Breaking the Energy Barrier in Fault-Tolerant Caches for Multicore Systems

Paul Ampadu, Meilin Zhang
Dept. of Electrical and Computer Engineering
University of Rochester
Rochester, NY, 14627, USA
<ampadu, mezhang>@ece.rochester.edu

Vladimir Stojanovic
Dept. of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, 02139, USA
vlada@mit.edu

Abstract—**Balancing cache energy efficiency and reliability is a major challenge for future multicore system design. Supply voltage reduction is an effective tool to minimize cache energy consumption, usually at the expense of increased number of errors. To achieve substantial energy reduction without degrading reliability, we propose an adaptive fault-tolerant cache architecture, which provides appropriate error control for each cache line based on the number of faulty cells detected at reduced supply voltages. Our experiments show that the proposed approach can improve energy efficiency by more than 25% and energy-execution time product by over 10%, while improving reliability up to 4X using Mean-Error-To-Failure (METF) metric, compared to the next-best solution at the cost of 0.08% storage overhead.**

Keywords—Energy efficiency, fault tolerance, cache, VLSI, multicore.

I. INTRODUCTION

Energy efficiency and reliability are two main concerns in future many-core systems design. On the one hand, technology scaling makes it possible to integrate unprecedented number of transistors on a single chip, enabling many-core system design. On the other hand, this integration introduces energy and reliability issues for the whole system. Moreover, as the speed gap between processor and external memory increases, large-volume and high-density caches are required, worsening reliability and energy issues further. Supply voltage scaling is one of the most effective ways to reduce system energy consumption at the cost of performance and reliability. Low supply voltages increase the impact of process and other variations on circuit functionality and performance. Eventually, the system will fail below a minimal supply voltage V_{DDMIN}. Typically, the cache arrays in the system limit V_{DDMIN} of the whole processor [1]. Consequentially, it is necessary to provide fault tolerance for the cache under low supply voltages to improve system energy efficiency while maintaining reliability.

We can classify cache errors as hard or soft errors [2][3]. Hard errors may be caused by manufacturing defects, threshold or supply voltage variations, or device aging, and soft errors are introduced by external particle strikes or other random noise. To address both hard and soft errors in low-supply voltage cache system, an adaptive fault-tolerant cache architecture, which provides appropriate error control capability for each cache line based on the number of faulty cells detected, is proposed to enable reliable energy-efficiency cache operation. The remainder of the paper is organized as follows: Section II reviews various fault-tolerant architectures for low-supply voltage cache operation; in Section III, the proposed adaptive fault-tolerant architecture is described; Section IV provides analysis and experimental results about energy efficiency, cache reliability, and storage overhead; conclusions are presented in Section V.

II. RELATED WORK

To address cache soft error, various error control codes (ECC) have been investigated and employed. IBM Power 6 applies parity code to L1 cache, and single-error correcting double-error detecting (SECDED) codes to L2 cache [4]. Chishti et al., employ 4-bit segmented orthogonal Latin square coeds (OLSC) to correct multiple faulty cells in single cache line [5]. Wilkerson et al., propose strong BCH code, such as 5-error correcting and 6-error detecting (5EC6ED) code to address reliability issues in the case of high cache error rate [6]. However, these simple ECCs are ineffective in handling the increasing burst or high error rates in aggressively scaled technologies; unfortunately, complex ECCs introduce large power, area and latency overhead.

One general solution to burst errors is interleaving [7][8]. Basically, two types of interleaving, inter-cacheline and intra-cacheline, have been proposed. Inter-cacheline interleaving distributes burst error among different cache lines, and applies ECC to each of them. This requires multiple cache line access even when only one is required, leading to unnecessary power and energy overhead. Intra-cacheline interleaving distributes burst errors among different sub-blocks within the same cache line, and applies ECC to each sub-block. Both interleaving methods are only effective against single burst errors but fail in the case of multiple random errors.

Various alternative solutions, which combine ECC and interleaving, have also been proposed to address soft errors. Kim et al., use two-dimensional ECC to deal with multi-bit clustered errors [9]. Yoon et al., propose a so-called memory mapping ECC, which applies weak but simple ECC codes to all cache lines, and store strong but complex ECC codes into the next-level memory [10]. These methods can increase error

This work was supported in part by the U.S. National Science Foundation (NSF) under Grants ECCS-0925993, ECCS-0903448, CAREER Award ECCS-0954999, and the Semiconductor Research Corporation Award SRC-2009-HJ-2000.

