Performance-Effective Operation below Vcc-min

Nikolas Ladas, Yiannakis Sazeides
University of Cyprus

Veerle Desmet
Ghent University, Belgium

Abstract

Continuous circuit miniaturization and increased process variability point to a future with diminishing returns from dynamic voltage scaling. Operation below Vcc-min has been proposed recently as a mean to reverse this trend. The goal of this paper is to minimize the performance loss due to reduced cache capacity when operating below Vcc-min. A simple method is proposed: disable faulty blocks at low voltage. The method is based on observations regarding the distributions of faults in an array according to probability theory. The key lesson, from the probability analysis, is that as the number of uniformly distributed random faulty cells in an array increases the faults increasingly occur in already faulty blocks. The probability analysis is also shown to be useful for obtaining insight about the reliability implications of other cache techniques.

For one configuration used in this paper, block disabling is shown to have on the average 6.6% and up to 29% better performance than a previously proposed scheme for low voltage cache operation. Furthermore, block-disabling is simple and less costly to implement and does not degrade performance at or above Vcc-min operation. Finally, it is shown that a victim-cache enables higher and more deterministic performance for a block-disabled cache.

I. Introduction

While continuous technology miniaturization enables the doubling of transistors in a chip with every new technology node, supply voltage decrease-rate is slowing down. This alarming trend [21] can translate to unrestrained power growth which can not be sustained due to cost, battery, power-delivery, reliability, and environmental concerns. What is more, such a development may preclude the simultaneous use of all available resources on a chip and, therefore, limit or even halt the performance/cost gains from technology scaling. Consequently, on-chip power is a prime constraint in the design of modern processors [2], [13], [9] and techniques for mitigating it are at the focus of numerous research studies.

One of the most effective and widely used methods to limit power is dynamic voltage scaling [16]. It enables cubic reductions in the dynamic power at the expense of a linear frequency decrease. Scaled down voltage, also, means lower static power since leakage currents have a dependence on supply voltage.

However, voltage can not be reduced to an arbitrary value because below a certain voltage, known as minimum supply voltage or Vcc-min, some on-chip devices become unreliable. Consequently, the overall Vcc-min value is determined by the maximum Vcc-min value required for reliable operation by any device on a given chip. Furthermore, non-determinism in the manufacturing process leads to large variations in threshold voltage between transistors on the same chip [3]. This process induced variability, that is expected to worsen with smaller feature size, requires larger voltage noise margins and prevents the lowering of Vcc-min. These developments point to a future with diminishing returns from dynamic voltage scaling and performance severely constrained by power.

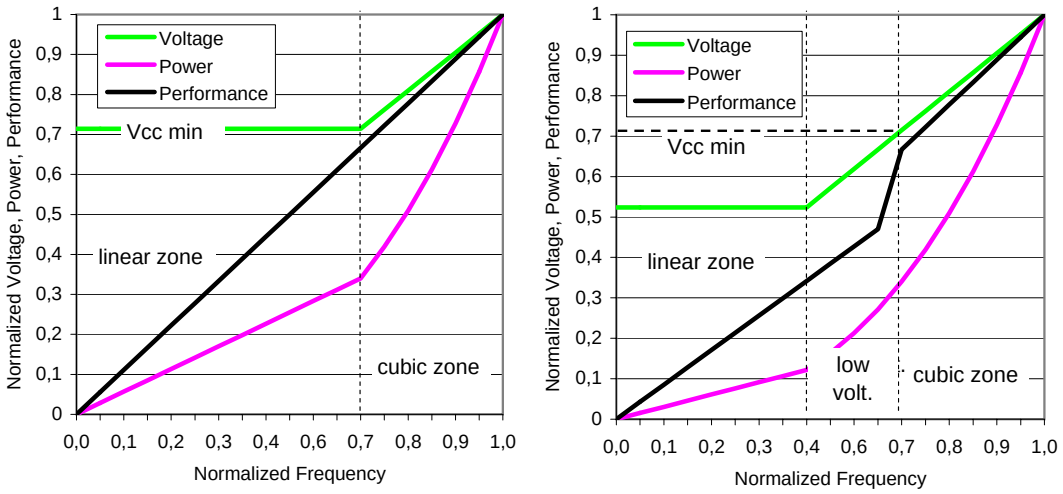
The implications of dynamic voltage scaling on dynamic power (P_{dyn}=CV^2F) and performance are illustrated pictorially in Fig. 1.a[1]. The figure highlights the significance of the Vcc-min that divides the graph into two regions of power reduction: the cubic and the linear. The trend that this paper is trying to reverse is the shrinking of the region with cubic power reduction with each technology node.

A. Operation Below Vcc-min

One recent work, aimed to retain the benefits of dynamic voltage scaling for future technology nodes, proposed to allow operation with supply voltage below Vcc-min [22]. Obviously this can enable cubic power reductions over a larger region but this comes at a price: transistors become unreliable when operating below Vcc-min. It has been shown [12] that the probability of cell failure is growing exponentially with voltage decrease and, depending on the voltage and cache size, can be prevalent with 100s or even 1000s of faulty cells in an array. To avoid compromising correctness, the unreliable devices need to be identified a priori and not accessed when operating below Vcc-min[2]. Therefore, at low voltage operation [22] reliability is intentionally compromised in a prescribed manner. This enables trading-off performance, since fewer number of resources or caches with smaller capacity are available to use, for energy reduction.

[1]This example assumes linear relation of performance to clock frequency for illustration purposes

[2]Note that these devices are reliable when operating at Vcc-min or higher voltage



(a) Voltage scaling

(b) Voltage scaling below Vcc min

Figure 1. Voltage Scaling vs Power and Performance

The implications of operation below Vcc-min are highlighted in Fig. 1.b. Three regions of behavior can be observed. The cubic region has the same range as before. However, the linear region shrunk and a new region, the *low voltage*, appears between the cubic and linear. The cubic and linear regions exhibit similar behavior as in Fig. 1.a. The performance in these two regions can be lower than the corresponding performance in Fig. 1.a depending on whether the method used to provide reliable operation below Vcc-min affects the performance when operating in the other two regions. The main differences are in the low voltage region. The power reductions are now cubic but the performance degradation is sub-linear, instead of linear as in Fig. 1.a. The performance degradation gets worse as voltage is further reduced since increasingly more devices become unreliable.

The goal of this paper is to mitigate the performance loss due to reduced cache capacity when operating in the low-voltage region of Fig. 1.b. The paper focuses on caches since they are critical to performance, occupy most of the chip area and consume a large fraction of the power.

