Technical report, HP Laboratories, 2009.

[17] S. Ozdemir et al. Yield-Aware Cache Architectures. In *IEEE/ACM MICRO*, 2006.

[18] M. Powell et al. Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories. In *IEEE/ACM ISLPED*, 2000.

[19] D. Rossi et al. Error Correcting Code Analysis for Cache Memory High Reliability and Performance. In *DATE*, 2011.

[20] A. Sasan et al. A Fault Tolerant Cache Architecture for Sub 500mV Operation: Resizable Data Composer Cache (RDC-Cache). In *ACM/IEEE CASES*, 2009.

[21] A. Sasan et al. History & Variation Trained Cache (HVT-Cache): A Process Variation Aware and Fine Grain Voltage Scalable Cache With Active Access History Monitoring. In *IEEE ISQED*, 2012.

[22] P. Shirvani and E. McCluskey. PADded Cache: A New Fault-Tolerance Technique for Cache Memories. In *IEEE VLSI Test Symp.*, 1999.

[23] J. Wang and B. Calhoun. Minimum Supply Voltage and Yield Estimation for Large SRAMs Under Parametric Variations. *IEEE TVLSI*, 2011.

[24] N. H. E. Weste and D. M. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley, 4th edition, 2011.

[25] C. Wilkerson et al. Trading off Cache Capacity for Reliability to Enable Low Voltage Operation. In *IEEE ISCA*, 2008.