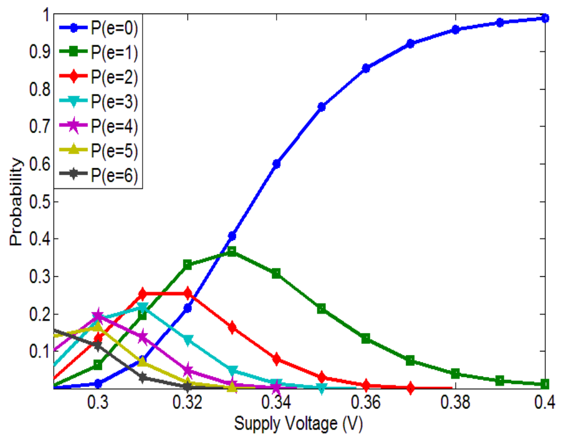


Fig. 1. Distribution of number of faulty cells in single cache line.

control capability to some extent but can still be ineffective in the case of high error rate.

Traditional approaches addressing cache hard errors introduce spare SRAM rows/columns [11][12]. These redundant rows/columns replace those faulty ones in the case of hard error. Spare rows/columns approach can be effective in the case of low error rate, but introduces too much redundant area and power overhead as cache error rate increases.

Agarwal et al., proposed a process-tolerant cache architecture using cache line disabling technique, which detects and deletes cache lines containing faulty cells statically [13]. In their scheme, the number and location of faulty cells are obtained and stored in a configurator, by performing a conventional BIST once operation conditions change. Based on this information, cache lines with faulty cells will be disabled during cache access operation. Cache line disabling can be effective when the cell failure rate is not high. However, as the number of faulty cells increases at lower supply voltage, cache capacity and system performance shrinks rapidly.

Other cache disabling methods include cache way disabling, cache set disabling, and sub-block disabling [14][15]. For cache way disabling technique, full cache ways will be disabled whenever they contains one or several faulty cells; cache set disabling technology disable full cache sets when they contains one or several faulty-cells; sub-block disabling technique only discard faulty sub-block, while the reset faulty-free sub-block can still be used in the case of faulty cells. The cache way and set disabling techniques discard too many fault-free cache lines, although they only introduce limited area and complexity overhead. Sub-block disabling technology can introduce less cache downsizing under faults but at the cost of large complexity and area overhead.

Lu et al., propose dynamic cache reconfiguration with error-correcting codes to address both hard error and soft error at low voltage [16][5]. In low supply voltage mode, a portion of the cache stores additional ECC information. All the physical ways in each cache set are divided into data ways and ECC ways. ECC ways store error correcting codes for data ways to correct possible soft or hard errors. Therefore, during low voltage operation, the proposed dynamic cache reconfiguration scheme sacrifice cache capacity by 50% assuming there is one ECC way for every data way as described in the work. Alameldeen et al., proposed to employ variable-strength ECC (VS_ECC) [17]. They use low error correction ECC in the common case and only use high error correction ECC for those ways containing multiple faulty cells. Three different variable-strength ECC schemes are proposed, and VS_ECC_Disabling is the most effective method among them. Our previous work proposes DECTED with cache line disabling to address both cache hard and soft error [24].

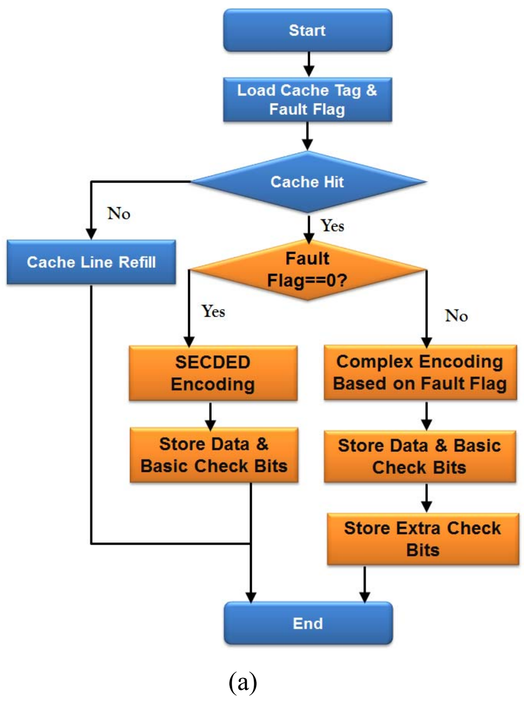
At the circuit level, a traditional approach toward reliable cache design is to increase memory cell size by employing larger transistor. In contrast to simply upsizing 6-transistor cell design, recent work has demonstrated reliability benefits with an 8-T or even a 10-T cell design. Chang et al., proposed adding two data output transistor to a conventional 6T-SRAM cell, which separates data read element from data retention element, in order to improve reliability [18]. Calhoun et. al., proposed adding four transistors to implement a buffer used for reading, which can separate the read and write word lines and bit lines [19]. These techniques can improve cache reliability to some extent but at the cost of high area overhead and low cache density.

