Modelling Permanent Fault Impact on Cache Performance

Daniel Sánchez[1], Yiannakis Sazeides[2], Juan L. Aragón[1] and José M. García[1]

Abstract—
The probability of parametric and wear-out failures exacerbates due to the increase of static and dynamic variations. Specifically, caches that dominate the area of modern processors and are built with minimum-sized SRAM cells are very susceptible to faults.

In this paper, we present an analytical model for determining the implications on cache miss-rate due to the use of block-disabling, a mechanism which disables faulty portions of the cache, to mitigate random cell failure. Whereas previous proposals are based on the simulation of different fault-maps, our model avoids them and provides exact measures rather than approximations.

Our evaluation reveals, for the assumptions, programs and cache configuration used in this study, that a relative small number of random fault maps, 100-1000, is sufficient to obtain accurate mean and standard-deviation values for the miss-rate.

I. Introduction

Over the past 50 years, technological advances have enabled continuous miniaturization of circuits and wires. Unfortunately, the scaling of device area has been followed by at least two negative consequences: a slowdown of both voltage scaling and frequency increase, due to slower scaling of leakage current as compared to area scaling [1], [2] and a shift to probabilistic design and less reliable silicon primitives as a result of static [3] and dynamic [4] variations.

A recently published resilience roadmap underlines the magnitude of the problem we are confronted with [5]. Table I shows the predicted p_{fail} (probability of failure) for inverters, latches and SRAM cells due to random dopant fluctuations as a function of technology node. This study clearly shows that, for all types of circuits, the p_{fail} increases at a much faster rate than the area scaling. However, not all circuits are equally vulnerable: SRAM cells, which are usually built with minimum-sized devices, are highly more likely to fail. These alarming trends are leading to forecast that the performance and cost benefits from area scaling will be hindered unless scalable techniques are developed to address the power and reliability challenges. Thus, the development of reliability techniques for future processors which are both scalable and performance-effective is essential, especially for caches that take most of the real-estate in processors and contain numerous SRAM vulnerable cells.

One option is to rely on the error-correction-codes (ECC) already in place to detect and correct soft-errors. However, ECC is not a performance-friendly mechanism for permanent errors because, potentially, every access to a faulty block will incur the ECC repair overhead. Furthermore, ECC soft-error capabilities are reduced when some bits protected by the ECC code are already faulty. Thus, ECC may not be the best option to repair a large number of parametric or wear-out faults in a cache.

TABLE I
Predicted p_{fail} for different types of circuits and technologies.

Technology	Inverter	Latch	SRAM
45nm	≈ 0	≈ 0	6.1e-13
32nm	≈ 0	1.8e-44	7.3e-09
22nm	≈ 0	5.5e-18	1.5e-06
16nm	2.4e-58	5.4e-10	5.5e-05
12nm	1.2e-39	3.6e-07	2.6e-04

Another approach is to disable cache portions such as blocks or words [6], [7], [8] that contain faulty bits upon permanent error detection (at manufacturing or in the field). These disabled blocks are not replaced with a spare[1], which results in a reduction of cache capacity. Block disabling is an attractive option because of its low overhead, e.g., 1 bit per cache block[2], but the reduced cache capacity can degrade performance. Therefore, it is important to determine the performance implications of block disabling to assess its usefulness.

Previous block disabling-based studies (such as [7], [9], [10], [11], [12], [13], [14], [15]) rely on the use of an arbitrary number (small or large) of random fault-maps. Each random fault-map indicates faulty cache cell locations and determines the disabled faulty cache blocks. The fault-maps are used either to obtain the performance degradation of a program through cycle accurate simulation, or to determine the impact on miss-rate of a program's address trace. However, the number of fault maps used in these studies is very small as compared to the number of all possible maps. Therefore, the accuracy of previous work in predicting expected performance has not been established.

Our proposition to address this shortcoming is an analytical model that calculates the Expected Miss Ratio (EMR) for a given address trace of an application, cache configuration and random probability of permanent cell failure. Furthermore, we show

[1]Dept. of Computer Engineering, Univ. of Murcia. e-mail: {dsanchez,jlaragon,jmgarcia}@ditec.um.es
[2]Department of Computer Science, Univ. of Cyprus. e-mail: yanos@cs.ucy.ac.cy

[1]Disabling can be employed after spares have been exhausted.
[2]This logical bit needs to be resilient either through circuit design or extra protection, because if faulty it renders whole cache faulty.

how to obtain the standard deviation for the EMR (SD_MR) which provides an indication for the range of expected degradation of the cache. Finally, we explain how to produce a probability distribution for the EMR for a given number of faulty blocks. All these are accomplished without producing or using fault-maps. This analytical model can be useful for manufacturers to analyze the impact of permanent faults in caches and tune their designs by using an appropriate number of spares or modifying the granularity of disabling techniques.

