A Defect-Aware Reconfigurable Cache Architecture for Low-Vccmin DVFS-Enabled Systems

Michail Mavropoulos, Georgios Keramidas, Dimitris Nikolos
Department of Computer Engineering & Informatics, University of Patras, Greece
{mavropoulo, gkeramidas, nikolosd}@ceid.upatras.gr

***Abstract*—As process technology continues to shrink, a large number of bitcells in on-chip caches is expected to be faulty. The number of defective cells varies from die-to-die, wafer-to-wafer, and in the field of application depends on the run-time operating conditions (e.g., supply voltage and frequency). Those trends necessitate i) to study fault-tolerant (FT) cache mechanisms in a wide spectrum of fault-probabilities and ii) to devise appropriate FT techniques that must be able to adapt their FT capacity to the volume of defective locations of the target faulty caches.**

It is well known that keeping the cache capacity, block size and the volume of defective cells constant, the average number of misses, due to faulty cells, decreases as the associativity of the cache increases. To this end, we propose DARCA, a *D*efect-*A*ware *R*econfigurable *C*ache *A*rchitecture, which is equipped with the ability of dynamically varying its associativity according to the volume of defective cells. To keep the hardware overhead very small, as the associativity of the cache is multiplied by a power of two, its block size is divided by the same number. Since almost all contemporary processors use prefetching, we also applied DARCA to prefetch-assisted caches. By performing cycle-accurate simulations for the SPEC2006 benchmarks assuming a wide range of fault-probabilities, we show that DARCA compares favorably against several known FT cache mechanisms with respect to the performance loss caused by defective cells.

I. Introduction

Reducing the supply voltage (Vcc) is an effective technique to lower the dynamic and static power consumption of ICs. This is mainly true due to the cubic relation between the dynamic power consumption and the supply voltage. Unfortunately, Vcc cannot be reduced to an arbitrary value below a certain level (called Vccmin), because some of the on-chip devices will become unreliable [16]. The biggest failures of this catastrophic effect are the SRAM structures e.g., L1 caches. The reason for this is that voltage scaling exponentially increases the impact of process variations on SRAM cells reliability resulting in an exponential increase in the number of faulty SRAM locations [16]. Failures in memory cells due to voltage scaling typically determines the minimum supply voltage Vccmin at which the cache can reliably operate [16]. As a result, the Vccmin of a processor as a whole is dictated by the Vccmin of its on-chip caches. Thus, it becomes obvious that the biggest roadblock towards near-threshold operation and low power computing is posed by on-chip caches.

Many researchers turned their attention in the area and provided various FT cache schemes. A broad category of techniques for masking out the memory faults relies on over-provision the cache with redundant wordlines [12], an external spare cache attached to an interconnection network [3], or a small fully associative cache [15]. A common characteristic of all prior FT redundancy-based approaches is that the spare resources are organized as extra memory placeholders indented to host both the data and tag parts of the defective cache blocks (we call it *data-based redundancy*). The target is to remap the defective cache parts to the functional spare elements. As noted also by other researchers, data-based redundancy offers a limited capability in the number of faults that can be tolerated [11] (defined by the number of spares). As industry moves into the near-threshold region characterized by high probabilities of failure (pfail), more scalable solutions are desirable.

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales "HOLISTIC". Investing in knowledge society through the European Social Fund.

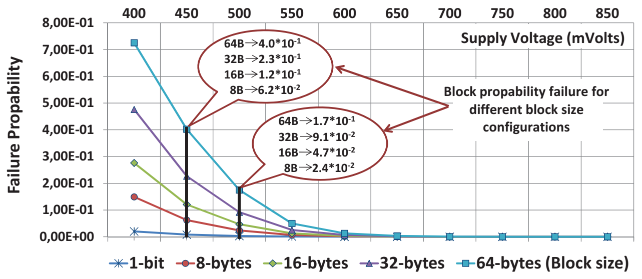

Fig. 1. Probability of persistent errors for different configurations of block size (adopted from [5]).

In this work we opt to follow a different approach (we call it *control-based redundancy*). In our case, the extra storage area is devoted to hold information to control how the faulty cache data array stores and disambiguates the cached data. In other words, no data contexts reside in the spare memory components. More specifically, we choose to use the redundant storage to include control information (extra tag and status bits) to dynamically reconfigure the cache block size. As we show in this paper, the proposed direction results in a remarkably higher FT capacity especially in high failure rate situations (e.g., during near-threshold execution in voltage scalable caches).

This work makes the following contributions:

- *The need for defect-driven reconfigurations:* As a first step, we quantify the impact of cache line size in the performance of faulty caches for a wide spectrum of pfails. In low pfails, large block sizes tend to exploit application spatial locality, whereas in high pfails smaller line sizes result to higher fault-free cache capacity (Fig. 1 illustrates the latter trend).
- *Defect-Aware Reconfigurable Cache Architecture (DARCA):* DARCA is suitable (due to its simplicity) for L1 faulty caches. The key concept is to adapt the block size of target caches with respect to the above-mentioned trends.