III. ADAPTIVE CACHE FAULT-TOLERANT ARCHITECTURE

In this section, we present the proposed adaptive cache fault-tolerant architecture in details. First, the distribution of the number of faulty cells in a single cache line, extracted from a real SRAM test chip[20], is shown to motivate our approach. Then, the proposed adaptive cache fault-tolerant architecture is detailed.

A. Motivation

Reducing supply voltage (V_{DD}), in the presence of process variations, can cause cache cell operation failure. Certainly, the distribution of the number of faulty cells in a cache line varies with supply voltage. Fig. 1 plots the distribution of the number of faulty cells at various supply voltages. Here, *P(e=0)* represents the probability of no fault, *P(e=1)* is the probability that a cache line contains one faulty cell, *P(e=2)* represents the probability of two faulty cells in a single cache line, and so on. Our analysis is based on faulty cell rate extracted from real SRAM chip. As shown in the figure, at high supply voltage, most cache lines are fault-free. As supply voltage decreases, the number of cache lines with a single faulty cell increases significantly first. With further supply voltage scaling, the number of cache lines with multiple faulty cells start to increase also. Therefore, a uniform cache fault-tolerant approach will either pay too much overhead or provide insufficient error control capability. In this paper, we propose an adaptive cache fault-tolerant system, which provides appropriate error control capability for each cache line, to enable low supply voltage cache operation while maintaining reliability.



Start
Load Cache Tag & Fault Flag
Cache Hit
No
Yes
Cache Line Refill
Fault Flag==0?
Yes
No
SECDED Encoding
Complex Encoding Based on Fault Flag
Store Data & Basic Check Bits
Store Data & Basic Check Bits
Store Extra Check Bits
End
(a)

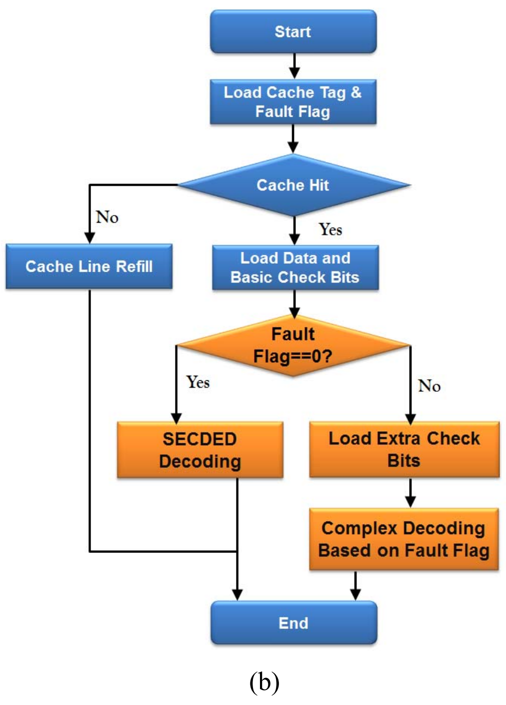


Fig. 2. (a) Flow chart of cache write operation, (b) flow chart of cache read operation.

