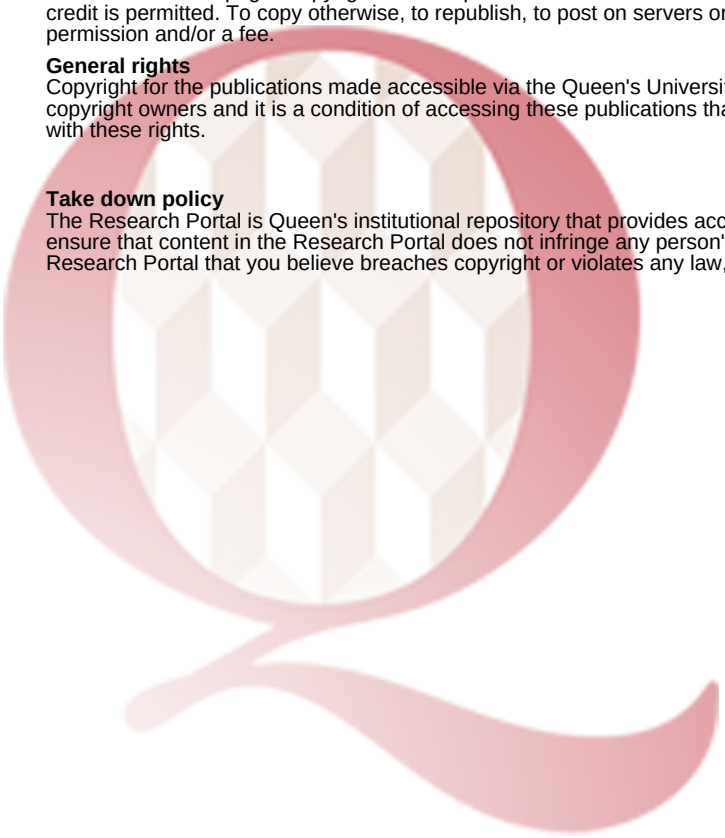
Mitigating the Impact of Faults in Unreliable Memories for Error-Resilient Applications

Ganapathy, S., Karakonstantis, G., Teman, A., & Burg, A. (2015). Mitigating the Impact of Faults in Unreliable Memories for Error-Resilient Applications. In Proceedings of the 52nd Annual Design Automation Conference. [102] ACM. DOI: 10.1145/2744769.2744871

Published in:
Proceedings of the 52nd Annual Design Automation Conference

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal



Mitigating the Impact of Faults in Unreliable Memories for Error-Resilient Applications

Shrikanth Ganapathy[†], Georgios Karakonstantis[‡], Adam Teman[†], and Andreas Burg[†]
[†]Telecommunications Circuits Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland
[‡] High Performance and Distributed Computing, Queen's University Belfast, United Kingdom
[†]{shrikanth.ganapathy, adam.teman, andreas.burg}@epfl.ch [‡]g.karakonstantis@qub.ac.uk

ABSTRACT

Inherently error-resilient applications in areas such as signal processing, machine learning and data analytics provide opportunities for relaxing reliability requirements, and thereby reducing the overhead incurred by conventional error correction schemes. In this paper, we exploit the tolerable imprecision of such applications by designing an energy-efficient fault-mitigation scheme for unreliable data memories to meet target yield. The proposed approach uses a bit-shuffling mechanism to isolate faults into bit locations with lower significance. This skews the bit-error distribution towards the low order bits, substantially limiting the output error magnitude. By controlling the granularity of the shuffling, the proposed technique enables trading-off quality for power, area, and timing overhead. Compared to error-correction codes, this can reduce the overhead by as much as 83% in read power, 77% in read access time, and 89% in area, when applied to various data mining applications in 28 nm process technology.