A simple method is proposed that disables faulty cache blocks [19] at low voltage. The rationale for such method is based on observations regarding the distributions of faults in an array according to probability theory. The key insight from the probability analysis is that as the number of random uniformly distributed faulty cells in an array increases the faults increasingly occur in already faulty blocks. This probabilistic analysis is also shown to be useful for assessing the potential of other cache configurations and techniques to mitigate the performance degradation due to faults.

For the configuration used in this paper, block disabling is shown to have better performance than a previously proposed scheme aimed to facilitate low voltage operation but based on word-disabling [22]. Furthermore, block-disabling is simple to implement, requires 1 bit per block instead of 1 bit per word and does not need extra alignment logic. Also, it does not degrade performance at or above Vcc-min operation. Finally, it is shown that a victim-cache enables higher and more deterministic performance for a block-disabled cache.

The remainder of the paper is organized as follows: Section II reviews word-disabling a previously proposed cache disabling scheme for low voltage operation. Section III presents block-disabling. Next in section IV we present a fault-distribution analysis based on probability theory. The experimental framework and results are presented in Sections V and VI respectively. Section VII discusses related work and, finally, Section VIII concludes the paper.

II. Cache Operation Below Vcc-min with Word-Disable

This section reviews the word-disable scheme proposed in [22] to enable correct cache operation below Vcc-min. This is the main mechanism that we compare our solution against.

For the remaining paper low-voltage refers to operation below Vcc-min and high-voltage to operation at or above Vcc-min.

The word-disable scheme tracks low-voltage faults at word granularity. It maintains a fault mask per block in the tag array. The fault mask contains as many bits

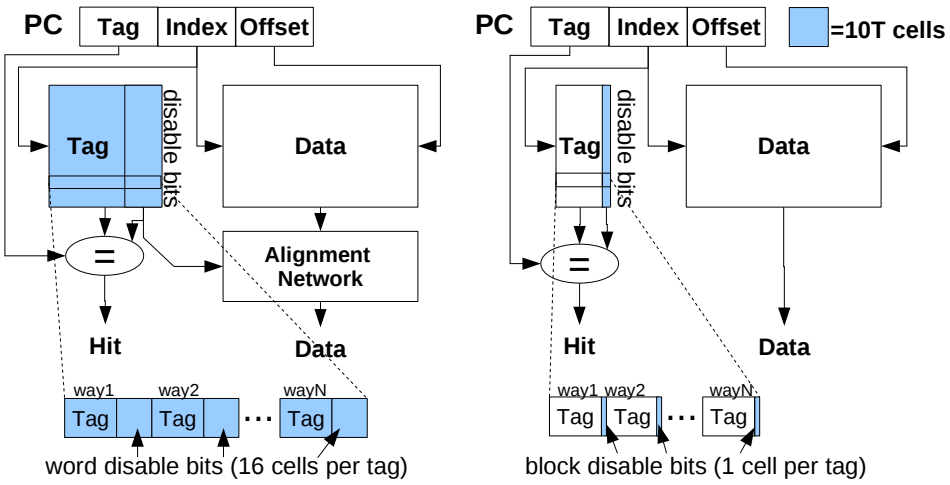


Figure 2. *Left: Word disabling mechanism. Right: Block disabling mechanism*

as words in a block and each bit indicates whether its corresponding word contains a fault. The fault mask is initialized during the boot sequence of a processor using low voltage memory tests.

During high voltage operation the fault-mask is ignored and cache operates normally. When operating at low-voltage a pair of physical blocks in a set is merged into one logical block. This divides by two the cache capacity and associativity[3]. The first physical block is responsible to provide only the first half of the logical block while the other half is provided by the second block of the pair. This means that up to n/2 faulty words can be tolerated for a block with n words. If a block has more than n/2 faulty words it turns the whole cache defective and not suitable for low-voltage operation. Section IV analyzes how fault distribution affects the likelihood of a word-disable cache to be classified as faulty.

To read out in aligned form the valid half block contained in each physical block, the data in each block need to pass through a shift-multiplexer network controlled by each block's fault-mask. This alignment network increases the access latency of the cache in low-voltage mode and may even increase the cache latency during high-voltage operation.

For this work when using word-disabling the subblock size is 8 words and, therefore, no more than 4 faulty words can be tolerated in each subblock. The paper by [22] shows that for an 8 word subblock size the alignment network increases cache latency by 1 cycle.

Word-disabling is only applied to the data array of a cache. The tag array where the fault-mask is stored uses 10-transistor Schmitt trigger cells (10T) which are known [12] to be robust even at low-voltage. These transistors have roughly twice the area overhead of a regular 6-transistor (6T) cell.

The operation and organization of word-disable is illustrated in Fig. 2.

In [22] the bit-fix mechanism is also proposed. This scheme repairs faults at the granularity of bit-pairs. The main downside of the bit-fix mechanism is that it requires a complex merging that increases the cache access latency. The performance analysis in [22], for a specific processor configuration, revealed that word-disabling is more suitable, as compared to bit-fix, for a first-level cache. The fastest access latency of word-disable makes up for its lower capacity as compared to the slower but with larger capacity bit-fix scheme.

For the rest of the paper we focus on improving the performance of the word-disabling scheme that has overall same or better performance and is simpler to implement as compared to bit-fix.

III. Block-Disabling and Victim caching

The goal of this paper is to minimize the performance loss due to the reduced cache capacity when operating below Vcc-min. A simple method is proposed: disable faulty blocks at low voltage. A block is disabled when there is a faulty bit in either or both the tag or data of a block.

Block disabling requires only an extra bit per block in the tag array to indicate whether a block is disabled during low voltage operation. The disabled bit can be set during low voltage tests at boot time. The disabled bit is a 10T cell [12] that is resilient against low-voltage induced faults. Block-disabling for a cache with 64 bytes/block and

[3]This scheme is only applicable to associative caches.

Scheme	Tag ((tag_size+1)* num_blocks)	Disable bits (mechanism_cost* num_blocks)	Victim $ cost (tag+num_entries* block_size(bits))	Alignment network	Total transistors
Baseline	25*512*6T	N/A	N/A	No	76800
Baseline+V$	25*512*6T	N/A	(31+16*512)*6T	No	126138
Word Disabling	25*512*10T	16*512*10T	N/A	Yes	209920
Block Disabling	25*512*6T	1*512*10T	N/A	No	81920
Block Disabling+V$ 10T	25*512*6T	1*512*10T	(31+16*512)*10T	No	164150
Block Disabling+V$ 6T	25*512*6T	1*512*10T	(31+16*512)*6T + 16*10T(dis. bits)	No	131418

Table I. Overhead Comparison of the different Disabling schemes

3 bytes/tag results in an overall cache increase of 0.4%. For this configuration, this overhead is smaller by more than an order of magnitude than what is required by word-disabling (0.4% vs 10%). Furthermore, block-disabling does not require an alignment network that increases the cache access latency of the word-disable scheme.