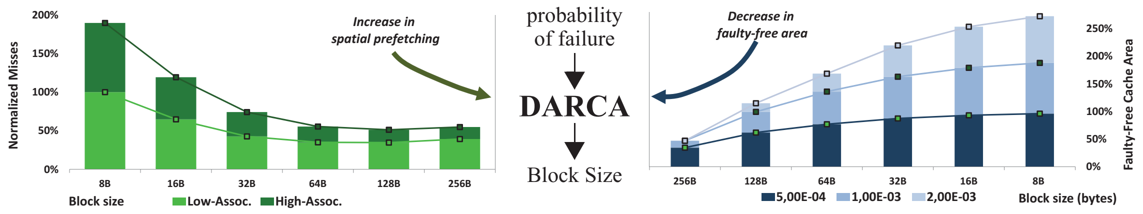

Fig. 2. Spatial locality (left graph) and faulty-free area (right graph) for different configurations of block size (the bars are in stacked format).

- *DARCA reconfiguration policy:* We present an experimental methodology to extract the reconfiguration policy of DARCA based on the target pfail. Our proposal is suitable for low Vccmin DVFS-enabled systems.
- *Impact of prefetching:* We thoroughly examine the efficiency of DARCA in prefetch-assisted caches. Using two prefetcher configurations, our results reveal that an additional asset of DARCA is that exhibits a prefetch-friendly behavior in contrast to prior FT techniques.
- *Control vs data redundancy for stronger fault-tolerance:* We question about the most appropriate type of redundancy (additional storage area) in faulty caches. Our results showcase that DARCA control-based redundancy offers a significantly higher and scalable FT capacity.

We evaluate DARCA using cycle-accurate simulations, SPEC2006 benchmarks, and 30 fault maps assuming a wide spectrum of pfails (10 pfails); prior cache FT papers examine a limited number (up to three) of pfails. In addition, we compare our approach against various FT schemes (coarse-grain and fine-grain block disabling schemes) and a FT cache redundancy scheme. DARCA reduces *miss ratio* due to faults by 71.8% and 51.6% compared to a well-known block disabling scheme and a cache redundancy mechanism, respectively. Furthermore, when the target faulty cache is assisted by a stride prefetching mechanism, DARCA manages to remarkably increase its distances from the previous FT schemes. Finally, we rely on CACTI 6.5 [14] to evaluate the latency and area overheads of our proposal.

Structure of this paper. Section II motivates this study. Section III presents the organization of DARCA. Section IV outlines our evaluation framework. Section V provides our experimental findings. Section VI puts this work in context of related work and Section VII concludes this paper.

II. Motivation: Trading Fault-Free Cache Capacity for Spatial Locality

Cache design is a multi-parametric optimization process involving several parameters e.g., block size, associativity, lookup overheads etc. The centerpiece of the cache parameters is the block size. Small blocks sizes tend to fetch fewer unused words, but impose significant performance penalties by missing opportunities for *spatial prefetching*. Larger line sizes minimize tag overhead and effectively prefetch neighboring words (spatial prefetching), but introduce the negative effect of unused words that increase network bandwidth.

The left graph in Fig. 2 quantifies the impact of spatial prefetching. Omitting at this point the simulation details, the reported statistics show the average number of misses for the block granularities shown in the x-axis. The light green bars/lines corresponds to a low-associative cache configuration while the dark green bars/lines corresponds to high-associative caches. All the results are normalized to the low-associative/8-bytes-block point. The trends in Fig. 2 are clear: increasing the block size (increasing spatial prefetching) leads to a substantial reduction in the numbers of misses. Indeed, a large 64- or 128-bytes block size is used in most commercial products (lines larger than 128 bytes result in poor cache line utilization).

Traditionally, the target objective of defining the ideal cache block granularity is to strike a balance between miss rate, bandwidth usage, and cache space utilization. However, in this paper, *we advocate that cache fault-tolerance capacity must be also taken into consideration as first-class cache design parameter*. The right graph in Fig. 2 depicts the impact of block granularity in the error-free space in process variation affected caches when the block disabling technique is employed. According to block disabling, the cache is equipped with the ability to disable physical cache frames that contain faulty bits upon permanent error detection. In general, block disabling is an attractive option because of its low overhead i.e., 1-bit, called *faulty bit*, is attached to each block (e.g., 1-bit for 64 bytes). Consequently, the right graph in Fig. 2 analyses the impact of increasing the block granularity (x-axis) in the fault-free cache area[1] for three pfails. In all cases, we assume that one faulty bit is assigned to each cache block independently of the selected block size[2].

As the graph indicates (also noted in [10]), there is a strong correlation between the selected block size and the non-faulty cache area. It is evident that smaller block sizes translate into higher fault-free cache capacity. For example, in the high pfail, a 256-byte block disables the whole cache while a 8-byte block results in a 85% faulty-free area. Thereby, by decreasing the block size, the effect on cache capacity of single faulty cell decreases and, therefore, overall capacity increases. However, the capacity increase comes at the expense of lower spatial locality (left graph in Fig. 2).