Categories and Subject Descriptors
B.8.1 [Reliability, Testing and Fault-Tolerance]
General Terms
Design, Management, Reliability
Keywords
Significance-driven computing, Priority-ECC, Bit-shuffling, Approximate Computing, Error-resilient Applications, Unreliable Memory, Error Correction, SRAM

1. INTRODUCTION

The demand for on-chip memory capacity continues to increase today with embedded memories already occupying 40%-60% of the die area in chip multiprocessors and expected to exceed 70% by 2017 [1]. While the aggressive shrinking of transistor sizes has been one of the primary facilitators of this trend, the consequent increase in parametric variations in deep sub-micron technologies has introduced new challenges, especially when designing high-density and energy-efficient memories. In particular, variations in transistor characteristics significantly reduce the noise margins of memory cells and the timing margins of memory blocks, leading to an increased number of memory failures [2]. When employing mechanisms such as voltage scaling for power savings, the noise margin of the cell worsens and the failure probability increases. In such scenarios, conventional approaches to ensure reliable operation attempt to correct every *single* failure through the use of redundant hardware [3]. One popular solution is the use of error correcting codes (ECCs) that can detect and correct one or more faults, depending on the number of extra parity bits added to each data word [4]. However, the addition of extra bits, along with the logic required for fault detection and correction, incurs significant energy, area, and delay overhead. Moreover, such techniques lead to extensive over-design, as they are targeted at maintaining fault-free functionality even for extreme outliers that can occur in the manufacturing process. The corresponding design margins and resulting wasted power are expected to further increase as silicon predictability diminishes, raising further doubts about the efficiency of such design approaches [3, 5].

The reality described above has led to the quest for alternative design strategies and to the promising *approximate computing* paradigm, which exploits the error resilient nature of many applications to relax the design constraints and to save power [3, 5–8]. This paradigm includes the development of processors and software that may not always produce 100% precise results, but their output fidelity is acceptable for human consumption [3, 7]. In this way, significant power reduction can be achieved, and the overhead required for fault tolerance mechanisms can be limited. When designing memories for error-resilient applications, various opportunities for resource saving emerge, such as restricting the use of robust and power hungry bit-cells to protect only the bits that play a more significant role in shaping the output quality [3, 9–11]. A recently proposed approach, known as *priority-based ECC* (P-ECC), limits the overhead by applying ECC only to the higher order bits [4, 12]. The proposed approach is targeted towards platforms used for multimedia applications. As P-ECC attempts to correct every failure (like ECC) on smaller words, the overheads incurred are still significant and therefore, further opportunities for overhead reduction exist.

In this paper, we propose an alternative system-level approach to reduce the impact of faults on the output quality of an application, quantified in terms of a suitable quality metric. Rather than correcting the faults (like ECC), the proposed approach minimizes the magnitude in error caused due to a faulty cell. This is ensured by placing bits of lower significance into the faulty cells. This is implemented through a hardware mechanism that circular-shifts the data word (upon every write) such that bits of lower significance (LSB) are stored in the faulty cells leading to a tolerable loss in output quality. The number of shifts required is determined apriori using a built-in self test (BIST) structure. The proposed scheme was implemented in a 28 nm

DAC'15, June 7-11, San Francisco, USA

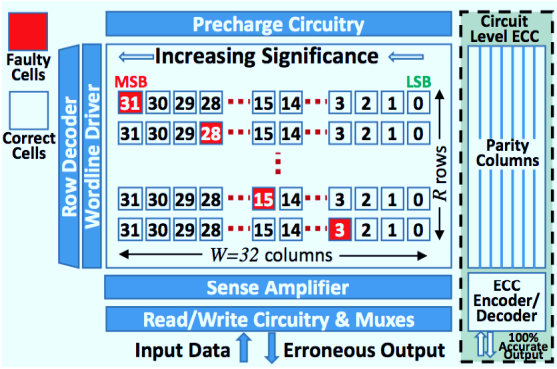


Figure 1: 32-bit SRAM array with ECC

FD-SOI process and compared with traditional ECC and P-ECC for a variety of data mining applications, showing beneficial trade-offs in terms of power, area, and delay at an acceptable quality loss.

Contributions: Our specific contributions are:

- A bit-shuffling error-mitigation scheme that takes into account the properties of binary representations that attribute different significance to each bit. The basic idea of the proposed scheme is to shift the bit-segments of the original stored word, in order to store the more significant bits in non-faulty bit-cells, while allowing less significant bits to be stored in faulty bit-cells. By doing so, the bit-error distribution is skewed towards the lower significance bits, substantially limiting the quality loss.
- Application of the proposed scheme to a variety of data mining and classification algorithms and evaluation of its efficacy in limiting the quality loss, quantified in terms of a suitable quality metric. The proposed scheme is compared to P-ECC, which we also apply to the studied applications. This analysis also further enhances the understanding of the efficacy of P-ECC since this method has so far only been applied to multimedia systems.
- Relaxation of the traditional zero-failure yield criterion to allow a limited number of faults/limited quality degradation, leading to a redefined test criterion, which is evaluated for our scheme.
- Implementation of the proposed scheme in a 28 nm FD-SOI process node, integrated with a 32-bit word memory block.
- Evaluation of the power, delay, and area overhead vs. loss of quality achieved by applying the proposed bit-shuffling scheme with different sizes of bit-segments, and comparison with the traditional ECC and P-ECC schemes.