The model capabilities are demonstrated through an analysis of the trends of the cache miss-rate mean and standard deviation with smaller feature size (and p_{fail}) for L1 data caches.

The remainder of the paper is organized as follows: Section II presents our model to calculate the EMR and SD_MR. In Section III we describe the methodology and the evaluation results. Finally, Section IV resumes the main conclusions of this work.

II. Exact Model for Cache Miss Rate Behavior with Faults

In this section, we present an analytical model that can determine the Expected Miss Ratio (EMR), standard deviation of the Miss Ratio (SD_MR), and a probability distribution of miss-ratios (PD_MR) for a given program address trace, cache configuration and random probability of permanent cell failure. The EMR captures the average cache performance degradation due to random faulty cells. The SD_MR provides indication about the range of this performance degradation, whereas PD_MR reveals its shape (distribution). These characteristics can be used to assess the implications of faults in a cache and compare different cache reliability schemes.

A. Assumptions and Definitions

The model assumes that permanent faulty cells occur randomly (uncorrelated) with probability p_{fail}. This random fault behavior is indicative of faults due to random-dopant-fluctuations [16] and line-edge-roughness [17], two prevalent sources of static variations. The systematic component of process variation manifests itself at a larger scale (e.g., at the granularity of microarchitectural units or whole core) and can be addressed by coarse-grain techniques like body biasing [3]. The random variation occurs at a finer grain and cannot be handled with manufacturing process tuning or coarse grain techniques [18]. The model presented in this work assumes that the systematic failures have been addressed and examines the implications of random faults in memory cells.

A cache configuration is defined by the number of sets (s), ways per set (n), and block size in bits (k). We consider a block containing one or more permanently damaged bits as faulty. In that case, the faulty block is disabled, reducing the capacity of the cache. Faults are assumed to be detected with post-manufacturing and/or boot time tests, ECC, and built-in self tests. The model is suited for policies that induce total priority order in the replacement. In our case, we have focused on a basic LRU policy.

Each program address trace is simulated through a cache simulator to obtain, for a given cache configuration, the vector M. This vector contains $n+1$ elements, an element more than the number of cache ways. M_i corresponds to the total misses when there are only $n-i$ valid ways in each set in the cache. More specifically, M_i equals to sum of all the references which hit in the i least recently used blocks in each set, plus the misses of the fault free cache. For example, M_0 equals to the misses of a fault-free cache, M_n represents the misses of a cache in which every way is entirely faulty, meaning all accesses are misses, and M_1 equals to the misses of the fault-free plus all the hits in the LRU position.

B. EMR and SD_MR

This section shows how the model obtains the EMR and SD_MR given a cell's p_{fail}, cache configuration and the miss vector of an address trace. The model calculates the probability for a cache block failure using the following expression (based on well-known binomial probability):

$$p_{bf} = 1 - (1 - p_{fail})^k \tag{1}$$

Although p_{bf} provides information about the fraction of blocks that are expected to fail in the cache, the impact on the miss ratio is unknown, as it depends on the fault location and the amount of accesses which maps to faulty block locations. However, with the p_{bf} we can obtain the probability distribution pe_i for the number of faulty ways in a set:

$$pe_i = \binom{n}{i} p_{bf}^i (1 - p_{bf})^{n-i} \tag{2}$$

which provides, for every possible value of i $[0...n]$, the probability of having $n-i$ non-faulty ways. This distribution is very useful because it provides insight about how likely it is to lose a given number of ways in a set and, what is more important, it can be used to obtain the expected number of misses. The expectation of a random variable $X = x_0, x1..., x_m$ in which each possible value has probability $p = p_0, p_1, ..., p_m$ is calculated as:

$$E[X] = \sum_{i=0}^{m} x_i \cdot p_i \tag{3}$$

In our case, the random variable X corresponds to the total number of misses for a cache with faults; x_i corresponds to the total misses when there are only $n-i$ valid ways in each set in the cache; and p_i the probability of having i faulty ways in a set. Therefore, we can express the expectation of the number of cache misses with disabled blocks as:

