This paper appears in the 2017 IGSC Invited Papers on Sustainable Memories

Holistic Energy Efficient Crosstalk Mitigation in DRAM

Donald Kline, Jr
Electrical and Computer Engineering
University of Pittsburgh
Email: dek61@pitt.edu

Rami Melhem
Computer Science
University of Pittsburgh
Email: melhem@cs.pitt.edu

Alex K. Jones
Electrical and Computer Engineering
University of Pittsburgh
Email: akjones@pitt.edu

***Abstract*—The scaling of DRAM to increasingly small geometries has resulted in considerable challenges to both reliability and energy consumption of memory systems. For example, this aggressive scaling has resulted in increased vulnerability to both bitline and wordline crosstalk. Moreover, deep scaling has also introduced an understudied implication of dramatically increased *embodied* energy, or energy due to manufacturing memory integrated circuits. While many correction schemes have been proposed targeting, often independently, metrics of reliability and operational energy, recent studies have demonstrated that the impacts of manufacturing on reliability and holistic energy consumption must also be considered.**

In this work, we propose a technique to evaluate memory systems and their tradeoffs for reliability, embodied energy, and operational energy. We use this technique to examine several proposed correction schemes for DRAM faults. Further, we study a novel bitline crosstalk error correction scheme, Periodic Flip Encoding, which has considerable advantages in sustainability and reliability at high error rates.

I. Introduction

Dynamic random access memory (DRAM) has been the principal technology for main memory for decades. In recent years, as computational devices constitute a growing percentage of energy consumed in modern countries, many companies and research teams have strived to minimize the consumption of energy in the *use phase* of such DRAM systems. While significant progress has been made in this area, recent studies [1–4] have demonstrated that the *embodied energy* of the manufacturing phase of these devices comprises a significant and in some cases dominant fraction of the total energy consumed over the lifetime of the device. Therefore, modern day architectural design choices should consider both the use phase and manufacturing impacts when determining the best overall solution for energy.

An additional critical challenge of DRAM, which has become pressing with deeply scaled geometries, is error avoidance, detection, and correction. With the reduced tolerance of coupling noise at shrinking dimensions, DRAM becomes more susceptible to both bitline and wordline crosstalk. While wordline crosstalk existed in what are now legacy technologies, concerns about the row-hammering security vulnerability [5] have recently ignited significant interest in the subject. Proposed solutions include the use of traditional error correction schemes [6], per-row and groups of counters [7], probabilistic refreshing [5], reducing the refresh interval, and runtime testing to identify weak (prone to crosstalk) DRAM cells [8].

*This work was supported in part by the NSF GRFP No. 1247842 and an MCSI Faculty Fellowship.

DRAM manufacturers have long used circuit-level techniques to mitigate bitline crosstalk by improving inter-cell isolation [9] and have also used post-production testing to check for such disturbance errors [10]. While circuit-level techniques reduce the bitline coupling noise, they complicate the layout, decrease storage density, exacerbate wordline crosstalk, and increase the risk of creating a short-circuit between non-adjacent bitlines. Therefore, several DRAM manufacturers have opted to abandon bitline twisting and other related techniques, and instead use the open bitline structure at the expense of higher bitline crosstalk noise [11, 12].

Given the increasing reliability concern of crosstalk as well as the growing challenges related to scaling, any design which claims to reduce or incur little additional energy must evaluate potential memory system architectures, not only on their reliability and their timing, but also on their holistic energy footprint. Toward this goal, we make the following contributions:

- We provide an analysis of the minimum required manufacturing overhead for each correction scheme to operate reliably for different ratios of cells vulnerable to wordline crosstalk.
- We provide a holistic energy analysis of several bitline crosstalk mitigation schemes at different usage scenarios and fault map overheads.
- We propose a dynamic mapping scheme which augments bitline correction schemes to achieve the acceptable threshold of error-free operation.