The rest of the paper is organized as follows. Section 2 describes the required background and traditional ECC approaches. Section 3 presents the proposed approach and Section 4 discusses an analytical formulation of its benefits in terms of yield and output quality. Section 5 analyses the trade-offs between energy savings and output quality and compares the obtained results to existing schemes. Section 6 concludes the paper.

2. BACKGROUND - MOTIVATION

In general, a static random-access memory (SRAM) is a 2-dimensional array consisting of $M = R \times W$ bit-cells organized in R rows and W columns. In many SRAM architectures, each row stores a W-bit vector, referred to as a *word*. These words can be accessed through peripheral memory circuits, which include address decoders, precharge circuits,

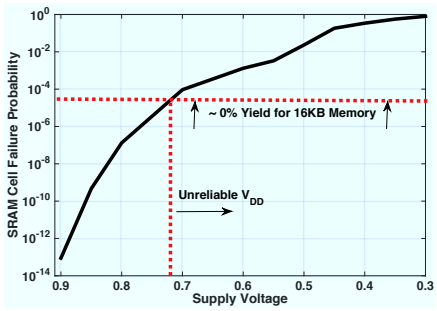


Figure 2: SRAM cell failure probability under V_{DD} scaling in 28nm process technology.

column multiplexers, and sense amplifiers, as depicted in Fig. 1 for an array with 32-bit words.

As discussed previously, while technology scaling may have helped meet increased memory density requirements, it also has increased the sensitivity of transistors to parametric variations. Such variations can impair the functionality of a bit-cell, rendering it unable to correctly store '0' and '1' levels and/or read out the stored level within the required time, leading to a number of bit-cell failures, N. The probability of having a certain number of failures $\Pr(N = n)$ depends on the memory size and the cell failure probability, P_{cell}, which is a function of various transistor characteristics and operating conditions.

Using an in-house developed framework based on SPICE-level simulations and hypersphere sampling [13] we estimated the total cell failure probability of a classical six-transistor (6T) SRAM cell in a 28nm process node under various supply voltages (V_{DD}), as depicted in Fig. 2. While V_{DD} scaling is one of the primary methods for power reduction [3], Fig. 2 shows that this results in a rapid increase in P_{cell}. It is important to note that once a memory array is manufactured, the number and location of any variation-induced bit-cell failures is persistent. In addition, in the presence of process variations, bit-cell failures caused by voltage scaling follow the *fault inclusion property*, i.e., bit-cells that fail at a given V_{DD} will fail for all lower V_{DD} [14]. Post-manufacturing tests are used to determine if the memory sample has any failure or not. As defined by the traditional yield criterion, all memory samples with 1 or more number of failures are rejected to ensure 100% data integrity. However, as P_{cell} increases under technology and V_{DD} scaling, such a conservative criterion provides a low yield (defined as $Y = (1 - P_{\text{cell}})^M$ for a memory of M total bit-cells). For example, the yield approaches zero for a 16 KB memory operating at 0.73 V, as shown in Fig. 2.

Various techniques are used to limit the yield loss, such as redundant rows and/or columns within each die that replace any row/column with a faulty cell. However, as the number of failures increases, the number of redundant rows/columns required to replace every faulty row/column increases tremendously [15]. The economics governing redundancy based schemes make it an unviable option when considering worst-case process variations.