As illustrated in Fig. 2 support for block-disabling requires minimal changes and should not have an impact on access latency of the cache either at high or low voltage. At high-voltage the disable-bit is ignored and a cache operates as normally. At low-voltage, however, a disabled block is never allocated for a fill on a miss. Therefore, block disabling results in a cache with variable associativity per set that is determined by the number and distribution of faults in the cache. This is in contrast to word-disabling where either each set has half the associativity and size or the whole cache is defective.

Block-disabling has been proposed by [15], [19] to increase processor yield and is used by modern processors [13] to continue operation in the presence of permanent-errors. In this work we consider it for low-voltage operation. The main contribution of the paper is not the notion of block-disabling but the analysis that explains why it can be an attractive option to consider. As demonstrated in the next Section IV, depending on the probability of failure and fault distribution, the number of valid blocks at low-voltage can be more than half when using block-disabling and, therefore, the overall capacity can be higher than word-disabling. The higher capacity of block-disabling in combination with its lower cost overhead and simplicity make it an attractive option to consider for low-voltage operation.

A. Victim-Caching

Although, block-disabling can have higher overall capacity some of its sets may have lower associativity than word-disable. This can translate to lower performance if the low-associativity sets happen to be accessed frequently with locality that is better captured with more ways. This possible shortcoming of block-disabling can be circumvented by using a victim cache [10]. In particular, a victim-cache may be more useful for a block-disabled cache, as compared to a word-disabled cache, because block-disabling's higher capacity and variable associativity may cause most replacements to come from few sets. This can translate to more temporal locality in the misses that is captured by the victim cache. With word-disable the overall cache capacity is smaller and all sets have same associativity. As a result, misses are more uniform from many sets and, therefore, there is less opportunity for hits in the victim-cache.

For a cache that is already backed-up by a victim-cache with 6T cells the overhead for using it in low-voltage mode is either (a) to double the victim cache area by implementing it with 10T cells to provide reliably the same associativity both at high and low voltage, or (b) add one 10T cell per block in the victim cache to indicate if it is valid for low-voltage use. The first option has higher performance potential but also higher cost. Its cost will be about half of word-disabling if the victim cache data storage in terms of bits equals the number of words in a cache.[4] Recall that with word-disable one bit/word indicates if a word is faulty. The second option has minimal overhead but may have lower performance due to reduced associativity at low-voltage.

For a cache without victim buffer the options are: a 10T cell victim cache or a 6T cell victim cache with one 10T cell per block used for disabling. When the victim data bits are as many as the words in the cache, the 10T cell victim cache has less but roughly the same overhead as word-disabling. The 6T cell victim-cache incurs about half the overhead of word-disabling.

Table I summarizes the cell transistor overhead of the different victim cache options assuming a 32KB 8-way 64B/block cache with a 24 bit tag, 6 bit index, 6 bit offset and 1 valid bit. It is evident that in all cases block-disabling has lower overhead. Section VI compares the performance of the different schemes.

[4]For a 32KB cache with 64bytes/block this corresponds to 8K cells or 16 blocks of data.

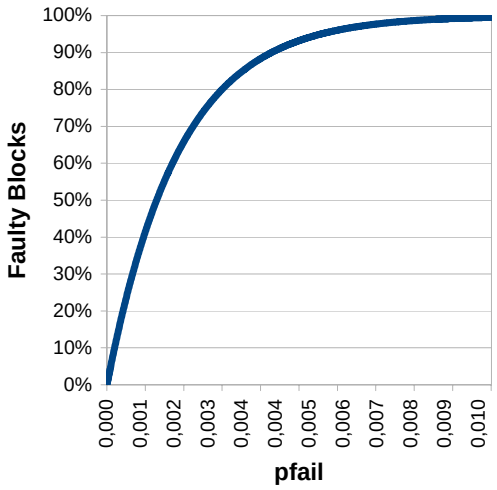


Figure 3. *Fraction of Faulty Blocks as a function of* p_{fail}

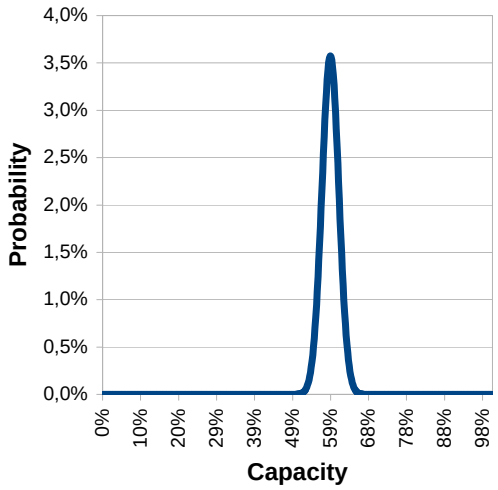


Figure 4. *Probability Distribution of Cache Capacity when* p_{fail}*=0.001 for a 32KB 64B/block cache)*

IV. Analysis of Distribution of Random Fault Cells in an Array

This section performs analysis using probability theory to establish the conditions under which a block-disable cache can have superior performance than word-disable [22]. In particular, we determine the conditions that result in block-disabling to have higher cache capacity than word-disabling at low-voltage. Recall that word-disabling gives up half the associativity and size at low-voltage. We also demonstrate how our analysis methodology can be used to estimate the capacity of other cache configurations and techniques.

We assume that faults occur with uniform random distribution at the granularity of a cell. Random process variation- the main source of faults at low-voltage - vary at such fine granularity [4]. The work that we compare against [22] also makes the same assumptions.

A. Estimating the Capacity of the Block-Disabling Scheme

For a fixed number of faults the problem we are trying to analyze is analogous to selecting at random `n` balls from an urn that contains `dk` balls without replacement,where `d` is the number of unique colors and `k` is the number of balls of each color. The urn represents the cache, the variable `n` the faults, `d` the number of blocks and `k` the number of cells in each block.

One key property of this problem is the mean number of distinct blocks, `u`, that contain at least one faulty cell in a cache with `n` faulty cells. This can be obtained using the following expression [23]:

$$u = d - d\prod_{i=0}^{k-1}(1 - \frac{n}{dk - i}) \quad (1)$$

For instance, for an example cache with 512 blocks, with 64B/block, 24 bit tag and 1 valid bit per block, d=512, k=64*8+24+1=537 and dk=274944 cells. If 1 out of 1000 cells are faulty, there will be 275 faulty cells that, according to Eq. 1, are expected to occur in 213 distinct blocks. The remaining 62 faults will occur in blocks that are already faulty.