Consequently, having highlighting the two conflicting trends (spatial locality vs. fault-free area), in this work we propose a reconfigurable cache architecture (DARCA) which is designed to dynamically adapt the block size of faulty caches. The two extreme cases are the following: in low fault rates, large block sizes tend to exploit application spatial locality, whereas in high fault rates smaller line sizes result to higher fault-free cache capacity. Of course, DARCA is able to pinpoint the correct block size in all the in-between pfails. As shown in the central part of Fig. 2, DARCA takes as input the cache pfail and outputs the most beneficial cache configuration.

III. Organization of the Proposed Mechanism

Since our intention is to introduce minimal changes in the underlying cache structure, *we opt to vary the cache block size in relation to cache associativity*. For example, if the associativity is multiplied by two, the block size is divided by the same factor. The main benefit of this design decision is that the data array of the cache remains intact.

In the context of this work, we assume that DARCA can be reconfigured as (block_size_bytes/assoc): 64/2, 32/4, and

[1] The fault-free cache area is defined as the percentage of the non-faulty blocks over the total cache blocks.

[2] Note that this analysis is independent of cache associativity.

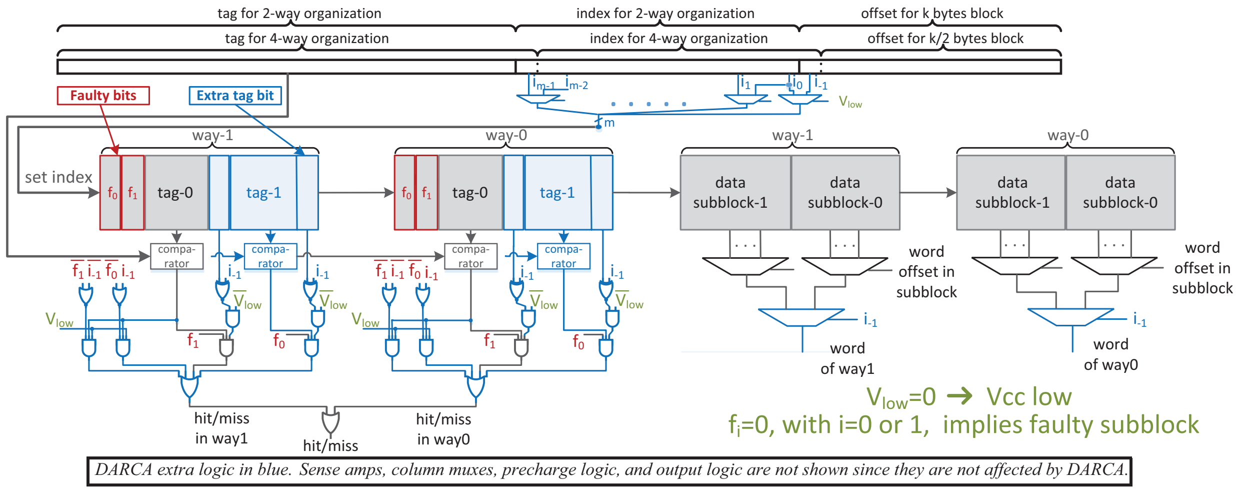
tag for 2-way organization
index for 2-way organization
offset for k bytes block
tag for 4-way organization
index for 4-way organization
offset for k/2 bytes block
Faulty bits
Extra tag bit
Vlow
set index
way-1
way-0
f0 f1 tag-0 tag-1
data subblock-1
data subblock-0
comparator
word offset in subblock
word of way1
word of way0
hit/miss in way1
hit/miss in way0
hit/miss
Vlow=0 → Vcc low
fi=0, with i=0 or 1, implies faulty subblock
DARCA extra logic in blue. Sense amps, column muxes, precharge logic, and output logic are not shown since they are not affected by DARCA.

Fig. 3. The anatomy of DARCA.

16/8 (although different configurations are possible). This implies that the baseline cache will operate as a 64-bytes/2-way cache and when the number of defective memory cells increases, a different configuration must be selected. As noted, the selection of the best-performing reconfiguration is done as a function of fault-rate and spatial locality. Thus, since DARCA can operate in three different block sizes, the baseline cache (64/2) must be enhanced with six additional tag-arrays[1] (containing separate tag, valid, status, faulty, and replacement bits). Consequently, when the block disabling technique is employed, the baseline cache will be able to keep data in cache lines even if they have some faulty subblocks i.e., four faulty-bits are attached in each 64-bytes physical cache frame or one faulty-bit every 16-bytes [1]. Those faulty-bits will track which subblocks are faulty, so as, hits in those subblocks can be detected (this scheme is typically referred as *subblock disabling*). Note that in all configurations, block disabling is applied in sub-sub-block level (16-bytes). For clarity, the following description of DARCA will be done assuming that DARCA can be configured only as 64/2 and 32/4 cache.