II. Background and Related Work

Recent studies have shown that although scaling increases transistor density, integrated circuit (IC) area within systems is increasing due in part to dramatic increases in transistors dedicated to processing (i.e., cores and hardware accelerators) and in particular memory and storage in these systems [2]. The significant contribution to manufacturing energy of (ICs) [1, 3] in systems is supported further in recent sustainability reports

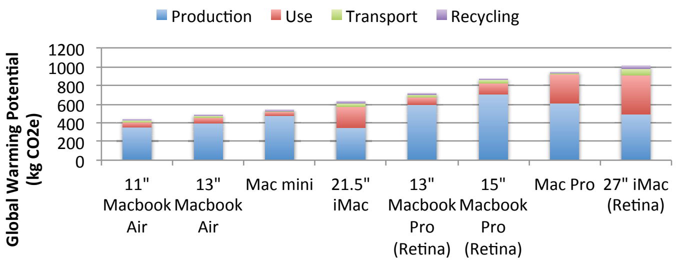


Figure 1. Carbon emissions (CO_2 equivalent) of manufacturing/production of ICs in "use phase optimized" systems from Apple Computer based on a two year operational lifetime [13].

from Apple Computer (Figure 1). Systems without an integrated display, the Mac Pro and Mac Mini, owe over two-thirds of their impacts to manufacturing, ostensibly dominated by ICs. Note that CO_2 emissions tend to follow closely with energy consumption making them analogous indicators of trends. Using our own extension to the GreenChip tool and methodology [4], we can combine embodied and operational energy of DRAM correction schemes, often required for deeply scaled technologies due to increased crosstalk, to provide a holistic evaluation of their overall energy impacts.

A. DRAM Crosstalk

Bitline crosstalk has traditionally been addressed with circuit techniques and wordline crosstalk has been primarily addressed with architecture approaches. Starting with the latter, wordline crosstalk occurs when the toggling of the wordline voltage for a particular row causes certain cells in nearby rows to leak their charge at an accelerated rate. From a previous study, it appears that the susceptibility to wordline crosstalk is not always related to the retention time of the cells, but can be caused by a variety of mechanisms that create "weak" cells [5].

Bitline crosstalk, reemerging as a significant form of crosstalk [14, 15], occurs in deeply scaled DRAM when a weak cell is deflected due to capacitive coupling with its adjacent bitlines [16]. This manifests with "bad patterns" (traditionally "000" and "111") in the data, where a bad pattern over a weak cell can cause the inversion of this middle bit (i.e., "010" or "101") [17]. Until recently, the issue of bitline crosstalk had been successfully mitigated using bitline twisting [9]. Unfortunately, as the technology node descends, bitline twisting exacerbates the occurrence of wordline crosstalk, and thus there has been a trend toward open bitlines [14] to alleviate this problem with wordline crosstalk.

To address crosstalk requires error mitigation through avoidance or correction, which can impact holistic energy consumption. We discuss this further in the next subsection.

B. Error Correction

Error Correction Codes (ECC) is a common, general purpose error correction strategy used in memory and communication. ECC level one (ECC_1) (single error correction, double error detection or SECDED) uses a Hamming code that can be computed through a series of XOR operations. Typically, while in-DRAM ECC exists, chipkill ECC [18] is used in most professional server settings for its ability to tolerate the failure of an entire memory chip in addition to one error in an individual row.

Error correcting pointers, or ECP [19], is a solution proposed for PCM which has also been adapted for DRAM [20, 21]. While ECP was designed for non-volatile memories in which errors are the result of permanent cell failures that are immediately detectable at write time, they have been adapted to handle DRAM hard failures detected at manufacturing time. ECP in DRAM circumvents the original requirement of non-volatility by loading the ECP pointers from a file in secondary storage at boot time [21].

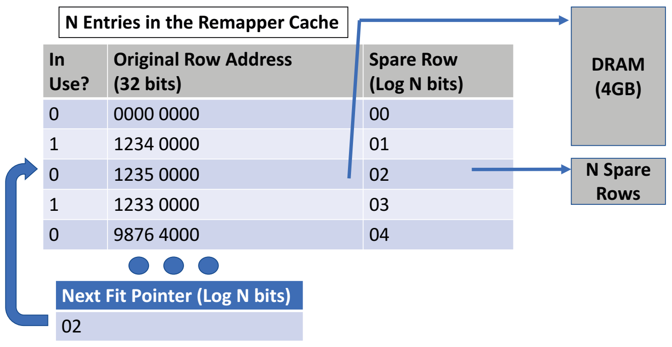
Periodic Flip Encoding (PFE) [22] is a simple yet elegant solution to attempt to eliminate sequences of "000" and "111" in data overlapping with weak cells to be written to the DRAM. PFE works by partitioning a data block into 3-bit groups, which can be collectively manipulated by one of four possible 3-bit patterns ("000," "001," "010," and "100"). These four patterns are represented by two encoding bits, which can be applied to any block size. When encoding, the 3-bit pattern that results in the least overlap of bad patterns with weak cells after XORing with every 3-bit group is chosen for the block. The XOR of the pattern with this original data at every 3-bit interval then creates the code word to be written to the DRAM. When decoding, the auxiliary bits determine the 3-bit pattern, which is again XORed with the block in the same manner to retrieve the original data.