This has led to the use of other schemes, such as ECC, for maintaining high yield under the impact of parametric failures due to process variations and voltage scaling [4, 15]. In ECC, each W-bit word is transformed into a C-bit codeword ($C > W$) by adding $c = W - C$ parity bits upon every write to the memory. This codeword is then compared (using specialised decoders) against a new codeword generated during a read to detect and/or correct any failures that might have occured between the last write and the subsequent read. In

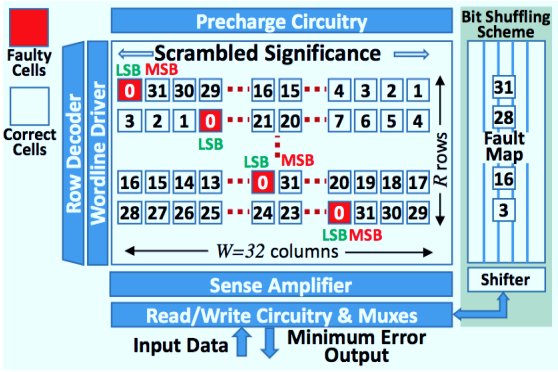


Figure 3: Proposed bit-shuffling scheme.

general, there are various classes of linear-time encodable and decodable ECCs, but one of the most popular, is the single error correction, double error detection (SECDED) Hamming code (which will also be used for comparison in this paper). For a 32-bit data word, $c = 7$ parity bits are needed for SECDED ECC, in what is known as an H(39,32) code. While such a code can ensure all words in the memory with a maximum of 1-failure can be corrected, the integration of the encoding/decoding circuitry, and storage cells for the parity bits, shown in Fig 1, results in significant area, delay, power overhead, as we will also show later.

The guarantee of 100% accurate operation in safety critical applications may justify the overhead incurred by the use of ECC. However, for many applications, the reliability constraints, and thus the overhead, can often be relaxed due to their inherent error resilience [8]. It has been observed, for instance, that in multimedia and communication applications, only errors in certain bits or types of data may lead to non-acceptable quality loss. Such an observation has led to unequal error protection schemes, according to which larger bit-cells and more complex ECC are used to guarantee the accuracy of critical bits, while leaving the rest of the bits unprotected or encoded with less complex ECC [4, 12, 16]. For instance, in [12], it was shown that by protecting only the 32-higher order bits of each 64-bit memory word can limit the quality loss in terms of peak-signal-to-noise-ratio (PSNR) in an H.264 video processing system, even under 30% voltage scaling, while incurring at least 50% lesser overhead when compared to traditional ECC schemes.

Although such schemes may reduce the overhead of traditional ECC, they primarily target multimedia applications, and their compatibility with other applications still remains unexplored. In addition, such schemes are still based on ECC, even if they restrict its use to only part of the memory words. By doing so they provide only partial protection of the overall word and thus errors in LSBs might get neglected leading potentially to non-acceptable quality loss. Such issues motivated us to introduce an alternative and generic fault mitigation scheme that is not based on ECC and that fully exploits the properties of common binary formats (which attribute higher significance to MSBs) for tackling any fault in each entire memory word.

3. PROPOSED SCHEME

Taking into account the increasing significance as we move from lower to higher order bits, we propose a significance-driven error mitigation approach that reduces the impact of faulty bit-cells to a level that is acceptable by the target application. The basic idea of this approach is to shift segments of the original data words, such that the data bits with higher significance are stored in non-faulty bitcells, thereby ensuring that faulty bitcells can only store data bits of lower significance. By modifying the number of bits that comprise a shifted segment, the designer can trade-off quality for power, delay, and area.

We use a simple example to illustrate the proposed scheme. Let us assume that 32-bit integers in 2's complement representation are stored in a 32-bit wide standard memory architecture with several faulty bits, as illustrated in Fig. 1. Without any correction mechanism, the output error magnitude would be as high as 2^{32}, for the top word in the illustrated memory. Although integrating an H(39,32) SECDED ECC scheme would correct such faults, this requires the addition of 7 extra columns for storing the parity bits and complex encoding/decoding logic, which adds approximately 13 gate delays to the read access time of the memory [17]. In addition, the power and area overhead of adding such a scheme can be significant, depending on the size of the memory [4].