$$E_{misses} = \sum_{i=0}^{n} M_i \cdot pe_i \tag{4}$$

and obtain the expected miss ratio of the cache using:

$$EMR = \frac{E_{misses}}{accesses} \quad (5)$$

This simple formula can be used to obtain the exact EMR without using fault-maps. I.e. it determines the EMR as if all possible fault-maps for a given random p_{fail} had been taken into account. The key insight behind this formula, expressed better in Eq. 2, is that caches have a useful property: for the same number of faulty blocks f in a set, the reduced associativity will be the same $n - f$. I.e., for analyzing block-disabling, what matters is the number of faulty-ways in a set, not which specific ways in the set are faulty. Thus, the complexity of the problem is reduced.

The EMR provides a useful indication about the average case performance for a given p_{fail}. However, we have no information about the variation in the miss ratio. Variation is useful for assessing whether disabled blocks lead to caches with wide variation (less predictable) miss rate.

One way to measure this variation is through the standard deviation of the MR or SD_MR. Unfortunately, the standard deviation cannot be directly obtained for the whole cache. However, given that we already know the probability distribution of faulty blocks in a set, we can calculate variation as:

$$\forall j[0...s], VAR\_E_{misses_j} = \sum_{i=0}^{n} pe_i \cdot (x_{ij} - E_{misses_j})^2 \quad (6)$$

where x_{ij} is the number of misses obtained when having $n - i$ non-faulty ways in the j_{th} set.

Although the total EMR is equal to the sum of individual sets EMR_j:

$$EMR = \sum_{j=1}^{s} EMR_j \quad (7)$$

we cannot combine the variation of each set in the same way. Instead, we compute the deviation for the misses of the whole cache SD_MR by using the root mean square in the form:

$$SD\_MR = \frac{\sqrt{\sum_{j=1}^{s} VAR\_EMR_j}}{accesses} \quad (8)$$

C. MR probability distribution

The SD_MR only provides the range of deviation of the miss ratio. But, what may be more useful to know is a probability distribution of cache misses (PD_MR) within the deviation range.

We propose to build a probability distribution of misses in a stepwise manner. We first calculate the EMR for every possible number of faulty blocks (from 0 to the number of cache blocks), and then we combine this information with the probability of that given number of faulty blocks to occur.

Equation 9, similar to Equation 2, gives the probability of x number of faulty blocks, for a given block probability failure:

$$\binom{s \cdot n}{x} p_{bf}^{x} (1 - p_{bf})^{s \cdot n - x} \quad (9)$$

This equation can be evaluated for different x values to obtain a probability distribution. Then, we need to calculate the EMR for every possible number of faults. So far, this problem has been solved by means of random fault-maps [9].

For a given number of faults, this problem is analogous to selecting at random n balls from an urn that contains dk balls without replacement, where d is the number of unique colours 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 bits in each block. The mean number of distinct blocks, u, that contains at least one faulty cell in a cache with n faulty cells can be approximated with high accuracy [19]:

$$u = d - d(1 - p_{fail})^k \quad (10)$$

This means that we can obtain the PD_MR analytically, without fault-maps, by simply using Equation 10 to convert the number of faulty blocks to p_{fail}. This, gives us the expression:

$$p_{fail_i} = 1 - \sqrt[k]{\frac{s \cdot n - x_i}{s \cdot n}} \quad (11)$$

This way, we can calculate which p_{fail} results in x_i faults in the cache. Then, every p_{fail_i} can be used to calculate the EMR associated to each number of faulty blocks, therefore, generating a probability distribution.

III. Evaluation

A. Methodology

The input to our model is a map of accesses to a cache for every application. To produce these maps we have used an algorithm called all-associativity simulation [20], previously used in [9]. This algorithm has a complexity order of $O(n^2)$. However, in practice, the complexity is much lower because of access locality, which limits the length of searches dramatically. Due to space limitations, we have not discussed this but refer to [20] for details. It is important to note, though, that the algorithm is applied offline and, with a single run per benchmark, is able to produce the data our model needs to evaluate any desired cache configuration.