B. Adaptive Fault-Tolerant Cache Approach

The key idea is to provide suitable error correction for each cache line adaptively based on the number of faulty cells detected by a conventional BIST. We use SECDED codes for each cache lines to address soft errors, and extra parity bits are used when there are hard errors. In the proposed approach, one field, called *fault flag,* is appended to cache tag for each cache line. *Fault flag* represents the number of faulty cells in the related cache line. For example, *Fault flag* =0 means there is no faulty cell in the cache line, *Fault flag* =1 means there is 1 faulty cell in the cache line, *Fault flag* =2 means there is 2 faulty cells in the cache line, and so on. In addition, the maximal *Fault flag* value is used to indicate the number of faulty cells is out of maximal ECC error correction capability (ECC_MAX). Meanwhile, the length of *Fault flag* is also determined by ECC_MAX. The higher ECC_MAX is, the high reliability can be provided by adaptive fault-tolerant cache approach. However, high ECC_MAX also leads to high overhead. In the case of cache operation condition changing, such as scaling supply voltage, a traditional BIST procedure is performed to configure Fault flag for each cache line.

Fig. 2(a) and (b) shows flow chart of cache write operation and read operation for the proposed adaptive fault-tolerance cache architecture. During cache read operation, both cache tag and associated Fault flag are loaded. As conventional cache architecture, we refill the cache line in the case of cache miss. However, new cache data is only refilled into cache lines with *Fault flag* !=ECC_MAX, which means errors can be corrected by ECC codes. Then, *Fault flag* is checked. In the case of Fault flag=0, which means there is no hard error in the associated cache line, a simple SECDED encoding process is performed to protect the cache line against soft error; in the case of Fault flag !=0, extra check bits are loaded to perform complex encoding process. For example, *Fault flag* of 1 indicates performing DECTED encoding process, *Fault flag* of 2 indicates performing 3EC4ED encoding process, and so on. At last, cache data, basic check bits, and extra check bits are stored.

Similarly, during cache write operation, both cache tag and *Fault flag* are loaded first, and cache line is refilled in the case of cache miss as conventional cache architecture. Again, new cache data is only refilled into cache lines with *Fault flag* !=ECC_MAX to the number of faulty cells can be corrected by ECC codes. Then, cache data and basic check bits are loaded, and *Fault flag* is checked. In the case of *Fault flag* =0, a simple SECDED decoding process is performed to protect the cache line against soft error; in the case of *Fault flag* !=0, extra check bits are loaded, and complex ECC decoding are performed based on the value of *Fault flag*.

IV. Experimental Results

In the section, we compare the proposed adaptive fault-tolerant approach with SECDED, DECTED, MS-ECC [5], VS_ECC_Disabling [17] in terms of reliability, energy, energy-execution time product, and storage overhead.

A. Experimental Setup

Our evaluation is based on a 1024-tile many-core system in 11 nm predictive tri-gate technology [21]. Each tile consists of a single-issue 64 bit core, 4-way 32KB private L1 instruction cache, 4-way 32KB private L1 data cache, and 16-way 256KB private L2 cache (Table 1 and Fig. 3). A mesh network-on-chip (NoC) with worm-hole flow control is adopted. Least Recently Used (LRU) cache replacement policy is used for

Table 1 1024-Tile System Parameter

Parameters		Value
Tile Number		1024
Core Type		64 bits single-issue
L1-I/L1-D Cache, Private		
	Cache Block Size	64 bytes
	Cache Size	32 Kbytes
	Associativity	4
	Replacement Policy	LRU
L2 Cache, Private		
	Cache Block Size	64 bytes
	Cache Size	256 Kbytes
	Associativity	16
	Replacement Policy	LRU

