Use Them -Don't Waste Them. Recruiting Strong ECC in L1 Caches for Hard Error Recovery Without the Penalty

Michail Mavropoulos, Thodoris Lappas, Georgios Keramidas, Dimitris Nikolos

Use Them - Don't Waste Them. Recruiting Strong ECC in L1 Caches for Hard Error Recovery Without the Penalty

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

Abstract— Operating below nominal voltage levels is a promising direction to enable ultra-low power CMOS-based sytems. This is true due to the cubic relation between the dynamic power consumption and the supply voltage. The main roadblock during aggressive voltage scaling is that the reliability of the circuits and especially of memory structures (SRAM cells) is exponentially decreased. The usage of well-known error correction codes (ECC) can only remedy a part of the problem. ECC have been proposed to protect the lower level caches (e.g., L2 or L3), but ECC cannot be employed in single-cycle first level caches due to their performance sensitivity to the additional latency of ECC. In this work, we propose a methodology to solve this problem by exploring the use of criticality metrics in aggressive superscalar processors. Instead of disabling the faulty frames of a faulty cache (as proposed in related work), we keep these frame "alive" (data are still placed and accessed in these faulty frames) and we use strong ECC to correct the hard multibit errors. The key idea is to direct to faulty cache frames data that can be delayed for one or more cycles without affecting program completion time i.e., instructions considered to not be on the critical path of the program execution. The basic mechanisms for hardening first level faulty caches with strong ECC protection without inflating program executing time are described in this position paper.

Keywords— fault tolerance, ECC, first-level caches, criticality, aggressive superscalar processors.

I. INTRODUCTION

Lowering the supply voltage (Vcc) is one of the most effective methods to reduce power consumption of integrated circuits (IC) [3]. Today's processors have the ability to operate in multiple power nodes. Thus, the Dynamic Voltage Frequency Scaling (DVFS) technique is a key technique to realize ultra low-power consumption and energy efficient systems. Unfortunately, during Vcc reduction, the impact of process, voltage and temperature (PVT) variations become predominant resulting in unreliable ICs. The problem is more pronounced in on-chip memories (i.e., caches) where Vcc scaling leads to an exponential increase in the number of faulty memory cells [7]. The rate of failures in memory cells typically determines the minimum supply voltage, known as Vccmin, at which the whole chip can reliably operate.

It becomes a real challenge to protect cache memories with new fault-tolerance (FT) mechanisms against hard faults appearing when operating below Vccmin (a.k.a. near-threshold execution). A widely used cache FT technique, also employed by many commercial products, is cache frame disabling (CFD). CFD follows the philosophy of graceful degradation and it is an appealing technique due to its simplicity: only an extra bit, called faulty bit, is attached to each cache frame and its value indicates a disabled faulty cache frame. On the other hand, the major disadvantage of CFD is that a significant portion of the cache is wasted (disabled) in high failure rate situations [4]. As Fig. 1 illustrates, more than 50% of cache area is disabled when the SRAM cells probability of failure (pfail) is more than 1e-03. pfail vs. Vcc scaling relation can be found in [4]. To address this inefficiency, the application of CFD in smaller cache frame granularities was proposed in [1]. However, the technique in [1] provides only a partial solution to the problem, since it is not scalable to high pfails.

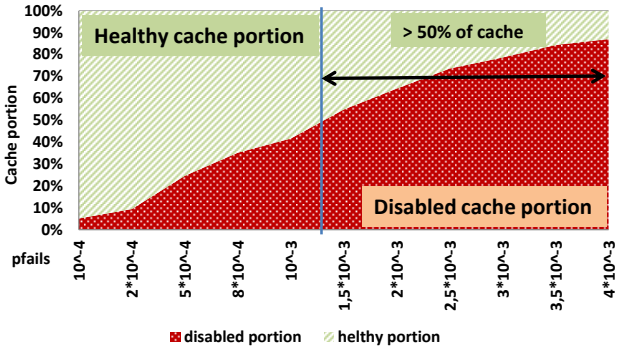

Fig. 1. Disabled cache portion though a wide range of pfails

A natural class of techniques for increasing cache resiliency is to rely on the various error correcting codes (ECC) to detect/correct the hard errors. These techniques are able to increase the effective fault-free cache capacity by correcting one or more bit-cell errors, but they introduce a constant repair time penalty rendering them not suitable for the latency sensitive L1 caches. An interesting study about this issue is presented in [2]. The authors observed that there is a tradeoff between cache capacity and cache hit latency. In particular, in low pfails (low number of faulty cache frames), it is preferable to follow the CFD technique. On the contrary, in high pfails, it is more beneficial to pay the extra latency of ECC in hit access time, otherwise the cost from the additional cache misses will become prohibitively high. However, in the approach of [2], only cache-wide decisions were employed.

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 National Strategic Reference Framework (NSRF) – Research Funding Program: Thales "Holistic". Investing in knowledge society through the European Social Fund

II. Our Proposal

In this position paper, we outline a methodology exploiting the strength of ECC memory protection in first-level caches without increasing unduly the execution time of the running applications. Our mechanism is based on the criticality metric. As demonstrated in [6] not all load instructions are equally important. In fact, many loads have significant tolerance for execution latency. In other words, there is a subset of program's load instructions, called non-critical [6] or not vital loads [5] that can be delayed for one or more cycles without affecting program competition time. Non-critical loads can be considered as load instructions not residing into the program's critical path i.e., the chains of dependant instructions that determines a program's execution time.