These error mitigation and correction strategies have different redundant storage and correction circuitry overheads resulting in a tradeoff of both embodied and operational energy overheads in addition to other factors such as reliability and performance. All of these schemes can be used in concert with a bank of additional redundant memory locations (row sparing [23]) when error mitigation alone is insufficient; we discuss this further in Section III.

III. Remapper Design

Crosstalk correction schemes, including ECC, ECP, and PFE, can use spare rows to augment their fault tolerance capabilities when error correction alone is insufficient for correctness. However, for data-dependent faults, such as bitline crosstalk, remapping a row will only be necessary with certain data words and not others. We assume a write failure occurs when the data is compared with a memory fault and the number of faults exceeds the protection of the error mitigation scheme, requiring the employment of a spare row.

To facilitate dynamic spare row use, a rowmapping scheme was designed to help address the data dependent employment of row-sparing. This "remapper," shown in Figure 2, stores pointers to the original rows, coupled with pointers to the spare row and a bit to indicate if the spare row is in use. Assuming a 4GB address space, the memory overhead is $N(32+log(N)+1)$ bits, where N is the number of spare rows. Thus, the first four bytes are the original DRAM address of a row which could not successfully write data due to bitline crosstalk, the



In Use?	Original Row Address (32 bits)	Spare Row (Log N bits)
0	0000 0000	00
1	1234 0000	01
0	1235 0000	02
1	1233 0000	03
0	9876 4000	04

N Entries in the Remapper Cache
Next Fit Pointer (Log N bits)
02
DRAM (4GB)
N Spare Rows

Figure 2. Remapper cache and next fit algorithm pointer.

next $log(N)$ bits correspond to a spare row pointer in non-addressable space with a final "in use" (valid) bit for the entry.

The spare rows are allocated using the *next fit* algorithm, a popular option for managing heap space. Next fit requires an additional pointer of size $\lceil log(N) \rceil$ (shown in Figure 2) to point to the next available location. When a new spare row is requested to replace a faulty row during a write, the writing into the spare row is checked. If the write is successful, the "in use" bit is set, and the global pointer is incremented to the next empty location. If the write fails, the next empty entry in the table is attempted, until the pointer has circled back to where it began the operation, at which point the write fails.

In order to minimize the runtime overhead of determining if a row in the DRAM is in the remapper cache, we use a slightly adjusted version of CiDRA [20] where a Bloom filter is used to quickly determine if a row can be accessed directly (i.e., it is not in a spare row). If a row hits in the Bloom filter on a read, the remapper cache is searched to find the corresponding spare row, which is subsequently read from DRAM. On a write, the original row address is always attempted on the hope the new data may not trigger bitline crosstalk faults and the spare row can be freed to store data from another row. In parallel with the write to the original address of the DRAM, in the memory controller, the Bloom filter is searched for the address. Upon a hit in the Bloom filter and a successful write in the original row, the corresponding address in the remapper cache is located and the "in use" bit is cleared.

IV. Evaluation

To model weak cells of the memory, maps of weak cells were created using a Bayesian distribution to mimic the impact of process variation and include spatial correlation of faults [10, 24]. We followed the model described in [10] to generate maps of weak cells for a 4GB DRAM.

Given the nature of wordline crosstalk, we assume cells susceptible to crosstalk can be detected at test time using deep regression testing. These detected cells can then be used to create a fault map of the memory. Given this assumption, five 4GB fault maps were created for 10 different error rates, using the method discussed in the previous paragraph. The simulation for each error rate ran through every row, correction scheme, and fault map, and for each row that failed after performing the correction scheme, that row was replaced with a spare row. The size of the necessary batch of spare rows for each fault map and correction scheme was thus determined for each error rate. The average of this overhead for the five fault maps, added to the initial overhead required by the correction scheme, then gave the minimum overhead required in order to maintain the original capacity of 4GB at that error rate.