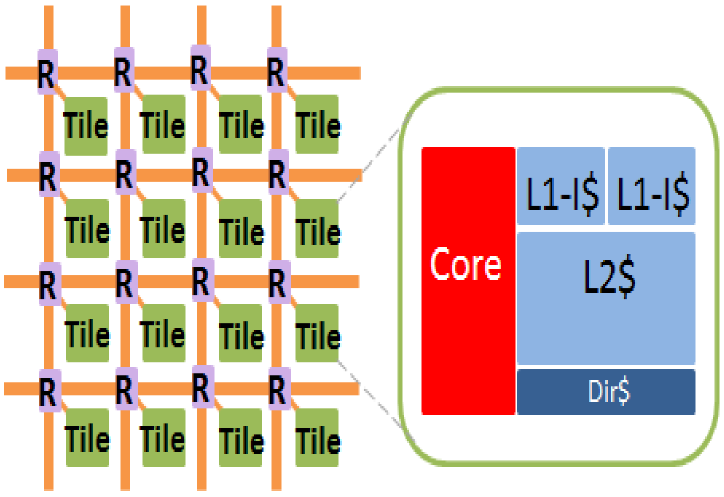
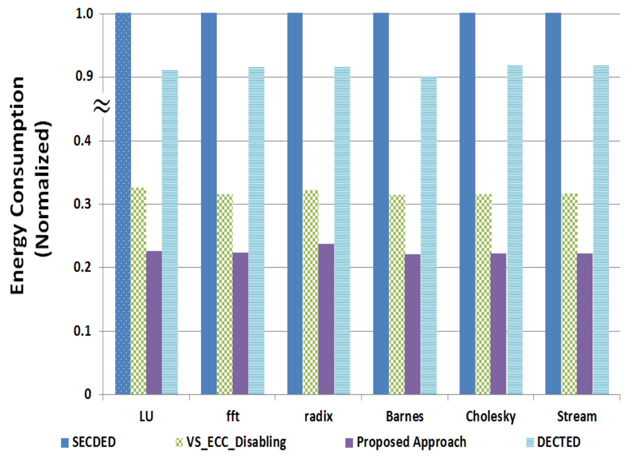
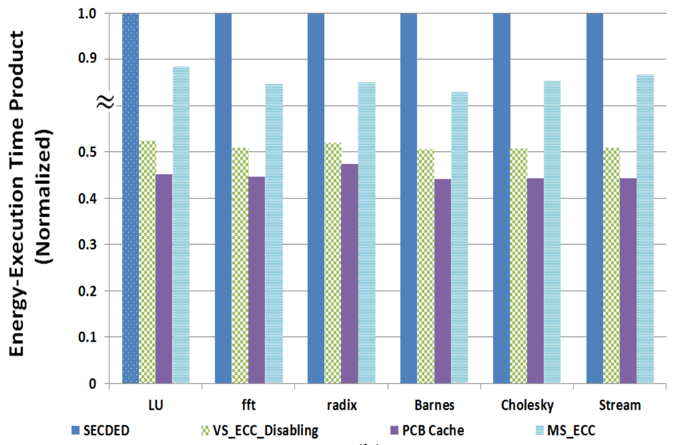


Fig. 3. 1024-Tile System Architecture.

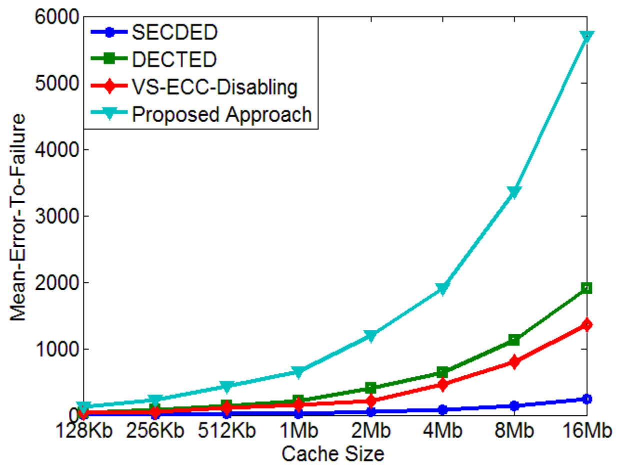


(a)

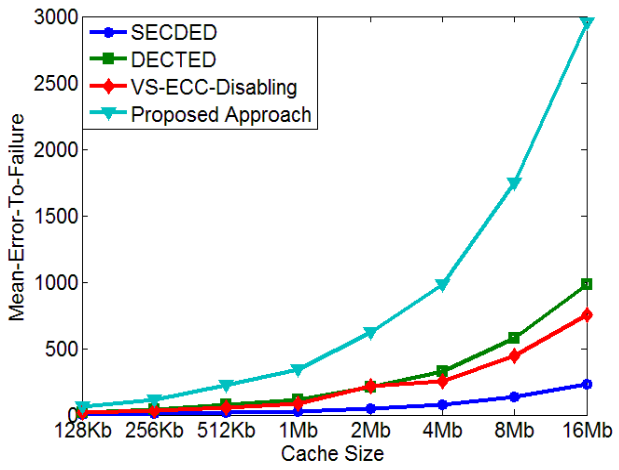


(b)

Fig. 4. (a) Energy consumption comparison, and (b) energy-execution time product comparison.