When `i` is much smaller than the total number of cells, `dk`, then Eq. 1 can be approximated by the expression[5] which is useful for a fixed probability of failure (p_{fail}):

$$u = d - d(1 - p_{fail})^k \quad (2)$$

Fig. 3 shows using Eq. 2 how the mean fraction of faulty blocks grows as a function of the pfail of faulty cells for a cache with 64B/block and 24 bit tag. The graph clearly illustrates that as the number of faulty cells in an array increases the faults increasingly occur in already faulty blocks. Probably the most important observation is that block-disabling offers more than half cache capacity when p_{fail} is less than 0.0013. Therefore, if low-voltage operation results in p_{fail} less than 0.013 then block-disabling offers larger cache capacity than word-disable.

This is an encouraging result for block-disabling but is only true for the mean number of faulty blocks for a given probability of failure. What is also important to know is the

[5]We found this to be an accurate approximation for all cache configurations we examined.

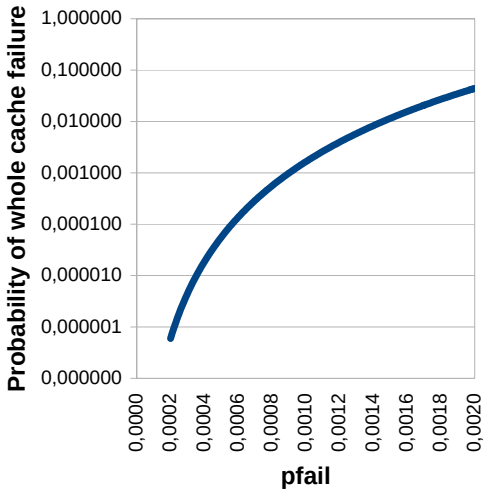


Figure 5. *Probability of a whole cache failure vs Cell Pfail for word-disabling (32KB cache, 64B/block)*

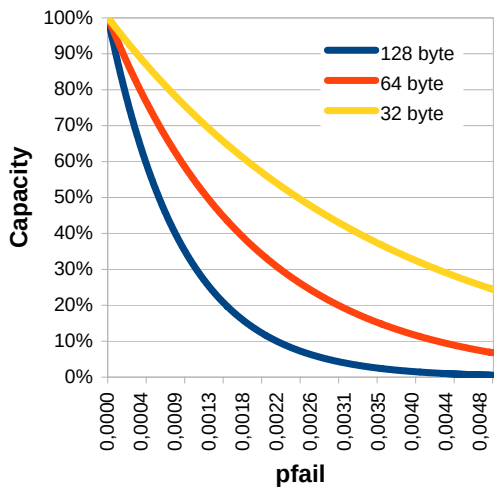


Figure 6. *Capacity for Different Configurations of Block Size*

probability distribution of the cache capacity. In particular, what is the probability for a cache to have x capacity, i.e. x fault free blocks, for a given p_{fail}. This can be obtained using binomial probability:

$$\binom{d}{x}(p_{bf})^x(1-p_{bf})^{d-x} \quad (3)$$

where p_{bf} is the probability of a block with k bits to contain a fault that is given by $p_{bf} = 1-(1-p_{fail})^k$

Eq. 3 produces the distribution shown in Fig. 4 for our running example cache configuration when pfail is 0.001. This is a normal distribution with mean at 58% and standard deviation of 2.02. Consequently, there is a 99.9% probability for a block-disable cache to have more than 50% capacity. Therefore, block-disabling will virtually always have higher capacity than word-disabling.

The results in Fig. 3 show that word-disabling may be a better option for low-voltage when p_{fail} is higher than 0.0013. However, with higher p_{fail} word-disabling has increasingly higher probability of having 8-word subblocks containing more than four faulty words. This renders a whole cache unfit for low-voltage operation [22]. To calculate the probability of whole cache failure (p_{wcf}) of the word-disabling scheme we use the following:

$$1-(p_{hbf})^{d\times 2} \quad (4)$$

where p_{hbf} is the probability that a half-block will contain more than 4 faulty words and can be calculated using the following:

$$p_{hbf} = \sum_{i=a/2+1}^{a/2}\binom{a/2}{i}(p_{wf})^i(1-p_{wf})^{a/2-i}, \quad (5)$$

where $p_{wf} = 1-(1-p_{fail})^{32}$ is the probability that a word will be faulty (assuming 32 bit words), and a is the number of words in a block. Note that the above equations do not take the tag bits into account since for the word-disabling scheme, the tag bits are assumed to be built using reliable 10T cells and are therefore always fault free.

Fig. 5 shows the the probability of a whole cache failure for a 32KB 64B/block word-disable cache. It can be observed that when p_{fail} is 0.001 the probability is small, almost 1 in 1000 caches are unfit. But, when p_{fail} grows to 0.0015 the cache failure probability increases by a factor of 10 to 1 out of 100.

Overall, the analysis reveals that block-disabling may be a useful alternative to consider for low-voltage operation. In Section VI we investigate how the larger capacity of block disabling translates to actual performance improvement.

Before reporting the performance results we highlight next some other uses of the probability analysis for assessing the potential of other cache configurations and techniques to minimize the performance degradation due to faults.

B. Effect of Different Block Size

To evaluate the sensitivity of block-disabling to the block size we evaluated Eq. 2 for three block sizes: 32B,64B and 128B. Fig. 6 shows the corresponding capacity for the three block sizes as a function of p_{fail}. In each configuration we kept the cache size and associativity constant and altered the block size and the number of sets. From the figure it is evident that smaller block size means higher capacity. In the block-disabling scheme, a

single faulty cell renders the whole block inoperable. By decreasing the block size, the effect on cache capacity of single faulty cell decreases and, therefore, overall capacity increases. The capacity increase comes at the expense of lower spatial locality. This can be mitigated, however, with the use of prefetching. Examining the effect of cache configuration parameters, such as the block size, along with supporting techniques, such as prefetching, on reliability represents an interesting direction for future work.

C. Evaluating an Incremental Word-Disable Mechanism

In this subsection we apply our analysis methodology to a variant of the word-disabling scheme named incremental word-disabling. This mechanism allows pairs of blocks that are fault free to operate at full capacity even at low voltage operation. Additionally, pairs of blocks that contain a half-block with more than 4 faulty words are disabled so that the whole chip does not have to be discarded. Block pairs that contain faults, but do not have to be disabled, operate at half capacity as in the original word-disabling scheme. To estimate the capacity of this scheme we use the following:

$$p_{bpff} + (1 - p_{bpff} - p_{bpd})/2 \tag{6}$$

where p_{bpff} represents the probability that a block-pair is fault-free, and p_{bpd} the probability that a block-pair is disabled. Therefore, expression $1 - p_{bpff} - p_{bpd}$ corresponds to the fraction of block pairs that operate at half capacity. To calculate p_{bpff} we use the following expression: $p_{bpff} = (1 - p_{fail})^{k \times 2}$ where k is the number of data bits in a block. p_{bpd} is calculated from $1-(1-p_{hbf})^4$ where p_{hbf} represents the probability that a half-block will contain more that 4 faulty blocks and is obtained using Eq. 5.