While wordline crosstalk can be evaluated at chip test time, bitline crosstalk is inherently data dependent, and therefore must be evaluated at runtime. Thus, we used DRAM memory traces of the PARSEC benchmarks [25]. Using these traces, we implemented the remapper discussed in Section III to track the additional accesses required, as well as to determine the minimum number of spare rows necessary to ensure reliable operation for each correction scheme. The timing, power, and area overhead of the remapper cache was found using CACTI [26]. The power results for the DRAM itself were calculated by running the traces through DRAMSim2 [27]. The average delays in the remapper cache for each correction scheme and error rate where added to the memory controller delays in Sniper [28] to determine the IPC used in holistic energy computations in Section IV-C. To estimate the use phase energy cost of the correction mechanisms, each scheme was synthesized in 45nm hardware using a Free PDK [29] (with the exception of the counters, which used the synthesized results from [7]). For bitline crosstalk, we analyzed spare rows alone (ECP$_0$), ECP$_1$-ECP$_9$, SECDED ECC, and PFE.

A. *Wordline Crosstalk Overhead Evaluation*

Figure 3 shows a storage overhead comparison of combining row-sparing (avoidance) with error correction to ensure error-free operation. ECC (SECDED 32,39) requires significant initial overhead and is only effective for lower ratios of weak cells, requiring considerable usage of spare rows when the ratio of weak cells becomes extremely high (10^{-2}). ECP$_k$ increases both the number of spare rows and, when necessary, the number of pointers per row, k, as the potential weak cell rate increases, dramatically reducing storage overhead against ECC. Row sparing alone (ECP$_0$) protects with the lowest storage overhead at potential weak cell rates $\leq 10^{-5}$.

In contrast, a tuned approach, which stores an activation counter [7] to refresh the neighboring rows after reaching an access threshold, provides the lowest storage overhead solution for high potential weak cell rates. ECP$_1$ bridges the gap between counters and row-sparing alone. However, ECP, row

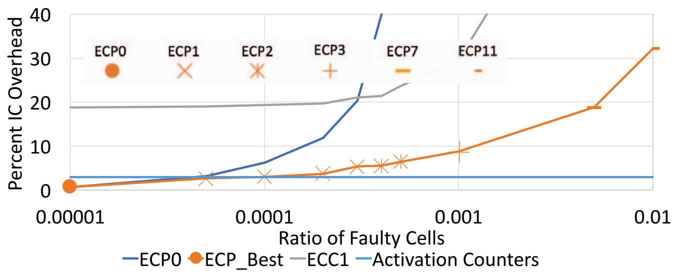


Figure 3. Minimum overhead required for wordline crosstalk fault avoidance (correction overhead plus spare rows).

sparing, and counters cannot protect against less predictable errors such as communication and single-event upsets, particularly of concern in server-grade systems. Thus, protection against these faults can still benefit from ECC/chipkill in concert with dedicated solutions for wordline crosstalk.

B. Bitline Crosstalk

Similar to Figure 3, Figure 4 displays the storage overhead comparison of combining row-sparing (including the dynamic remapper) with error correction for error-free operation. As with Figure 3, Figure 4 does not include underlying ECC/chipkill which would be necessary for server-based systems to protect against transient errors. Spare rows alone (ECP_0) provides the minimum overhead strategy up through $5{\cdot}10^{-4}$, where ECP_1 becomes the minimum. At 10^{-3}, ECP_2 is slightly more efficient in terms of overhead compared to PFE, and beyond this point PFE is the most space efficient error correction scheme.

Figure 5 shows the average DRAM accesses required per remap for each correction scheme as the weak cell ratio increases. Recall from Section III that if an original row cannot successfully write a given data word due to the number of faults (intersections of bad patterns with weak cells) exceeding the correction capability, then the remapper cache is used to find the next available spare row that can successfully write the original data. This metric is directly proportional to the performance penalty required to manage the spare rows and maintain sufficient reliability. After a weak cell incidence rate of larger than 10^{-3}, ECC_1 becomes unusable, rising quickly to hundreds of attempts per spare row allocation. Even with error rates as high as 10^{-2}, PFE only requires on average 1.016 attempts of trying a spare row to find one which can successfully write the data. Increasing the ECP pointers (k) progressively reduces the amount of calculations required, but even at ECP_9 does not approach the efficiency of PFE.