In the context of this work, given an ECC protected faulty cache, non-critical loads can be hosted in faulty frames of a faulty cache without hurting system performance. This is because it is safe to pay the overhead of ECC repair time in non-cirtical loads, since these loads can afford some degree of latency tolerance. A high level diagram of the proposed idea is depicted in Fig. 2. The right part of Fig.2 shows an exemplary 2-way first level instruction or data cache in which the red points correspond to errors and consequently to faulty cache frames. In contrast to previous approaches that propose to disable the faulty cache frames, as part of this work, we propose to keep these faulty cache frames "functional" and use ECC to detect and correct the hard errors. As noted, if non-critical data are carefully directed to faulty cache frames, then the program's execution time will remain almost intact.

The left part of Fig. 2 illustrates our approach. The top part shows the modifications required in cache placement logic, whereas the bottom part shows the corresponding modifications in cache hit path. In essence, we leverage the ability to predict whether a load is in the application's critical path using an appropriate criticality predictor. We have implemented three types of criticality predictors explicitly optimized to the needs of the proposed work and we are now in an evaluation phase. The proposed criticality predictors are based on i) the number of dependant instructions of a load instruction, ii) the behavioral of a load in the commit stage, and iii) the location of a load in the instruction window. Due to space reasons, we omit the details of this analysis.

At this point, it is worthwhile to clarify that the correctness of program's execution is not affected by the quality of

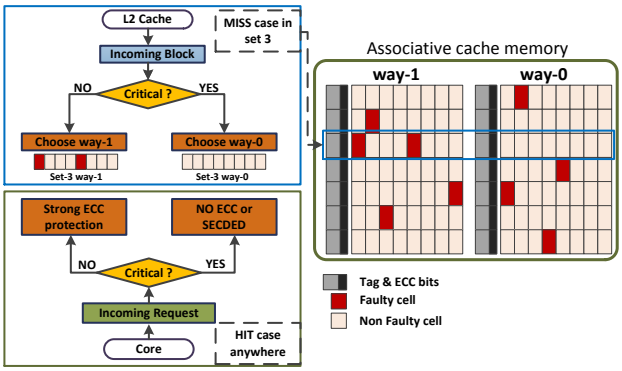

Fig. 2. The proposed mechanism.

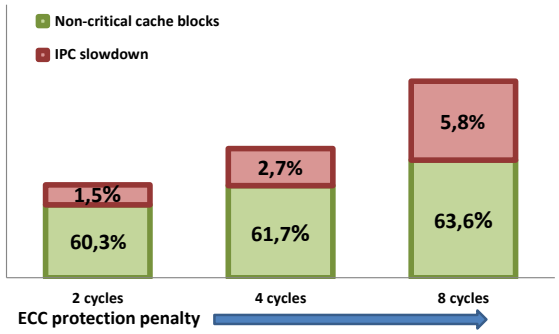

Fig. 3. Percetange of non-critical loads (green bars) and the impact on execution time when the critical loads are delayed (red bars).

criticality predictions. This is because, in our cache architecture, all cache frames are ECC-protected, but the process of data correction is activated only in cases that a specific data block resides in a faulty cache frame. Wrong predictions will result only in performance losses that would be avoided if more sophisticated predictions are issued.

However, in order to reveal the potential of the proposed methodology, Fig. 3 shows the impact on execution time when a specific percentage of non-critical loads is delayed. The results are extracted in SimpleScalar assuming an 8-issue, Out-of-Order, superscalar processor with 192-entries Reorder Buffer and a single-cycle 16KB L1 data cache. The green part of bars in Fig. 3 corresponds to the percentage of cache blocks characterized as non-critical by one of our criticality predictors. These blocks are delayed during cache access by 2 (stacked bar in left), 4 (middle), or 8 (right) cycles respectively. In essence, those delays might correspond to ECC implementations of different error recovery strengths e.g., SECDED and DECTED. As Fig. 3 indicates, by selectively employing ECC only to non-critical data, a minor increase in the execution time is observed e.g., 2.7% in the 4-cycle ECC case (stacked bar in the middle). Most importantly, a significant portion of data blocks is characterized as non-critical (more than 60% in all cases), thereof this portion of data blocks can be served by faulty cache frames.

References

[1] J. Abella, J. Carretero et al. Low Vccmin Fault-Tolerant Cache with Highly Predictable Prtformance. *Intl. Sym. on Microarchitecture*, 2009.

[2] F. Hijaz, Q. Shi et al. A private level-1 cache architecture to exploit the latency and capacity tradeoffs in multicores operating at near-threshold voltages. *Intl. Conference on Computer Design*, 2013.

[3] H. Kaul, M. Anders et al. Near-threshold voltage (ntv) design: opportunities and challenges. *Intl. Design Automation Conference*, 2012.

[4] M. Mavropoulos, G. Keramidas et al. A Defect-Aware Reconfigurable Cache Architecture for Low-Vccmin DVFS-Enabled Systems. *Intl. Conference Design, Automation, and Test in Europe*, 2015.

[5] R. Rakvic, B. Black et al. Non-vital loads. *Intl. Conf. on High-Performance Computer Architecture*, 2002.

[6] ST. Srinivasan and A. Lebeck. Load latency tolerance in dynamically scheduled processors. *Proc. of Intl. Symp. on Microarchitecture*, 1998.

[7] C. Wilkerson, H. Gao et al. Trading off cache Capacity for realiability to Enable Low Voltage Operation. *Intl. Symp. on Computer Architecture*, 2008