The all-associativity algorithm takes as input a trace of memory requests which converts into a map of cache accesses for any desired configuration (sets and ways) following a given replacement policy (LRU in our case). This allows us to obtain the number of accesses per way and per set within a single run. The output of the algorithm is a matrix in which each row corresponds to a set and each column to a position

in the LRU sequence. Each value of the matrix indicates the number of accesses to every position in the LRU sequence for every set. This information is very useful for offline analysis given that we can determine the number of misses for a given number of ways w in our cache by simply adding the accesses for the last $n - w$ columns of the matrix.

We can also use this matrix to compute the misses with faulty cells. For this purpose, and according to Equation 5, we need to calculate the number of misses if a given number of ways were disabled in the cache due to permanent faults. First, we accumulate the number of accesses in every position per set. Then, we perform the same operation per set to obtain a vector which indicates the exact number of misses our cache would suffer as a consequence of losing from 0 to w ways.

For our experiments, we have simulated a processor architecture by means of Virtutech Simics [21] and GEMS [22]. Simics is a functional simulator executing a Solaris 10 Unix distribution simulating the UltraSPARC-III ISA. GEMS is a timing simulator which, coupled to Simics, provides detailed results for the memory system. We have performed several modifications to the simulator to extract cache address traces. Then, these traces are used to generate the map of accesses for every possible cache configuration by means of the all-associativity algorithm as explained previously.

We have conducted our experiments by executing different applications from the SPECcpu-2000 [23] (bzip2, gap, gzip, parser, twolf, vpr). Benchmarks are run for 1 billion of cycles. In all cases, the warming up of caches has been taken into account.

The different p_{fails} used for the evaluation of the caches are shown in Section I with the exception of 6.1e-13, which produces virtually no faulty blocks in our experiments. Additionally, we have evaluated p_{fail} of 1e-03 which is considered in many related papers.

B. Random Fault Map Methodology

Before proceeding to determine the EMR using the proposed methodology we determine how well randomly generated fault-maps approximate the expected number of faults obtained using Eq. 1.

In Figure 1 we can see the probability distribution of the number of faulty blocks for different p_{fails} (we have omitted 7.3e-09 and 1.5e-06 because they offer from 0 to 1 and 0 to 4 faulty blocks, respectively) in a 32KB, 2-way associative cache with 558 bits per block[3]. Results show the estimated faulty blocks obtained analytically (analytical line) and by different numbers of faulty maps (from 100 to 10 millions). As it is observed, few faulty maps are not able to capture the exact behaviour of the analytical model. However, when the number of maps increases (1K maps or more), the number of faulty blocks becomes more accurate. Nonetheless, this study cannot conclude how well random maps approximate the expected misses of a cache, since misses directly depend on the location of faults among the different cache sets.

[3]We consider blocks comprised of: 64 bytes for data and 11 bits for its ECC, 25 bits for the tag and 7 bits for its ECC, and 3 control bits for valid, disable and dirty states.

C. EMR and SD_MR for SPEC applications

In this section we show the calculated EMR and SD_MR for several benchmarks and a 2-way 32KB L1 cache with different p_{fails}.

Surprisingly we can see in Figure 2 that a small number of faulty maps, 100-1000, is enough to approximate the EMR and SD_MR provided by the model. The reason for this is the access homogeneity to the different sets of the cache. In other words, for the applications we have evaluated, there are no particular sets that are clearly more accessed than others during the overall execution of the benchmark. This makes the EMR and SD_MR virtually independent from the fault locations and that is the reason why fault maps are able to provide such good estimations.

We establish the cache access homogeneity with a study of the correlation of accesses between all the sets in our cache by calculating the Pearson correlation coefficient. When the Pearson coefficient is close to 0, it means that there is no correlation between variables, whereas when it is close to 1, it means a correlation between them. We have calculated the matrix of correlations of the number of accesses for a 2-way 32KB L1 cache for the evaluated benchmarks. Table II reflects the average value for the Pearson coefficients as well as its standard deviation. As we can see, all coefficients are very close to 1, which means that the accesses among sets are highly correlated.