Fig. 7 shows that the incremental word-disabling mechanism for a 32KB 64B/block cache performs well. At low probabilities of failure, the number of fault free block-pairs is high with capacity over 50%. As the number of faulty cells increases, more block-pairs will contain faults and capacity begins to saturate at 50%. When we move to higher probabilities of failure, more block-pairs are disabled which decreases capacity to a value below 50%. The analysis shows that the incremental word-disabling scheme degrades more gracefully than the word-disabling while completely avoiding the whole cache failure scenario. We do note however, that this scheme may not be easy to implement. A block pair can be in three states: fault-free, disabled, half capacity. A different access path is required for the fault-free and half capacity blocks since the later requires the block to pass through the word-disabling shifting network. This can increase the cache access time non-determinism and may complicate implementation.

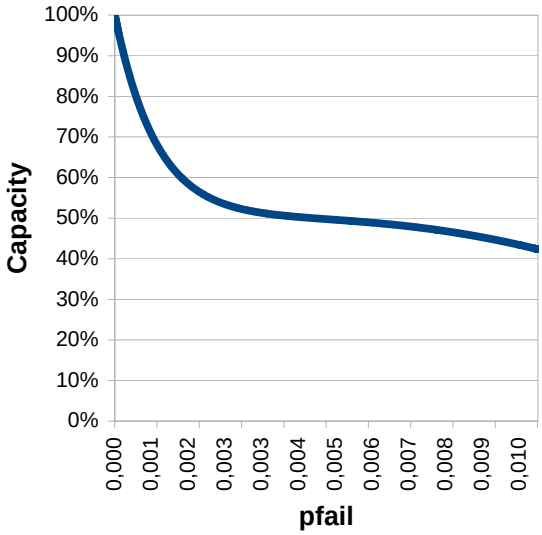


Figure 7. *Capacity as a function of p_{fail} for the incremental word-disabling scheme*

V. Evaluation Framework

For our experiments we used the validated cycle accurate simulator *sim-alpha* [6] that models a high-performance out of order superscalar processor. The simulator is extended to support cache block disabling. Table II contains the processor parameters that are constant for all runs while Table III displays configuration specific parameters. For all configurations, we run all 26 SPEC CPU 2000 benchmarks for 100M committed instructions using reference inputs. The simulation regions they were selected using an in-house SimPoint-like tool.

The performance of block-disabling is evaluated with many simulation runs since faults can occur randomly at any cell. In particular, block-disabling configurations are evaluated with 50 random fault map pairs. Each pair consists of two maps one for the instruction cache and another for the data cache. The cell probability failure is assumed to be 0.001 the same as in [22]. The analysis in Section IV shows that using a higher p_{fail} will result in many whole cache failures. Faults can occur either in the tag array or data array of a cache. For a given fault map any block that has at least one faulty cell is marked disabled for low-voltage operation. In section VI we report, for each block-disabling configuration and benchmark, the average and minimum values for 50 runs.

Block-disabling does not affect the cache latency whereas word-disabling adds one cycle both in high and low voltage. Additionally, for the low-voltage configura-

Parameter description	Setting
Pipeline depth	15 stages
Line Predictor	6.5 KB
RAS	16 entries
Branch Predictor	8 KB gshare (15 bits history)
Fetch/Decode/Issue/Commit	up to 4/4/6/4 instr. per cycle
Issue Queue	40 INT entries, 20 FP entries
Functional Units	4 INT ALUs, 4 INT mult/div, 1 FP ALUs, 1 FP mult/div
Reorder buffer	128 entries
L2 unified cache	2 MB, 8-way, 64 B blocks, 20-cycle hit latency, LRU

Table II. Parameters that are constant for all configurations

Operation Mode	Configuration	Frequency	Memory latency (cycles)	L1(I+D) size, associativity, block	L1(I+D) latency (cycles)	Victim$ entries/latency
High Voltage	Baseline	3GHz	255	32 KB, 8-way, 64 B	3	N/A
	Word disabling				4	N/A
	Word disabling+V$				4	16/1
	Block disabling				3	N/A
	Block disabling+V$				3	16/1
Low Voltage	Baseline	600MHz	51	32 KB, 8-way, 64 B	3	N/A
	Word disabling			16 KB, 4-way, 64 B	4	N/A
	Word disabling+V$			16 KB, 4-way, 64 B	4	16/1
	Block disabling			32 KB, 8-way, 64 B	3	N/A
	Block disabling+V$ 10T			32 KB, 8-way, 64 B	3	16/1
	Block disabling+V$ 6T			32 KB, 8-way, 64 B	3	16/1

Table III. Configuration dependent parameters

tions of word-disabling we reduced IL1 and DL1 cache associativity and size in half.

In section III we describe two possible implementations for block disabling with victim cache. The first implementation assumes that the victim cache is built using 10T cells so that it can operate reliably and at full capacity at low voltage. When evaluating this scheme we use a 16-entry victim cache. The second implementation assumes that the victim cache is built using 6T cells and has an additional 10T cell per block that is used to disable unreliable blocks during low voltage operation. When evaluating this scheme we assume that half of the victim cache entries will contain a fault and so we effectively evaluate the performance with an 8-entry victim cache. This is a conservative assumption since analysis with p_{fail} of 0.001 reveals that the mean number of faulty victim cache blocks is 6.5.

VI. Experimental Results

This section presents performance results for our block-disabling scheme and compares it against word-disabling as proposed by Wilkerson *et al.* [22]. We present separate results for low and high voltage operation.

A. Low-voltage Operation

Fig. 8 shows normalized performance at low-voltage for each benchmark with respect to the baseline without victim cache. Word disabling–the first bar in each group–matches earlier reported results [22] and suffers an average of 11.2% performance loss. The proposed block-disabling scheme reduces the average penalty of low-voltage operation to 8.3%. Considering that block-disabling is simpler and less costly, these results make it an attractive option for low-voltage operation. The improvements of block-disabling is due to its higher capacity and faster access time.

However, when considering the minimum performance of block-disabling it performs worse than word-disabling for the benchmarks `mesa`, `wupwise`, `gap`, `gzip`, and `perlbmk`. This is due to the variable associativity of block-disabling which may result in some frequently accessed sets having less valid ways than word-disabling.