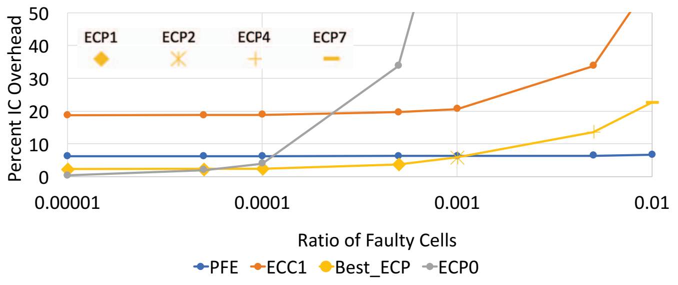


Figure 4. Minimum overhead required for bitline crosstalk fault avoidance (correction overhead plus spare rows). This *excludes* the contribution of the fault map.

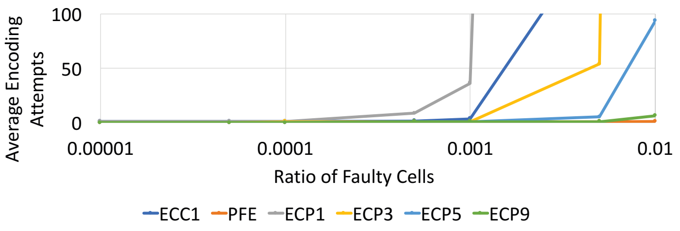


Figure 5. Average encoding attempts per spare row allocation.

C. Holistic Indifference Analysis

A holistic energy analysis is required to understand the true energy tradeoffs of the different error correction schemes. We leverage indifference analysis for evaluating sustainable computing [4], to calculate the time when the total energy, including manufacturing and use phase energy, of two different schemes to achieve the same reliability are equivalent (indifference point) using Equation 1. The use phase powers in the denominator of the expression can be calculated based on different usage scenarios (Equation 2) [4], where r_S is the sleep ratio, r_A represents the active to idle ratio, P_D is the dynamic power, P_S is the static power, and P_L is the sleep power. For each benchmark, we calculate the indifference points for three active and sleep scenarios: a high-performance-computer (HPC) (r_A=0.95,r_S=0.05), mobile computing system (r_A=0.9,r_S=0.92), and cloud server (r_A=0.3,r_S=0.05) [4].

$$t_I = \frac{M_1 - M_0}{P_0 - P_1} \tag{1}$$

$$P = (1 - r_S)(r_A(P_D + P_S) + (1 - r_A)P_S) + P_L \tag{2}$$

The IPC when using the correction scheme was calculated using SNIPER [28] in conjunction with the simulated delays including the encoding hardware and row remapper as required to maintain correctness using the spare rows. The computed IPC can then affect the indifference point calculation by replacing r_A in the power calculation (Equation 2) of system 1 with $r'_A = r_A(\frac{IPC_0}{IPC_1})$ [4]. This adjustment reduces the active time of the system with the higher IPC, accounting for the fact that it needs less time to complete an equivalent workload.

1) Wordline Crosstalk: The holistic energy analysis of wordline crosstalk reveals that the additional static energy in the use phase from the spare rows and correction bits dominates in the comparison between correction schemes. As a result, in all compared cases a higher embodied energy resulted in a higher operational energy, producing a negative result for Equation 1. A negative indifference result indicates that the indifference time is actually infinite, or in other words there is no use phase energy savings to recoup the additional manufacturing energy. Therefore, for the wordline crosstalk correction schemes examined, it makes sense to always choose the scheme with the lower initial manufacturing energy, because it also happens to be correlated with a lower use phase energy. Thus, the most energy efficient scheme results in the choice indicated by Figure 3. For victim cell rates $<= 10^{-5}$, row sparing alone is the most energy efficient method to protect against wordline crosstalk (assuming the victim cells can be accurately profiled). For incidences of victim cells larger than $5{\cdot}10^{-4}$, using a basic system of uniform counters is the most energy efficient solution examined. Various improvements to these basic counters, including those discussed in [7], could potentially further increase this advantage.