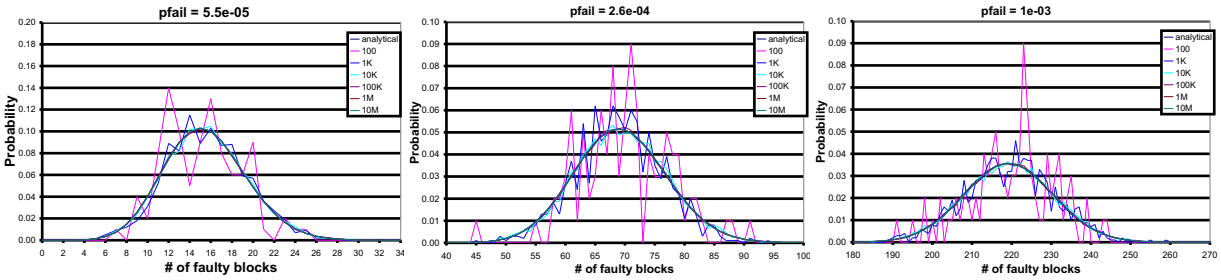
TABLE II
Pearson Coefficient Matrix for each benchmark.

Benchmark	Mean Pearson Coeff.	DEV
bzip2	.993	.007
gap	.9	.086
gzip	.997	.002
parser	.998	.003
twolf	.943	.119
vpr	.995	.006

The key insight from this study is that, because of the high correlation, a small number of random fault maps is sufficient to obtain accurate expected cache behavior with faults. If data accesses among sets are not highly correlated, a few fault maps would not be able to provide an accurate prediction of the expected behaviour with faults.

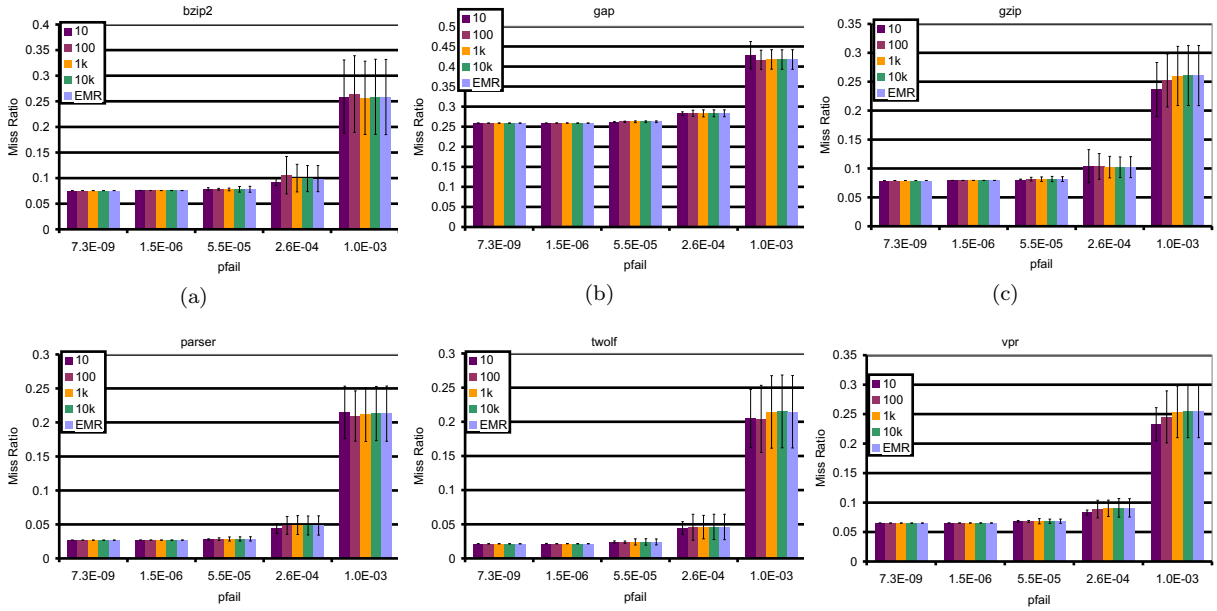
D. PD_MR for SPEC applications

In Section II-C, we have developed a method to calculate a PD_MR for the expected values of the EMR. As explained, we follow a constructive approach, calculating the different p_{fail} from 0 to n faulty blocks. Then, for each of these values we calculate its EMR.