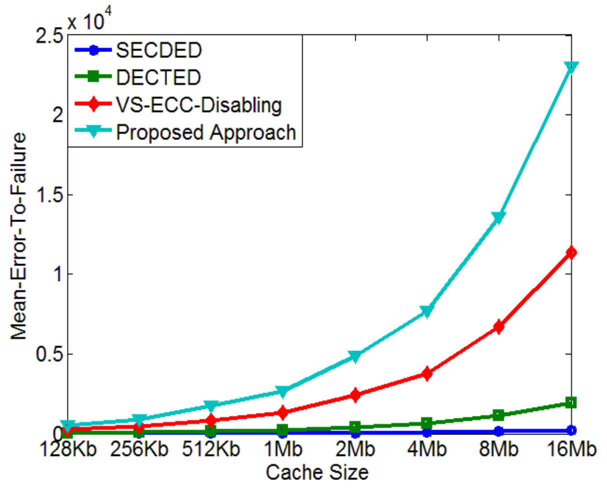


(a)

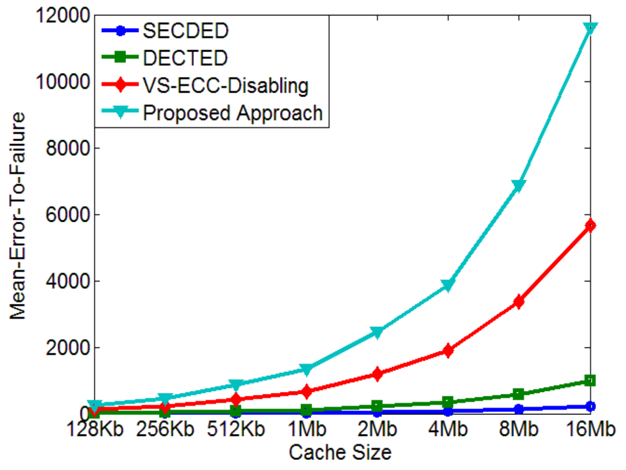


(b)

Fig. 5. Mean-Error-To-Failure comparison (a) cache line size = 512 bits, effective cache size rate = 100%, (b) cache line size = 1024 bits, effective cache size rate = 100%.



(a)



(b)

Fig.6. Mean-Error-To-Failure comparison (a) cache line size = 512 bits, effective cache size rate = 90%, and (b) cache line size = 1024 bits, effective cache size rate = 90%.

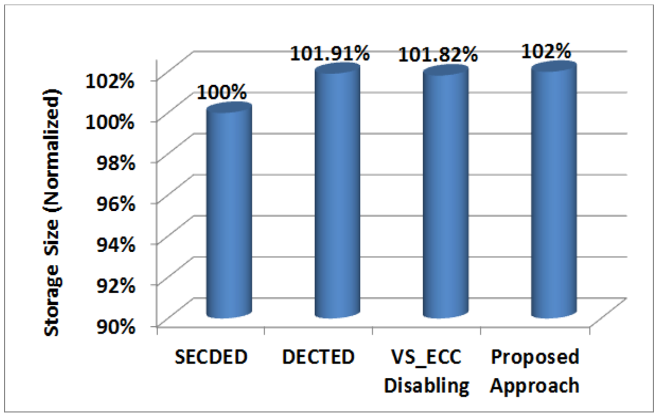


Fig.7. Storage requirement comparison.

both L1 and L2 caches. We use a parallel many-core simulator, Graphite [22], which integrates CACTI [23], to evaluate performance and energy consumption. For the proposed approach, we assume maximal ECC error correction capability of 4, and 120 extra parity bits for each set to control encoding/decoding complexity and storage overhead.

B. Energy Efficiency Evaluation

Several standard benchmarks, including LU, FFT, Radix, Barnes, Cholesky and Stream are used in our simulation of energy and performance. Fig. 4 (a) and Fig. 4 (b) show energy and energy-delay product results of our 1024-tile many-core system using various fault-tolerance approaches. Here, we assume cache yield loss of 10^{-3} and effective cache size of 90%. As shown in the figure, our approach achieves the best energy consumption and energy-delay product among all fault-tolerance methods. Specifically, our approach reduces energy consumption by 76%-78%, 68%-71%, and 25%-29% respectively compared to SECDED, DECTED, and VS_ECC_Disabling. Furthermore, our approach reduces energy-execution time product by 52%-55%, 46%-49%, and 10.1%-12.9% respectively compared to SECDED, DECTED, and VS-ECC-Disabling respectively.