2) Bitline Crosstalk: Similar to the results for wordline crosstalk, for bitline crosstalk with both the 10^{-5} and 10^{-4} ratios of weak cells, we find that for all usage scenarios and correction scheme comparisons the result of Equation 1 was

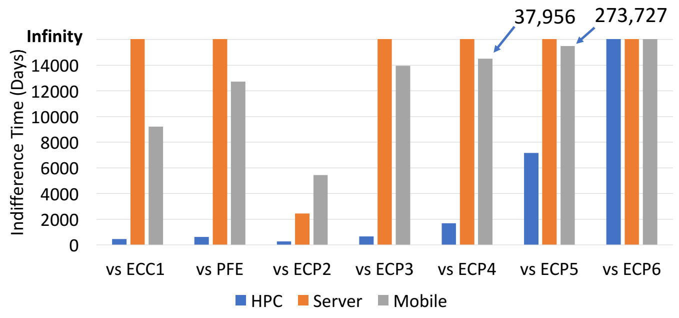


Figure 6. Indifference points (days) for ECP_1 at a 10^{-3} weak cell incidence rate.

also negative, which corresponds to infinite time required to make up the higher manufacturing cost. At the weak cell rate of 10^{-3}, ECP_1 has the lowest manufacturing cost but its less effective correction and many required attempts to find suitable spare rows incurs significant use phase penalties. The indifference point analysis for ECP_1 versus the other correction schemes at 10^{-3} weak cell rate and different usage scenarios can be observed in Figure 6. Because ECP_1 has the lowest manufacturing cost an infinite indifference time in Figure 6 corresponds to ECP_1 always being the more holistically energy efficient solution, while a time less than infinite indicates the time after which the scheme is more energy efficient than ECP_1, due to lower use phase energy.

The comparison of ECP_1 against ECC shows ECC has a higher manufacturing cost, but due to the performance penalty of ECP_1 during attempts to use spare rows to maintain correctness in the use cases of HPC and Mobile systems, ECC_1 has a lower use phase energy. For the server scenario, ECC_1 has both a higher use phase energy and manufacturing cost than ECP_1, resulting in an infinite indifference time, where ECP_1 is always the more energy efficient solution.

For the HPC scenario, both ECC_1 and ECP_2 recover their embodied energy overhead compared to ECP_1 in less than a year, while PFE and ECP_3 take less than two years. For these correction schemes with a life-cycle of at least two years, this indicates that at a 10^{-3} weak cell incidence rate in a high performance scenario they are more energy efficient solutions than ECP_1. In contrast, for the server scenario of usage, PFE and ECP_3 are always less energy efficient than ECP_1. While ECC_1, PFE, and ECP_k where $2 \leq k \leq 5$ all have positive indifference times compared to ECP_1 for the mobile scenario, they are all at least 10 years. Therefore, for systems with lifetimes less than 10 years, ECP_1 is more efficient for the mobile scenario at a weak cell incidence rate of 10^{-3}.

While ECP_1 was the most energy efficient solution for weak cell rates of 10^{-5}, 10^{-4} and in some cases for 10^{-3}, it is never the most energy efficient solution for the 10^{-2} weak cell incidence rate. Figure 7 shows the indifference times of ECP_1 compared to the other correction schemes.The indifference times are all less than 90 days.

Moreover, PFE consistently has the lowest use phase energy overhead, due to its high correction capability and very low use of spare rows. The indifference points for PFE at the 10^{-2} weak cell rate with different usage scenarios can be observed in Figure 8. PFE always has a lower manufacturing cost and correspondingly an infinite indifference time compared to ECC. In comparison with correction schemes with lower manufacturing energies (ECP_1, ECP_2), PFE recovers its additional manufacturing cost within 20 days for each of the three scenarios, clearly indicating the additional manufacturing investment is worthwhile in terms of holistic energy efficiency. For ECP_3 and beyond, PFE has both a lower initial manufacturing energy and a lower use phase energy, indicating it is always the more energy efficient choice.

D. *Fault Map Overheads*

The effectiveness of many error correction methodologies can be enhanced by a knowledge of the location of faulty rows, words, or individual cells. ArchShield uses a word-level fault map in its error correction scheme [8]. SFaultMap is a sustainability aware bit-level fault map designed for deeply scaled memories [30]. These fault maps can dramatically enhance the capabilities of error correction such as ECP and PFE but can also have a significant impact on area, power, and performance overheads of the memory system. Moreover, as error rates tend to increase with scaling, the size of these fault maps relative to the whole memory system will increase significantly.

To see the impact of these fault maps, the manufacturing cost (normalized to ECC) for several different fault map overhead percentages for various correction schemes is shown in Figure 9. For example, a relatively low area encoding technique such as PFE that requires a 10% fault map can increase its manufacturing cost to be on par with ECP_8 without a fault map. Moreover, introducing a substantial fault map (15% of the memory storage overhead or higher) can change some of the indifference analyses from the previous discussion. For example, ECP_1 and PFE are no longer always more efficient than ECC and each requires an indifference analysis to determine which solution to use to minimize holistic energy.