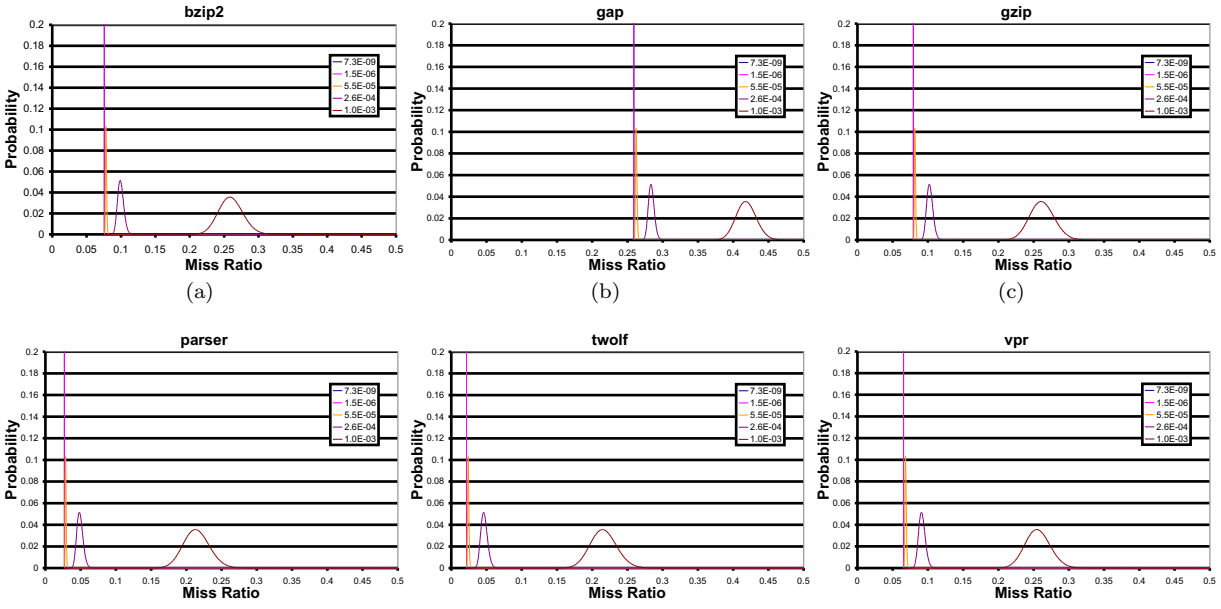
(a) (b) (c)

Fig. 1. Probability distribution of the number of faulty blocks obtained by our model and by random fault-maps in a 2-way associative 32KB cache.



(a) (b) (c) (d) (e) (f)

Fig. 2. EMR and SD_MR for different applications in a 2-way associative 32KB L1 cache with different p_{fails}.



(a) (b) (c) (d) (e) (f)

Fig. 3. PD_MR for different applications and p_{fails} in a 2-way associative 32KB L1 cache.

The probability distribution results in Figure 3 provides valuable information. As a consequence of the increasing number of faulty blocks, the shape of the distribution is wider with higher p_{fails}. Within a single chart we can infer the likelihood of a miss rate in our cache to occur according to the used technology scale. As a conclusion, this study reveals that, in the future, the performance of caches will be more un-predictable due to permanent errors. This model could be used by chip manufacturers to analytically determine what is going to be the expected percentage of chips that should be discarded because of faulty cells.

IV. Conclusions

This paper proposes an analytical model to determine the Expected Miss Ratio (EMR) and its Standard Deviation (SD_MR) for a given application when it is executed in a cache with a random probability of cell failure. This analytical model enables designers to perceive the real impact of faults in caches without the need of executing any experiments with random fault maps. We have also presented an analytical model which provides the probability distribution for the EMR which represents another valuable information for designers about the shape of the miss-rate distribution of faulty cache units for a given process technology.

In the evaluation we show, for the benchmarks and configurations used, that the random fault map methodology provides high accuracy when using 100-1000 maps for an L1 data cache. This is due to the high homogeneity of accesses to the different sets of a cache which makes the EMR and SD_MR virtually independent of the allocation of faults.

Acknowledgements

This work has been jointly supported by the Spanish MEC and European Commission FEDER funds under grants "Consolider Ingenio-2010 CSD2006-00046" and "TIN2009-14475-C04-02". Daniel Sánchez is also supported by a research grant from the Spanish MEC under grant "Consolider Ingenio-2010 CSD2006-00046" and a mobility grant by HiPEAC (FP7 Network of Excellence). The research leading to this paper is supported by the European Commission FP7 project "Energy-conscious 3D Server-on-Chip for Green Cloud Services (Project No:247779 "EuroCloud")".

Bibliography

[1] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23 –29, 1999.

[2] Y. Taur, "CMOS design near to the Limit of Scaling," *IBM Journal of Research and Development*, vol. 46, no. 2/3, pp. 213–222, 2002.