C. Reliability Evaluation

In order to evaluate reliability of various cache fault-ttolerant approaches, we simulated mean-error-to-fail (METF), which represents the average number of errors to cause cache failure. Our simulations are run for various cache sizes, such as 128Kb, 512Kb, 1Mb, 2Mb, 4Mb, 8Mb and 16Mb. For each cache size, 1000 simulations are tested. VS_ECC_Disabling and the proposed approach can shrink cache size under different faulty cells number. Therefore, we test two different cases: all 100% cache lines are effective, which means no cache line is disabled, and 90% cache lines are effective, which means 10% of cache lines are disabled. Fig. 5(a), 5(b), 6(a) and 6(b) show Mean-Error-To-Failure (METF) for four different scenarios: 512-bits cache line and 100% effective cache size rate, 1024-bits cache line and 100% effective cache size rate, 512-bits cache line and 90% effective cache size rate, and 1024-bits cache line and 90% effective cache size rate, respectively.

In all the scenarios, the proposed approach can achieve highest METF, which means highest reliability. For example, in the case of original cache size =8Mb, 100% effective cache size rate, and cache line size=512 bits, the proposed approach provides METF of 3365, which is 4.2x, 3.0x, 23.7x METF achieved by VS_ECC_Disabling, DECTED and SECDED respectively. In the case of original cache size =8Mb, 90% effective cache size rate, and cache line size=512 bits, the proposed approach provides METF of 13575, which is 2.0x, 12.0x, 95.6x METF achieved by VS_ECC_Disabling, DECTED and SECDED respectively.

One interesting observation is that VS_ECC_Disabling achieves slightly less METF than that DECTED did when effective cache size rate is 100%. However, when effective cache size rate is 90%, VS_ECC_Disabling achieves much higher METF than DECTED did.

D. Storage Overhead Evaluation

To correct t-bit errors and detect (t+1)-bit errors, a BCH code require $t*m + 1$ check bits, where m is the smallest possible number that meets the requirement $(k+t*m+1) < 2^m$. Therefore, for a 64 bytes (512 bits) cache line, SECDED ECC requires 11 check bits, DECTED requires 21 check bits, and 4EC5ED requires 41 check bits.

For SECDED design, each set consists of 16 cache lines, which has 523 bits. Thus, each set has 8368 bits. Similarly, For DECTED, each cache line has 533 bits, and each set has 8528 bits. For VS_ECC, among one set, there are four cache lines with 4EC5ED codes, and 12 cache lines with SECDED codes. Besides, each cache line has two status bits, and thereby there are 32 status bits in each set. Thus, each set has 8520 bits. The proposed approach has 3 *fault flag* bits, 11 basic check bits for each cache line, and extra check bits are 120 bits for each set. Therefore, there are 8536 bits for each cache line. Therefore, compared to SECDED design, DECTED design has storage size of 101.91%, VS_ECC_Disabling has storage size of 101.82%, and the proposed approach has storage size of 102% (as shown in Fig.7). Comparing to VS_ECC_Disabling, the proposed approach only introduces 0.08% storage overhead.

V. Conclusion

Energy efficiency and reliability are two main concerns for future many-core systems. Because of ever-increasing speed gap between external memory and processor core, high density and volume cache is required for the system, which exacerbates existed energy and reliability issue. This paper proposes an adaptive cache fault-tolerant system, which provides appropriate error correction for each cache line capability based on the number of faulty cells detected by a traditional BIST, to enable energy-efficient many-core system while maintaining reliability. Our experiments show that the proposed approach can improve energy efficiency by more than 25% and energy-execution time product by more than 10%, and increase reliability by 102%-317% using METF metric, compared to next-best solution at the cost of 0.08% storage overhead.

References

[1] H. R. Ghasemi, et al., “Low-voltage on-chip cache architecture using heterogeneous cell sizes for multi-core processors,” *IEEE International Symposium on High-Performance Computer Architecture*, pages 38–49, February 2011.

[2] M. Agostinelli, et al., “Erratic fluctuations of SRAM cache Vmin at the 90nm process technology node,” *International Electron Devices Meeting*, Dec 2005.

[3] S. Borkar, “Designing reliable systems from unreliable components: the challenges of transistor variability and degradation,” *IEEE Micro*, 25(6):10–17, Nov-Dec 2005.