V. Conclusions and Future Work

The analysis of wordline crosstalk indicates that below 10^{-5} incidence rate of susceptible cells, the most energy efficient manufacturing scheme is to simply manufacture a small number of spare rows, test the rows and assign spares to those which fail, and then use a mechanism such as CiDRA [20] for low overhead access to spare rows at runtime. Below 10^{-5} weak cell incidence rate, using counters is the most efficient solution in terms of manufacturing energy and space overhead.

In the context of bitline crosstalk, for both 10^{-5} and 10^{-4} weak cell incidence rates, then ECP_1 is the most appealing candidate in terms of holistic sustainability because of its low manufacturing cost and use phase power. However, for weak cell rates larger than $5 \cdot 10^{-3}$, PFE is the clear and superior choice for sustainability. The size of ancillary overheads from fault tolerance schemes such as fault maps can be a significant factor in determining the most sustainable choices. For PFE,

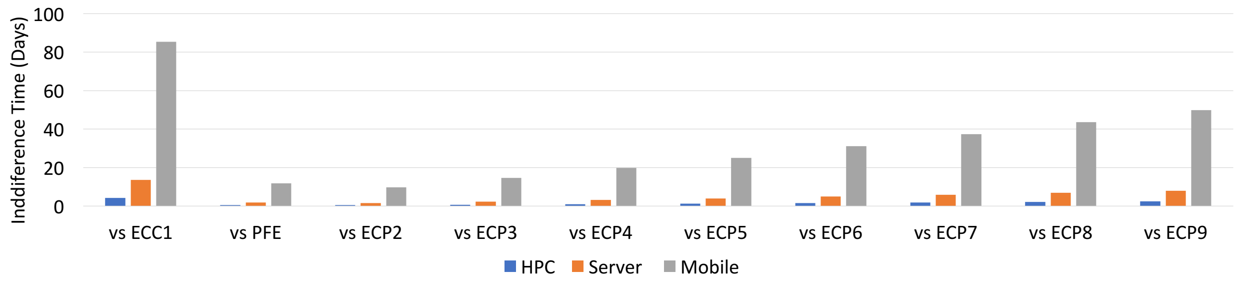


Figure 7. Indifference points (days) for ECP_1 at the 10^{-2} weak cell incidence rate. Indifference times describe the point after which the compared schemes are more energy efficient than ECP_1.

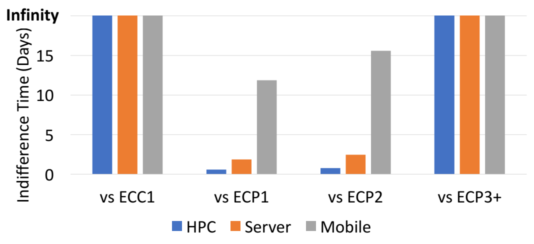


Figure 8. Indifference points (days) for PFE at the 10^{-2} weak cell incidence rate.

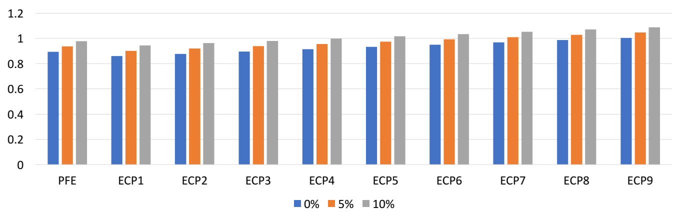


Figure 9. Manufacturing cost normalized to ECC_1 for different fault map overheads.

as long the fault map overhead less than 20% PFE can make up its manufacturing deficits in less than a year (often less than a week). However, to determine accurate indifference points requires deeper analyses of these overheads, which we plan to study in our future work.

References

[1] A. Jones, Y. Chen, W. Collinge, H. Xu, L. Schaefer, A. Landis, and M. Bilec, "Considering fabrication in sustainable computing," *ICCAD*, 2013.

[2] M. A. Yao, T. G. Higgs, M. J. Cullen, S. Stewart, and T. A. Brady, "Comparative assessment of life cycle assessment methods used for personal computers.," *Env. Sci. & Tech.*, Vol. 44, No. 19, 2010.

[3] P. Teehan and M. Kandlikar, "Comparing Embodied Greenhouse Gas Emissions of Modern Computing and Electronics Products," *Env. Sci. & Tech.*, Vol. 47, No. 9, 2013.