To remedy this non-determinism of block-disabling, we consider block-disabling together with a victim cache with 10T cells. Such a configuration has still significantly lower overhead than block disabling. As shown in Fig. 8 block-disabling with a victim cache has consistently higher average performance than word-disabling and block-disabling without victim cache. Average performance penalty is down to 5.3%. This represents an average 6.6% improvement over word-disabling. Crafty improves the most getting 29% better as compared to word-disabling. Moreover, the minimum performance for block-disabling with a victim cache is virtually always higher than for word-disabling. This happens because the victim cache acts as a fail-safe mechanism for the few sets

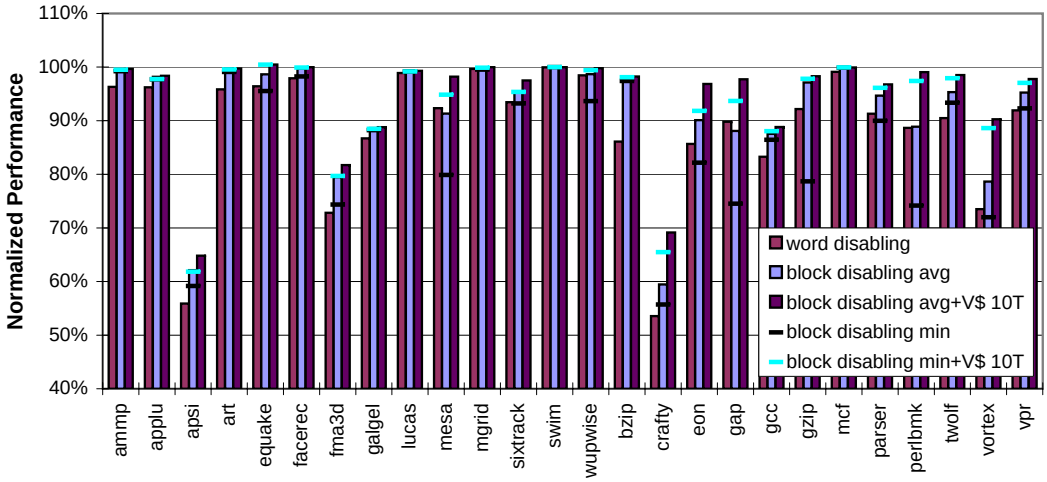


Figure 8. *Below Vcc-min mode results normalized to baseline without victim cache*

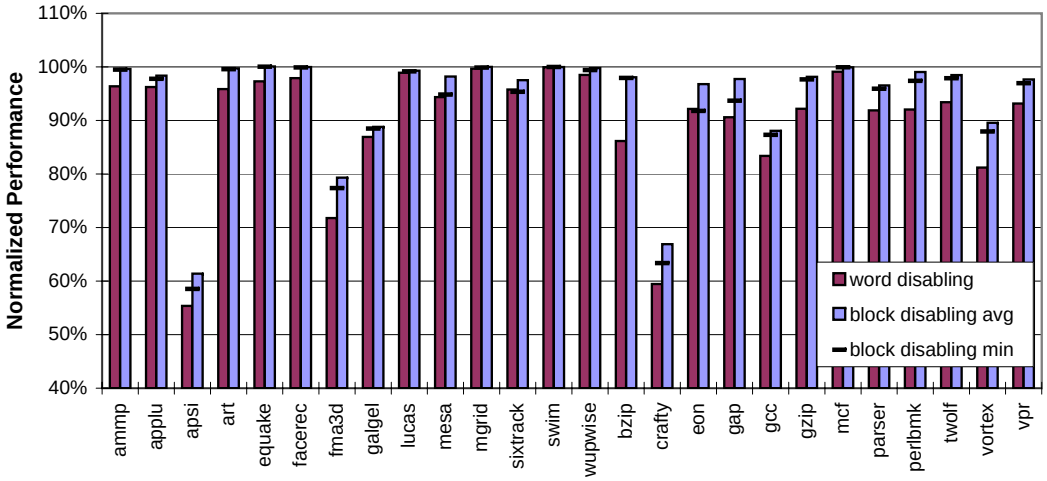


Figure 9. *Below Vcc-min mode results normalized to baseline with victim cache (10T cells)*