The black/grey parts of Fig. 3 show the functionality of a conventional cache and the blue parts depict the extra circuitry required by DARCA. The whole operation of DARCA is controlled by the V_{low} signal. If V_{low} signal is set (clear), the 64/2 (32/4) configuration is forced. As noted, the main modification required by DARCA is the addition of duplicate tag arrays in each associative way. The tag arrays named as "tag-1," in Fig. 3 correspond to the extra tag arrays (used only in the 32/4 point), whereas "tag-0" arrays are the original tag arrays of the 64/2 configuration. Apart from the extra tag structures, DARCA requires minimal additional logic. Since smaller block sizes use a smaller number of offset bits, a different number of tag bits should be used during the tag comparison process for each of the individual block sizes offered by DARCA. This issue is addressed by the logic (in blue color) at the bottom of the tag arrays. The idea is to control via V_{low} if the MSB of the index part of the address must participate in the hit/miss decision logic. In addition, appropriate logic must be also introduced to select which bits of the address are needed to formulate the cache index. This is implemented through a series of one-level 2-1 multiplexers. Note that the number of index bits remains the same independently of the selected DARCA configuration.

Storage requirements. Considering a baseline 16KB/64-bytes/2-way cache and a DARCA configuration that can operate in three different block sizes (64B, 32B, and 16B), the memory overheads of our approach are the six extra tag arrays. For a 32-bit logical address space, those overheads are: 768 (entries) x 24 bits (19 tag bits, 2 status bits, 2 LRU bits, 1 valid bit)=2304 bytes. In addition, extra tag bits must be inserted in the original tag arrays. This extra memory is 64 bytes. Thus, the overall memory requirements of DARCA are 2368 bytes or 13.8% over a conventional cache[2]. Note that if we consider only two reconfiguration points (e.g., 64/2 and 32/4), the memory overheads are 6.9%.

However, for a fair comparison, the overheads of DARCA must be compared against prior redundancy-based FT schemes. A typical data-based redundancy (DBR) scheme enhances the faulty cache with additional memory placeholders (cache lines). For a 32-entry/64-byte-block DBR scheme, the extra storage is equal to 2184 bytes or 12.7% over a conventional cache. Thereby, the storage requirements between DARCA and a typical 32-entries DBR scheme are comparable. However, DARCA follows a different approach. The extra storage area is devoted to hold information to control how the cache data array disambiguates the cached elements (***control-based redundancy***). Our results indicate that DARCA approach offers a significantly higher FT capacity.

Time considerations. DARCA does not introduce extra delays in the cache access path. This is important since our target is to increase the reliability of the latency-sensitive L1s. In order to calculate access time, we modified CACTI source code to integrate the DARCA extra components into the design. In particular, the extra components are i) the variable tag selection logic (below tag arrays), ii) the index bit selection logic (before cache decoder), and iii) the extra level of multiplexer (below data arrays). The latter component does not add additional logic over a baseline cache, because ending up with a multi-level multiplexer design is already part of CACTI normal optimization process. The other two components reside on the critical path, but *they do not affect the cache cycle (pipeline) time*. This is because under today's wire-limited technology, cache latency is dominated by the bit/wordlines

[1] Based on the configuration, some or all those extra arrays can be turned-off to avoid leakage power loss using previously proposed leakage savings techniques like gated-V_{DD}.

[2] Since our proposal is built on top of the subblock disabling scheme, we do not account the faulty flag bits as extra overhead of DARCA.

Table 1. Stride-prefetcher configuration.

Configuration	Conservative	Aggressive
Reference Prediction Table	32 entries, fully-assoc.	128 entries, fully-assoc.
Confidence	2 bits	2 bits
Degree	1 stride	4 strides
Stride size	8 bits	8 bits
Input/Output Queue	16/16 entries	16/16 entries

and h-tree traversals. Indeed, our CACTI simulations for 22nm and 32nm process technologies show that the cache cycle time is dictated by the data memory array access path (delay inside MAT plus the wordline/bitline reset/restore delays). We add the delay of the index selection logic (2 gate levels) to the decoder pipeline stage and the delay of variable tag selection logic (3 gate levels) to the tag comparison pipeline stage. As expected, in both cases, the cache cycle time remained unaffected. Thereby, DARCA does not influence the operational frequency of the target faulty cache.

IV. Evaluation Framework

Simulation infrastructure. We performed cycle accurate simulations using a dynamic 2-issue core. In this work, we assume that faults are injected only in L1 data caches. In addition, we assume that DARCA can be reconfigured to the following cache organizations: *64-bytes/2-way (baseline), 32-bytes/4-way, and 16-bytes/8-way*. The reconfiguration policy of DARCA is presented in Section V. Finally, we use 18 SPEC2006 benchmarks and we simulate the largest-weight 200-million-instruction Simpoint sample per benchmark.