[4] D. Kline Jr, N. Parshook, X. Ge, E. Brunvand, R. Melhem, P. K. Chrysanthis, and A. K. Jones, "Holistically Evaluating the Environmental Impacts in Modern Computing Systems," *IGSC*, 2016.

[5] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," *ISCA*, 2014.

[6] M. Seaborn and T. Dullien, "Exploiting the DRAM rowhammer bug to gain kernel privileges," *Google Project Zero Blog*.

[7] S. M. Seyedzadeh, A. K. Jones, and R. Melhem, "Counter-Based Tree Structure for Row Hammering Mitigation in DRAM," *IEEE Computer Architecture Letters*.

[8] P. J. Nair, D.-H. Kim, and M. K. Qureshi, "ArchShield: Architectural framework for assisting DRAM scaling by tolerating high error rates," *ISCA*, 2013.

[9] T. Yoshihara and et al., "A twisted bit line technique for multi-Mb DRAMs," *ISSCC 1998*.

[10] Z. Al-Ars, *DRAM fault analysis and test generation*. TU Delft, Delft University of Technology, 2005.

[11] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi, "Rethinking DRAM design and organization for energy-constrained multi-cores," *SIGARCH Comput. Archit. News 2010*.

[12] S. Khan, C. Wilkerson, D. Lee, A. R. Alameldeen, and O. Mutlu, "A Case for Memory Content-Based Detection and Mitigation of Data-Dependent Failures in DRAM," *in CAL 2016*.

[13] Apple Inc., "Environmental Report." [Available Online]: http://www.apple.com/environment/reports/, 2015.

[14] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An experimental study of data retention behavior in modern DRAM devices: Implications for retention time profiling mechanisms," *ISCA*, pp. 60–71, 2013.

[15] S. Khan, D. Lee, and O. Mutlu, "PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM," *DSN*, 2016.

[16] Z. Yang and S. Mourad, "Crosstalk induced fault analysis and test in DRAMs," *Journal of Electronic Testing*, Vol. 22, pp. 173–187, 2006.

[17] Y. Konishi and et al, "Analysis of coupling noise between adjacent bit lines in megabit DRAMs," *Journal of Solid-State Circuits 1989*.

[18] T. J. Dell, "A white paper on the benefits of chipkill-correct ECC for PC server main memory," *IBM Microelectronics Division*, pp. 1–23, 1997.

[19] S. Schechter, G. H. Loh, K. Strauss, and D. Burger, "Use ECP, not ECC, for hard failures in resistive memories," *ISCA*, pp. 141–152, 2010.

[20] Y. H. Son, S. Lee, O. Seongil, S. Kwon, N. S. Kim, and J. H. Ahn, "CiDRA: A cache-inspired DRAM resilience architecture," *HPCA*, pp. 502–513, 2015.

[21] C.-H. Lin, D.-Y. Shen, Y.-J. Chen, C.-L. Yang, and M. Wang, "SECRET: Selective error correction for refresh energy reduction in DRAMs," *ICCD*, pp. 67–74, 2012.

[22] M. Seyedzadeh, D. Kline Jr, R. Melhem, and A. K. Jones, "Mitigating Bitline Crosstalk Noise in DRAM Memories," *MEMSYS*, 2017.

[23] M. Horiguchi and K. Itoh, *Nanoscale memory repair*. Springer Science & Business Media, 2011.

[24] T. Yuan, S. Z. Ramadan, and S. J. Bae, "Yield prediction for integrated circuits manufacturing through hierarchical Bayesian modeling of spatial defects," *Transactions on Reliability 2011*.

[25] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," *PACT*, 2008.

[26] S. J. Wilton and N. P. Jouppi, "CACTI: An enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits*, Vol. 31, No. 5, pp. 677–688, 1996.

[27] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A Cycle Accurate Memory System Simulator," *IEEE Comp. Arch. Let.*, Vol. 10, No. 1, pp. 16–19, 2011.

[28] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-core Simulation," *SC*, 2011.

[29] J. E. Stine *et al.*, "FreePDK: An open-source variation-aware design kit," *MSE*, pp. 173–174, 2007.

[30] D. Kline Jr, R. Melhem, and A. K. Jones, "Sustainable Fault Management and Error Correction for Next-Generation Main Memories," *IGSC*, 2017.