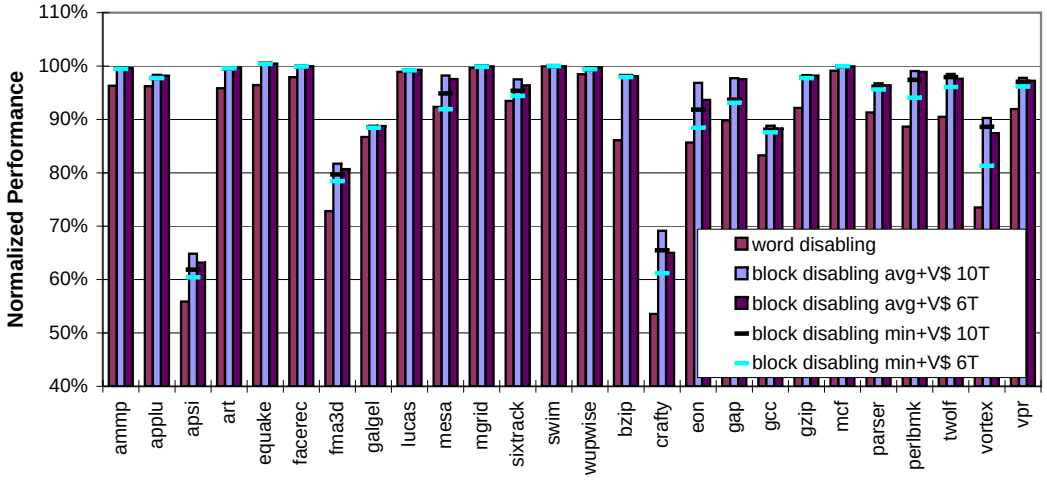


Figure 10. *16-way victim cache: 10T vs 6T*

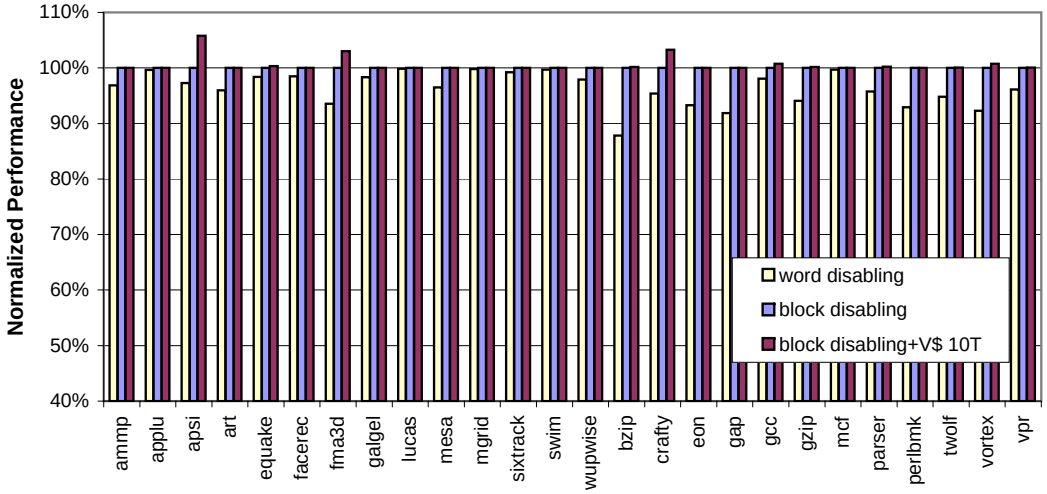


Figure 11. *High voltage mode results normalized to baseline without victim cache*

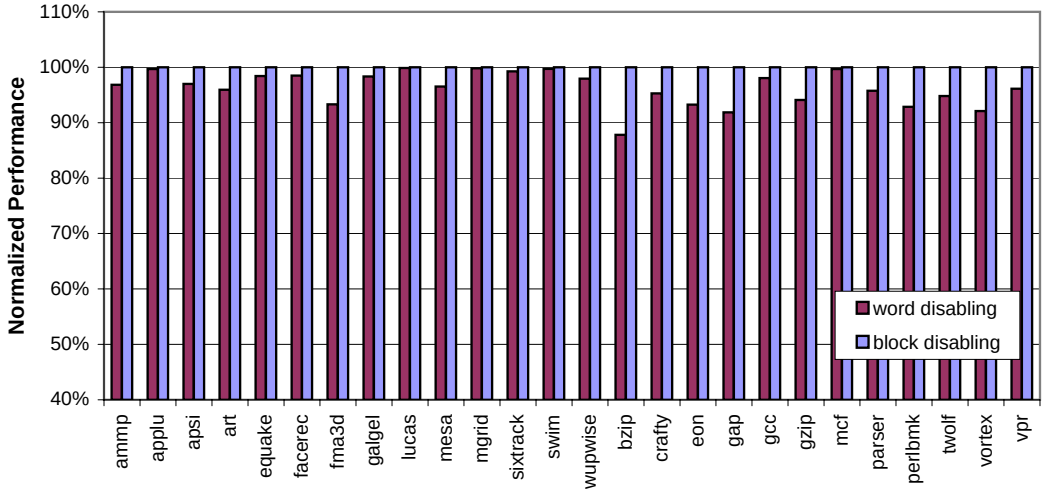


Figure 12. *High voltage mode results normalized to baseline with victim cache*

in the cache that have few valid blocks.

Figure 9 reports normalized performance assuming that all configurations including the baseline are with a 10T cell victim-cache. The average word-disabling degradation is 10% whereas block-disabling reduces this penalty down to 5.8%. The minimum degradation of block-disabling is consistently same or smaller than word-disabling, underlying again the usefulness of a victim cache for a block-disabled cache. What is more, for most benchmarks the minimum values are very close to the average values indicating that the combined use of block-disabling and victim caches produces more predictable performance.

Figure 10 highlights the influence of using a victim cache implemented using 6T vs 10T cells. Both configurations have lower overhead than word-disabling but 6T cells is less expensive. Recall from Section V that this victim cache is assumed to have only 8 valid entries during low voltage operation. Although a few benchmarks show a pronounced decrease of performance with 6T cell victim cache, the average and minimum performance remains better as compared to word-disabling.

B. High-voltage Operation

Having demonstrated the performance benefit of block-disabling at low-voltage, we now consider the effect of disabling during normal operation, i.e. at or above Vcc-min. In high-voltage reliable operation of the entire cache is guaranteed.

Figure 11 shows the normalized performance for word-disabling, block-disabling and block-disabling with a victim cache when operating at high voltage. All results are normalized to a baseline without victim cache. The block-disabling schemes clearly outperform word-disabling. This

is because word-disabling has a longer cache latency in high-voltage due to its alignment network.

In contrast, block disabling adds no overhead when operating at high voltage and thus achieves same performance as the baseline. Adding a victim cache, which is needed for robust low-voltage performance, provides small improvement for `apsi`, `fma3d` and `crafty`.

Finally, Figure 12 compares the performance of block disabling and word disabling when both configurations include victim caching. Normalization is done with respect to a baseline with a victim cache. Here too, the extra latency cycle required by word disabling makes its performance degrade, whereas block-disabling performance is same as the baseline.

VII. Related Work

Sohi [19] studied the performance impact of cache organization with disabled portions, such as ways and sets. The goal of that work was to improve yield without noticeable performance degradation. One interesting observation in that work is that the impact of faults in equal size caches is less pronounced for more associative structures. This observation motivated us to consider a victim cache as a backup to block-disable caches.

Related research was performed by Pour and Hill [17] to study the performance impact of manufacturing faults in caches using a more analytical approach. Lee *et al.* [14] also explored various fault masking strategies for permanent manufacturing faults in caches. The authors measure performance degradation by disabling cache lines, sets, ways, ports or even the complete cache. In this work, we present a scheme to minimize the performance loss when experiencing many faults due to operation below Vcc-min.

Agarwal *et al.* [1] developed a fault-tolerant cache mechanism that dynamically scales down the cache size by replacing faulty cells by correct cells. When faults are detected, the column MUX is forced to select another non-faulty block in the same row.

Victim caching was originally proposed by Jouppi [10] as an approach to keep recently evicted blocks from the main cache in a small fully associative structure. Several variations on the original idea have been developed including selective victim caching [20] and time keeping memory systems [8].

Qureshi *et al.* [18] presented a variable way cache which exploits the non-uniformity of accesses over the different ways to maximize performance. In a sense, one of our cache schemes also benefits from demand-based associativity driven by the amount of unreliable cache blocks, and recovering cache performance by adding a simple victim cache.

In the context of reliability, victim cache-like schemes have been used before. Das *et al.* [5] propose a small fully associative array to keep duplicated data for slow to access, due to process variation, cache words. Zhang [24] proposes replica victim caching to raise the level of error correction to multi-bit errors by storing a limited number of replicas for frequently used blocks in places occupied by dead blocks otherwise.