Fault model. In accordance to the *graceful degradation* paradigm, faulty caches are equipped with the ability to disable cache portions such as blocks, subblocks, or words that contain faulty bits upon permanent error detection. An obvious design trade-off is to identify the appropriate number of divisions that each cache block should be split. Each block division will be guarded by a separate faulty bit and all the faulty bits will constitute the cache *fault map*. As noted, we assume that the cache disabling scheme is applied in a sub-subblock granularity i.e., a separate faulty flag is assigned to every 16-bytes independently of the DARCA operating point. In addition, a random distribution of faulty cells is considered. In all cases, faulty bits can be part of both tag and data arrays of the target cache, while for the valid and dirty bits, faulty bits, and replacement bits is assumed that are not affected by process variation-induced errors e.g., by employing more reliable SRAM cells [9]. Furthermore, for each program run, we performed 30 times the simulation for different faulty cell locations and we average the obtained results. Finally, since our effort is to provide a defect-driven reconfigurable cache architecture, we study a wide spectrum of SRAM pfails —10 in total— ranging from 1e-04 up to 5e-03. The relationship between pfail and Vcc is depicted in Fig. 4.

Reconfiguration Strategy. Fig. 4 presents also the reconfiguration policy of DARCA. Our goal is to tailor the cache block size to the selected Voltage/frequency (V/f) level of the cache. As shown in Fig. 1, on aggressive V/f scaling, the SRAM pfail ramps up exponentially. In this work, our view is not to identify the optimal V/f point, but to effectively reconfigure the cache structure based on the chosen V/f level. In other words, the chosen V/f point (and respectively the pfail) is just an input to our methodology.

Prefetching. As it will become evident in Section V, one of the main assets of DARCA is that it exhibits a prefetch-friendly

Vdd(mV):	850	625	575	537	526	518	509	500	487	478	465
Faults:	0	13	74	149	220	297	366	441	512	583	729
Conf:	**64/2**	**64/2**	**64/2**	**64/2**	**32/4**	**32/4**	**16/8**	**16/8**	**16/8**	**16/8**	**16/8**

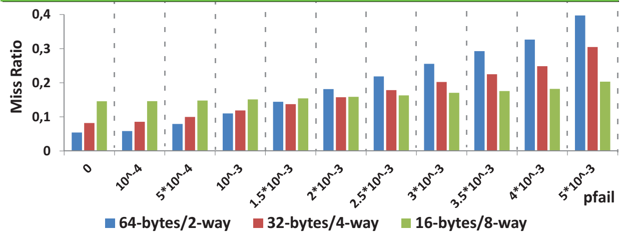

Fig. 4. DARCA reconfiguration policy.

behavior in contrast to (sub)block disabling techniques that are proved to operate in a prefetch-hostile manner. *To the best of our knowledge, this is the first work that examines the impact of prefetching in faulty caches.* For our elaboration, we rely on a common stride prefetcher implemented as in [4]. Stride prefetchers do not require much state because each entry can prefetch an entire stream that may span several tens or hundreds of cache lines. In this work, we use two prefetch configurations representative of a conservative and a more aggressive prefetch scenarios (Table 1). Finally, in order to get a more clear understanding of the behavior of the prefetcher, we assume that the prefetcher structures are error-free.

Data-Redundancy. Typical redundancy-based FT schemes [2,3,12,15] enhance the faulty cache with additional blocks to remap the defective cache blocks to the functional spare blocks (*data-based redundancy* or DBR). For comparison reasons, we implement the spare cache technique [15]. Spare cache is a small fully associative cache accessed in parallel to the primary cache. Read/write requests referring to faulty blocks in the primary cache are served by spare cache. Contrary, an access to a non-faulty cache location is exclusively managed by the primary cache. As in [15], the eviction policy of spare cache is based on LRU. Finally, according to the assumed fault-model, spare memory structures are prone to errors similarly to the tag/data arrays of primary cache.

V. Results

The first part of section presents our experimental approach to extract the reconfiguration policy of DARCA. In the second part, DARCA is compared, in a per-benchmark basis, against various FT schemes assuming a high pfail situation. The third part depicts our overall range of results for all studied pfails. In the two latter parts, special effort is given to reveal the efficiency of DARCA in prefetch-assisted caches.

Reconfiguration policy. The premise of this work is to offer a reconfigurable cache which is able to morph its internal architecture to the volume of defective memory cells. We assume that DARCA can be reconfigured as (block_size_bytes/assoc): 64/2, 32/4, and 16/8. The vertical axis in Fig. 4 shows the reported miss ratios (averaged for all benchmarks), while the pfails (including the error-free case tagged as “0”) are depicted in x-axis. For the sake of completeness, the number of faulty cells and the voltage levels corresponding to each pfail are presented. Each group of bars in Fig. 4 reflects the miss ratio of the three DARCA operational points. Finally, the green box contains the selected reconfiguration points i.e., the DARCA reconfiguration policy.