[4] K. Reick, et al. “Fault-Tolerant Design of the IBM Power6 Microprocessor ”, *IEEE Micro*, Vol. 28, Issue 2, pp30-38, 2008

[5] Z. Chishti, et al., “Improving Cache Lifetime Reliability at Ultra-low Voltages”, *International Symposium on Microarchitecture*, pp. 89-99, Dec. 2009.

[6] C. Wilkerson, et al., “Reducing Cache Power with Low-Cost, Multi-bit Error-Correcting Codes,” *Proc. ISCA*, 2010.

[7] D. Radaelli, H. Puchner, S. Wong, and S. Daniel, “Investigation of multi-bit upsets in a 150 nm technology SRAM device,” *IEEE Trans. Nucl. Sci.*, vol. 52, no. 6, pp. 2433–2437, Dec. 2005.

[8] S. Baeg, S. Wen, and R. Wong, “SRAM Interleaving distance selection with a soft error failure model,” *IEEE Transactions on Nuclear Science*, vol. 56, no. 4, pp. 2111-2118, Aug. 2009.

[9] J. Kim, et al., “Multi-bit Error Tolerant Caches Using Two-Dimensional Error Coding,” *40th International Symposium on Micro-architecture (Micro-40)*, December 2007.

[10] D. H. Yoon and M. Erez, "Memory Mapped ECC: Low-Cost Error Protection for Last Level Caches", *Proceedings of the 36th International Symposium on Computer Architecture (ISCA-36)*, pp. 116-127, June, 2009.

[11] Stanley Schuster, “Multiple Word/Bit Line Redundancy for Semiconductor Memories”, *IEEE Journal of Solid-State Circuits*, vol. 13, no. 5, pp. 698-703, Oct. 1978.

[12] M. Horiguchi, “Redundancy techniques for high-density DRAMS,” *Proceedings of the 2nd Annual IEEE Int. Conf. Innovative Systems Silicon*, Oct. 1997, pp. 22–29.

[13] A. Agarwal et al., "A Process-Tolerant Cache Architecture for Improved Yield in Nanoscale Technologies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol 13, no 1, pp. 27-38, Jan. 2005.

[14] H. Lee, S. Cho, and B. R. Childers, “Performance of Graceful Degradation for Cache faults,” *ISVLSI ’07*, Mar. 2007.

[15] J. Abella et al., “Low Vccmin Fault-Tolerant Cache with Highly Predictable Performance,” *MICRO ’09*, Dec. 2009.

[16] S.-L, Lu, et. al, “Architectural-level error-tolerant techniques for low supply voltage cache operation”, In *ICICDT ’11*, 2011.

[17] A. Aladmeldeen, I. Wagner, Z. Chishti, W. Wu, C. Wilkerson, and S. Lu, "Energy-efficient cache design using variable-strength error-correcting codes," *38th International Symposium on Computer Architecture*, 2011.

[18] L. Chang, D. Fried, J. Hergenrother, J. Sleight, R. Dennard, R. R. Montoye, L. Sekaric, S. McNab, W. Topol, C. Adams, K. Guarini, and W. Haensch, “Stable SRAM cell design for the 32 nm node and beyond,” *Symp. VLSI Technology Dig.*, 2005, pp. 128–129.

[19] H. Calhoun and A. Chandrakasan, “A 256 kb sub-threshold SRAM in 65 nm CMOS,” *IEEE J. Solid-State Circuits*, vol. 42, no. 3, pp. 680–688, Mar. 2007.

[20] Z. Guo, et al., “Large-Scale SRAM variability characterization in 45nm CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 44, no.11, pp. 3174-3192, Nov. 2009.

[21] G. Kurian, et. al., “Cross-layer Energy and Performance Evaluation of a Nanophotonic Manycore Processor System using Real Application Workloads”, *IPDPS* 2012.

[22] J.Miller et al., “Graphite: A Distributed Parallel Simulator for Multicores,” *HPCA*, 2009.

[23] S. Thoziyoor, J. Ahn, M. Monchiero, J. Brockman, and N. Jouppi, “A Comprehensive Memory Modeling Tool and its Application to the Design and Analysis of Future Memory Hierarchies,” *ISCA*, 2008.

[24] M. Zhang, V. Stojanovic, and P. Ampadu, “Reliable Ultra-Low Voltage Cache Deisgn for Many-Core Systems,” *IEEE Trans. on Circuit and Systems II-Express Briefs* (to appear).