Another extension to typical error detecting/correcting codes (EDC/ECC) applied to memory systems has been presented by Kim *et al.* [11] to enhance yield. Their two-dimensional error coding schemes can detect and recover from large-scale multi-bit errors while increasing the storage overhead. Yet, this area overhead for both ECC and spares becomes very inefficient when the number of faults becomes large as is the case below Vcc-min.

Kulkarni *et al.* [12] presented a reliable 10 transistor Schmitt trigger cell suitable for sub-threshold operation. They also report a failure rate analysis at low voltage operation.

Based on dynamic voltage scaling Ernst *et al.* [7] proposed 'Razor' as an approach where voltage is tuned while measuring the rate of circuit timing errors. As a result their design eliminates the need for worst case voltage margins, and therefore high energy savings can be achieved by further scaling down the operation voltage.

VIII. Conclusions

Operating below Vcc-min has been proposed to extend the power reduction provided by dynamic voltage scaling for future technology nodes. In this paper, we recommend a simple to implement block-based technique to minimize the performance degradation due to disabling unreliable parts in caches when operating below Vcc-min. The scheme builds on a detailed probability analysis of random fault distribution, that demonstrates that as the number of faults increases is more likely for faults to occur in already faulty blocks. The probability analysis is shown to be useful to assess the potential of other cache configuration and techniques.

Our experiments demonstrate that when pfail is 0.001 the block-disabling scheme outperforms a previously proposed word-disabling on average by 6.6%. Furthermore, block-disabling has much lower overhead, is simple to implement and does not degrade performance at or above Vcc-min operation. In addition, we show that the addition of a small victim cache can be a very useful add-on to block-disabling to get higher and more deterministic performance.

Future work will extend the analytical framework to consider the effects of bit-interleaving and non-uniform fault clustering. It will also examine the potential of block-

disabling for lower level caches and consider its interaction with other mechanisms such as prefetching.

Acknowledgments

This work is partially supported by the University of Cyprus, Ghent University, HiPEAC, and Intel. The authors would like to recognize Constantinos Kourouyiannis for helping with the preliminary studies leading to this work.

References

[1] A. Agarwal, B. C. Paul, H. Mahmoodi, A. Datta, and K. Roy. A process-tolerant cache architecture for improved yield in nanoscale technologies. *IEEE Trans. Very Large Scale Integr. Syst.*, 13(1):27–38, Jan. 2005.

[2] AMD. Family 10th Phenom™II processor. (46878), Feb. 2009.

[3] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter variations and impact on circuits and microarchitecture. In *Proceedings of the 40th Design Automation Conference*, pages 338–342, June 2003.

[4] K. Bowman, D. Brooks, G.-Y. Wei, and C. Wilkerson. Tutorial on design variability: Trends, models and design solutions. In *Tutorial at MICRO*, Nov. 2008.

[5] A. Das, B. Ozisikyilmaz, S. Ozdemir, G. Memik, J. Zambreno, and A. Choudhary. Evaluating the effects of cache redundancy on profit. In *Proceedings of the 41st International Symposium on Microarchitecture*, pages 388–398, Nov. 2008.

[6] R. Desikan, D. Burger, S. Keckler, and T. Austin. Sim-alpha: a validated execution driven Alpha 21264 simulator. Technical Report TR-01-23, CS Dept., University of Texas at Austin, 2001.

[7] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: A low-power pipeline based on circuit-level timing speculation. In *Proceedings of the 36th International Symposium on Microarchitecture*, pages 7–18, Dec. 2003.

[8] Z. Hu, M. Martonosi, and S. Kaxiras. Timekeeping in the memory system: Predicting and optimizing memory behavior. In *Proceedings of the 29th International Symposium on Computer Architecture*, pages 209–220, May 2002.

[9] Intel. First the tick, now the tock: Next generation Intel®Microarchitecture (Nehalem). *White Paper*, 2008.

[10] N. P. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *Proceedings of the 17th International Symposium on Computer Architecture*, pages 364–373, June 1990.

[11] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. C. Hoe. Multi-bit error tolerant caches using two-dimensional error coding. In *Proceedings of the 40th International Symposium on Microarchitecture*, pages 197–209, Dec. 2007.

[12] J. P. Kulkarni, K. Kim, and K. Roy. A 160 mv, fully differential, robust schmitt trigger based sub-threshold sram. In *the International Symposium on Low Power Electronics and Design*, pages 171–176, Aug. 2007.

[13] H. Q. Le, W. J. Starke, J. S. Fields, F. P. O'Connell, D. Q. Nguyen, B. J. Ronchetti, W. Sauer, E. M. Schwarz, and M. T. Vaden. IBM POWER6 microarchitecture. *IBM Journal of Research and Development*, 51(6):639–662, Nov. 2007.

[14] H. Lee, S. Cho, and B. R. Childers. Performance of graceful degradation for cache faults. In *IEEE Computer Society Symposium on VLSI*, pages 409–415, Mar. 2007.

[15] D. A. Patterson, P. Garrison, M. Hill, D. Lioupis, C. Nyberg, T. Sippel, and K. V. Dyke. Architecture of a vlsi instruction cache for a risc. In *ISCA '83: Proceedings of the 10th annual international symposium on Computer architecture*, pages 108–116, 1983.

[16] T. Pering, T. D. Burd, and R. W. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In *the International Symposium on Low Power Electronics and Design*, pages 76–81, June 1998.

[17] A. F. Pour and M. D. Hill. Performance implications of tolerating cache faults. *IEEE Transactions on Computers*, 42(3):257–267, Mar. 1993.

[18] M. K. Qureshi, D. Thompson, and Y. N. Patt. The V-way cache: Demand based associativity via global replacement. In *Proceedings of the 32nd International Symposium on Computer Architecture*, pages 544–555, June 2005.

[19] G. S. Sohi. Cache memory organization to enhance the yield of high performance VLSI processors. *IEEE Transactions on Computers*, 38(4):484–492, Apr. 1989.

[20] D. Stiliadis and A. Varma. Selective victim caching: A method to improve the performance of direct-mapped caches. *IEEE Trans. Comput.*, 46(5):603–610, May 1997.

[21] The International Technology Roadmap for Semiconductors. Edition 2007. Technical report, ITRS, Dec. 2007.

[22] C. Wilkerson, H. Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and S.-L. Lu. Trading off cache capacity for reliability to enable low voltage operation. In *Proceedings of the 35th International Symposium on Computer Architecture*, pages 203–214, June 2008.

[23] S. B. Yao. Approximating block accesses in database organizations. *Communications of the ACM*, 20(4):260–261, Apr. 1977.

[24] W. Zhang. Replica victim caching to improve reliability of in-cache replication. In *Asia-Pacific Computer Systems Architecture Conference*, pages 2–15, Sept. 2004.