As illustrated in Fig. 4, in low pfails (up to 1e-03, including the error-free case) the 64/2 configuration exhibits the higher miss-savings-capacity due to higher opportunities for spatial prefetching. As pfail increases, a different DARCA

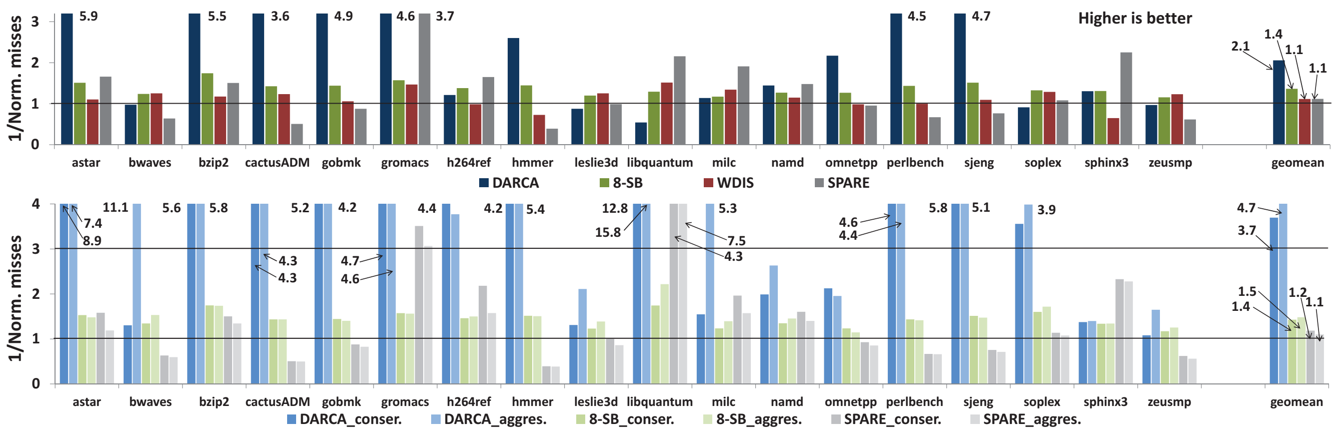

Fig. 5. Comparison between DARCA, fine-grain block disabling techniques, and SPARE without (top graph) and with (bottom graph) prefetching (pfail=2.5e-03).

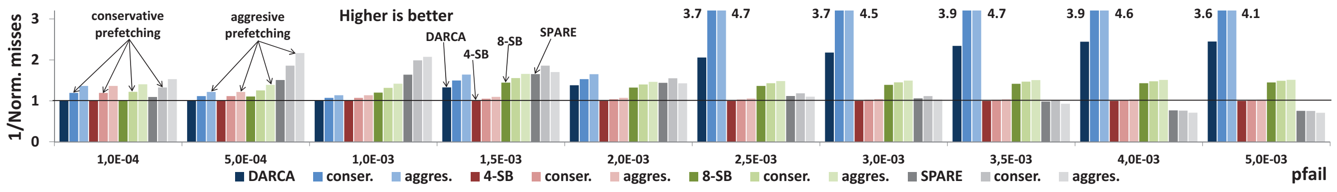

Fig. 6. Comparison for 10 fault probabilities between DARCA, two block-disabling techniques of different granularities, and SPARE. For each technique the impact of conservative and aggressive prefetching is presented.

organization is needed. The 32/4 configuration is proved to be the best option for pfails up to 2e-03, whereas in even higher pfails, DARCA must be configured as a 16/8 cache. In the rest of this section, DARCA will be adapted based on the just-mentioned reconfiguration policy. In addition, we have to note, that even in this work we follow a statistically-defined reconfiguration approach, more dynamic policies tailed to the unique memory behavior of individual applications are also possible to device, however this analysis is left for future work.

Per-benchmark statistics. The graphs in Fig. 5 illustrate our gathered results in a per-benchmark basis for a 2.5e-03 pfail (DARCA is morphed as a 16/8 cache). The impact of prefetching is depicted in the bottom graph (the top graph corresponds to the no-prefetching case). In both figures the results are given with respect to the sub-subblock disabling scheme, according to which one faulty bit is assigned in every 16-bytes (termed as 4-SB hereafter). In particular, in the y-axis, the following metric is used:

1/norm._misses=misses_of_4-SB/misses_of_a_technique.

In the top graph, the following schemes are compared: i) DARCA (blue bars), ii) 8-sub-block (8-SB) corresponding to a word-level disabling scheme —one faulty bit is assigned in every 8 bytes (green bars), iii) WDIS [16] (red bars), and iv) SPARE, a cache with block disabling coupled together with a 32-entries spare cache (grey bars).