[3] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings of the 40th annual Design Automation Conference*. 2003, pp. 338–342, ACM.

[4] Keith Bowman, James Tschanz, Chris Wilkerson, Shih-Lien Lu, Tanay Karnik, Vivek De, and Shekhar Borkar, "Circuit techniques for dynamic variation tolerance," in *Proceedings of the 46th Annual Design Automation Conference*. 2009, pp. 4–7, ACM.

[5] Sani R. Nassif, Nikil Mehta, and Yu Cao, "A resilience roadmap," in *Design, Automation, and Test in Europe*, 2010, pp. 1011–1016.

[6] David A. Patterson, Phil Garrison, Mark Hill, Dimitris Lioupis, Chris Nyberg, Tim Sippel, and Korbin Van Dyke, "Architecture of a vlsi instruction cache for a risc," in *Proceedings of the 10th Annual International Symposium on Computer Architecture*. 1983, pp. 108–116, ACM.

[7] G. S. Sohi, "Cache memory organization to enhance the yield of high performance vlsi processors," *IEEE Transactions on Computers*, vol. 38, pp. 484–492, April 1989.

[8] C. McNairy and J. Mayfield, "Montecito error protection and mitigation," *HPCRI'05: 1st Workshop on High Performance Computing Reliability Issues, in conjunction with HPCA'05*, 2005.

[9] A.F. Pour and M.D. Hill, "Performance implications of tolerating cache faults," *IEEE Transactions on Computers*, vol. 42, no. 3, pp. 257 –267, 1993.

[10] Philip P. Shirvani and Edward J. McCluskey, "Padded cache: A new fault-tolerance technique for cache memories," in *Proceedings of the 17TH IEEE VLSI Test Symposium*. 1999, pp. 440–, IEEE Computer Society.

[11] Hyunjin Lee, Sangyeun Cho, and B.R. Childers, "Performance of graceful degradation for cache faults," in *IEEE Computer Society Annual Symposium on VLSI*, 2007, pp. 409 –415.

[12] Hyunjin Lee, Sangyeun Cho, and B.R. Childers, "Exploring the interplay of yield, area, and performance in processor caches," in *25th International Conference on Computer Design*, 2007, pp. 216 –223.

[13] T. Ishihara and F. Fallah, "A cache-defect-aware code placement algorithm for improving the performance of processors," in *IEEE/ACM International Conference on Computer-Aided Design*, Nov 2005, pp. 995 – 1001.

[14] D. Roberts, Nam Sung Kim, and T. Mudge, "On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology," in *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools*, 2007, pp. 570 –578.

[15] N. Ladas, Y. Sazeides, and V. Desmet, "Performance-effective operation below vcc-min," in *IEEE International Symposium on Performance Analysis of Systems Software*, 2010, pp. 223 –234.

[16] A.J. Bhavnagarwala, Xinghai Tang, and J.D. Meindl, "The impact of intrinsic device fluctuations on cmos sram cell stability," *Solid-State Circuits, IEEE Journal of*, vol. 36, no. 4, pp. 658 –665, apr 2001.

[17] B. Cheng, S. Roy, G. Roy, F. Adamu-Lema, and A. Asenov, "Impact of intrinsic parameter fluctuations in decanano mosfets on yield and functionality of sram cells," *Solid-State Electronics*, vol. 49, no. 5, pp. 740 – 746, 2005.

[18] Keith Bowman, David Brooks, Gu-Yeon Wei, and Chris Wilkerson, "Tutorial on design variability: Trends, models and design solutions," in *Tutorial at MICRO'08*, Nov. 2008.

[19] S. B. Yao, "Approximating block accesses in database organizations," *ACM Communications*, vol. 20, pp. 260–261, April 1977.

[20] M. D. Hill and A. J. Smith, "Evaluating associativity in cpu caches," *IEEE Transactions on Computers*, vol. 38, pp. 1612–1630, December 1989.

[21] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, B. Werner, and B. Werner, "Simics: A full system simulation platform," *Computer*, vol. 35, no. 2, pp. 50–58, 2002.

[22] Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood, "Multifacet's general execution-driven multiprocessor simulator (gems) toolset," *SIGARCH Computer Architecture News*, vol. 33, no. 4, 2005.

[23] J.L. Henning, "Spec cpu2000: measuring cpu performance in the new millennium," *Computer*, vol. 33, no. 7, pp. 28 –35, July 2000.