As Fig. 5/top graph indicates, our approach shows noticeable improvements over all competitive techniques (even the SPARE technique!). DARCA manages to ameliorate the reported misses compared to 4-SB by more than 20% in 15 benchmarks (out of 18). The 8-SB scheme also shows noteworthy improvements over 4-SB. This is the result of recruiting the block disabling technique in a smaller granularity, thus a larger fault-free area is made available to cache insertion/eviction logic (17.8% in this case). In addition, WDIS and SPARE end up with almost equivalent miss savings capabilities. On average and as shown by the rightmost group of bars, our proposal reduces the number of misses by 33.7%, 45.7%, and 45.6% over 8-SB, WDIS, and SPARE respectively.

Moreover, when prefetching in employed (bottom graph of Fig. 5), DARCA manages to further increase its distance from prior FT schemes. The graph presents our gathered results when a conservative and a more aggressive stride-prefetcher is utilized (Section IV). Attached to each benchmark, there are six bars. The blue bars correspond to DARCA, the green bars to 8-SB, and the gray bars to SPARE. In all cases, the conservative prefetching scenario is depicted by the dark bars, while the light ones show the benefits provided when an aggressive prefetcher is utilized.

By comparing the two graphs of Fig. 5 (with and without prefetching), the same trends appear in all benchmarks. DARCA_aggres. surpasses all the other techniques in 17 out of 18 benchmarks, while DARCA_conser. is better than 8-SB and SPARE in 14 benchmarks. Those results prove that DARCA exhibits a prefetch-friendly behavior and it is able to report benefits in both prefetch configurations. In particular, the miss-savings of DARCA_conser. and DARCA_aggres. are increased by 44.3% and 56% respectively over the no-prefetch-assisted DARCA. In contrast, the impact of prefetching in 8-SB varies from 5.6% (conser. case) to 8% (aggres. case). The results in SPARE are even worsen. A meager improvement (5.4%) appears by the conservative prefetcher, while a cache performance reduction of 1.44% is observed when the aggressive prefetcher is employed.

Overall statistics. Finally, Fig. 6 shows our full range of results for all studied pfails. The graph in Fig. 6 contains four distinct FT mechanisms marked by different colors. The four mechanisms are: DARCA (blue bars), 4-SB (red bars), 8-SB (green bars), and SPARE (grey bars). The dark bars correspond to one of the four basic mechanisms and the two light bars at the right part of each dark bar depict the impact of conservative and aggressive prefetching. To the best of our knowledge, *this is the first approach that re-examines the effectiveness of prior FT schemes in a such wide range of pfails.*

Several interesting results can be drawn from Fig. 6. Due to space reasons, we concentrate on supporting the main reasoning behind this work which is the control vs. data redundancy trade-off on cache reliability. DARCA and SPARE are appealing in the sense that both require minimal hardware modifications. Indeed, as Fig. 6 indicates, in low pfails (up to 1e-03), the SPARE technique shows significant miss-savings capabilities and it cooperates in a constructive way with the underlying prefetching mechanism. However, as we move to higher pfails (dictated by upcoming near-threshold process technologies), the strength of SPARE is discredited. Note that the two schemes require almost the same amount of extra storage and for a fair comparison we assume that in both cases this extra storage is amenable to process-variation-induced errors.

Contrary, DARCA shows a remarkable miss-recovery scalability in higher pfails. As it is evident from Fig. 6, DARCA is clearly the best performing mechanism in pfails higher than 2e-03. Apart from that, DARCA is the only mechanism that is able to improve cache performance in the aggressive prefetching case. For example, under the latter scenario, the distance between DARCA and SPARE starts from 0.22x in the 2e-03 pfail and ramps up to 3.37x in the 5e-03 pfail. Another interesting remark is the outstanding improvement in the effectiveness of aggressive prefetcher beyond the 2.5e-03 pfail. This is because at the specific pfail, DARCA is configured to operate as 16/8 cache (inflection point between 32/4 and 16/8 configurations). Consequently, the 16/8 configuration depicts a strong prefetch-friendly behavior. This observation necessitates a revert-back step in order to re-extract the reconfiguration policy of DARCA taking into account the impact of prefetching. This represents a promising direction for future work.

VI. RELATED WORK

Cache fault-tolerance techniques can be classified into four categories. The first category includes circuit-level techniques [9], however these approaches incur large area overheads and require considerable effort to redesign the cache structure. The second class of FT schemes relies on the usage of error correcting codes (ECC) to detect/correct the hard errors. Unfortunately, the use of ECC is not practical for hard errors due to its large storage overhead and ECC repair time penalty. The third category of proposals relies on redundancy and tries to substitute the defective cache parts with functional spare elements in a static [3,12] or dynamic fashion [2,15]. The common ground in these redundancy schemes is that the spare resources are organized as extra placeholders indented to host both the data and tag parts of the defective cache blocks. However, this approach offers a limited scalability (defined by the number of spares) in the faults that can be tolerated. DARCA also over-provisions the cache with extra storage, but in our case the additional memory is used to hold information (extra tag and status bits) to control how the cache data array stores and disambiguates the cached data. As proved, our proposal results in a significantly higher FT capability.

The fourth set of methods of FT caches is based on restructuring the cache in order to combine partially faulty physical blocks to get a smaller number of non-faulty blocks [8,13,16]. A nice survey of these techniques can be found in [6]. These techniques increase the critical path delay of the cache and they are particularly expensive. For example, the BFIX technique [16] requires around 250 10-input 2-bit wide multiplexers to align the appropriate non-faulty cache bits. On the other hand, DARCA does not influence the cache critical path and introduces minimal modifications in the underlying cache logic. In addition, DARCA is the first proposal that is able to adjust its operation based on the target pfail in relation to the spatial locality of the executing applications. Finally, a new architectural-level technique was recently proposed [7]. The authors introduced a FT-aware insertion/eviction policy dedicated for only 2-way associative caches.

VII. CONCLUSIONS

In this work, we propose DARCA, a fault tolerant aware scheme which leverages previous block disabling schemes in a reconfigurable manner. At the circuit level, DARCA is equipped with the ability of dynamically varying the block size of faulty caches. At the microarchitectural level, the target block size is defined as a function of fault-rate. Our analysis shows that by employing a simple reconfiguration policy, DARCA exhibits significant fault recovery capacity compared to previous FT techniques. In addition, we show that DARCA exhibits a prefetch-friendly behavior (in contrast, previous FT approaches show a prefetch-hostile behavior). Finally, we question about the most appropriate type of redundancy, data based or control based. We showed that control based redundancy (as in DARCA) offers a more scalable FT capacity over previous data based redundancy FT techniques.

REFERENCES

[1] J. Abella, J. Carretero et al., "Low Vccmin Fault-Tolerant Cache with Highly Predictable Performance," in *Proc. of Intl. Symposium on Microarchitecture*, 2009.

[2] J. Abella, E. Quinones et al., "RVC: a Mechanism for Time-Analyzable Real-Time Processors with Faulty Caches," in *Proc. of Intl. Conference on High Performance and Embedded Architectures and Compilers*, 2011.

[3] A. Ansari, S. Gupta et al., "Zerehcache: Armoring Cache Architectures in High Defect Density Technologies," in *Proc. of Intl. Symposium on Microarchitecture*, 2009.

[4] J. Baer and T. Chen, "Effective Hardware-Based Data Prefetching for High-Performance Processors," in *Transactions on Computer*, 1995.

[5] Z. Chishti, A. Alameldeen et al., "Improving Cache Lifetime Reliability at Ultra-Low Voltages," in *Proc. of Intl. Symposium on Microarchitecture*, 2009.

[6] Y. Choi, S. Yoo et al., "MAEPER: Matching Access and Error Patterns With Error-Free Resource for Low Vcc L1 Cache," in *Transactions on VLSI*, 2013.

[7] G. Keramidas, M. Mavropoulos et al., "Spatial Pattern Prediction Based Management of Faulty Data Caches," in *Proc. of Intl. Conference Design, Automation, and Test in Europe*, 2014.

[8] C.K. Koh, W.F Wong et al., "Tolerating Process Variations in Large Set Associative Caches: The Buddy Cache," in *Transactions on Architecture and Code Optimization*, 2009.

[9] J. Kulkarni, K. Kim, and K. Roy. "A 160 mV Robust mitt Trigger Based Subthreshold SRAM," in *Journal of Solid-State Circuits*, 2007.

[10] N. Ladas, Y. Sazeides, and V. Desmet. "Performance-Effective Operation below Vcc-min," in *Proc. of Intl. Symposium on Performance Analysis of Systems & Software*, 2010.

[11] H. Lee, S. Cho, and B.R. Childers. DEFCAM: A design and evaluation framework for defect-tolerant cache memories," in *Transactions on Architecture and Code Optimization*, 2011.

[12] T. Mahmood and S. Kim. "Realizing Near-True Voltage Scaling in Variation-Sensitive L1 Caches via Fault Buffers," in *Proc. of Intl. Conference on Compilers, Architectures and Synthesis for Embedded Systems*, 2011.

[13] A.B. Mofrad, H. Homayoun, and N. Dutt. "FFT-cache: a Flexible Fault-Tolerant Cache Architecture for Ultra Low Voltage Operation," in *Proc. of Intl. Conference on Compilers, Architectures, and Synthesis for Embedded Systems*, 2011.

[14] N. Muralimanohar, R. Balasubramonian, and N. Jouppi. "CACTI 6.0: A Tool to Model Large Caches," *Technical report*, *HP Laboratories*, 2009.

[15] H.T. Vergos and D. Nikolos. "Performance Recovery in Direct-Mapped Faulty Caches via the Use of a Very Small Fully Associative Spare Cache," in *Proc. of Intl. Computer Performance and Dependability Symposium*, 1995.

[16] C. Wilkerson, H. Gao et.al. "Trading off Cache Capacity for Reliability to Enable Low Voltage Operation," in *Proc. of Intl. Symposium on Computer Architecture*, 2008.