The same faulty array is illustrated in Fig. 3 with the proposed bit-shuffling scheme integrated instead of ECC. In this example, a 5 column wide fault-map look-up table (FM-LUT) is used. Each entry in this FM-LUT indicates the amount of right-circular shift that needs to be applied to ensure that only the least significant segment of a word stored on a row with a faulty cell will be affected by the potential error. In the particular example (of a 32-bit wide memory), the 5-bit wide FM-LUT allows application of shifting at a single-bit granularity. The overall procedure for applying the proposed scheme consists of two steps. First, the location of the faulty cell in each row/word is detected during BIST and a shifting value, $x_{\mathrm{FM}}(r)$, is recorded in the FM-LUT for each row r. Second, when writing to the memory, the shifting value for the selected word address is read out and a right-circular shift is applied to the data word according to this value. By doing so, the least significant segment of the data word is moved to the location of the faulty cell before storing the data. In the example of Fig. 3, the LSB (due to the assumed single-bit segment size) of the top word is shifted-right by 31 positions and stored in bit-position 31 of the memory word. During readout, the FM-LUT is checked again, this time initiating a left-circular shift of the readout value, thereby restoring the original bit locations. In this single-bit segment case, a maximal error-magnitude of 2^0=1 is obtained for each row under the assumption of a single fault per word

Integration of the bit-shuffling scheme into a standard memory architecture requires the addition of the FM-LUT and a shifter. The number of bits per row in each FM-LUT entry, n_{FM}, sets the segment size S of the bit-shuffling scheme, which is given by:

$$S = W/2^{n_{\mathrm{FM}}} \quad (1)$$

with $1 \leq n_{\mathrm{FM}} \leq \lceil log_2 W \rceil$ for a W-bit word. The segment size should be selected according to the residual error tolerated by the application. Fig. 4 displays the error magnitude per faulty bit position of all n_{FM} implementation possibilities for a 32-bit memory storing 2's complement integers. While the maximum error magnitude for n_{FM}=5 is 2^0, the area, delay, power overhead can be reduced by choosing lower values of n_{FM}. For all cases, the worst-case error magnitude is bounded by 2^{S-1}, based on the maximum distance of the faulty bit from the LSB of the shifted segment.

The locations of the faulty bits, as previously mentioned, are determined through the memory BIST, which can be executed either during post-fabrication testing or during power-on startup testing (POST). The process of performing this routine every time the system is booted provides the advantage of tracking potential failures induced by temporal degradation (i.e., due to aging). According to the fault locations, the shifting value, $x_{\mathrm{FM}}(r)$, is stored in the FM-LUT

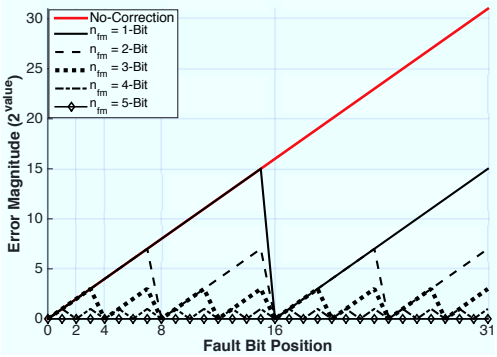


Figure 4: Error-magnitude of failures per faulty bit position for all options of FM sizes, assuming a 32-bit integer in case of 2's complement representation.

for each row r. This defines the circular shift rotation magnitude $T(r)$ that needs to be applied to the data word, as:

$$T(r) = S \cdot (2^{n_{\mathrm{FM}}} - x_{\mathrm{FM}}(r)). \quad (2)$$

For example, in the case of the faulty array of Fig. 3, with W=32 and n_{FM}=5, the bottom word has a failure in its third bit. Therefore, T(bottom row)=29, and the data word is circularly shifted right by 29 positions, such that the LSB is stored in the faulty position.

4. YIELD CRITERION AND IMPACT

As discussed in Section 2, yield is traditionally defined as the percentage of memories that are fault-free (zero failures). However, to depart from the paradigm of 100% reliable operation, the yield criterion has to be modified; memories with up to a certain number of failures still qualify, as long as the failing bit-cells have no relevant impact on the quality of service of the considered application [8, 10, 11]. As the quality metric and the error tolerance limits vary across applications, this type of yield criterion will have to be set on a per-application basis. Furthermore, this criterion needs to account for any technique integrated on the chip to mitigate the quality-impact of failing bit-cells.

To this end, we propose a qualitative cost function that considers not only the number of failures, but also the specific application-error magnitude of each tested die. To illustrate this idea, let us assume that the impact of faults in an unreliable memory with N failures is quantified in terms of a quality metric Q. The joint probability that a memory has quality Q=q with N=n failures is given by:

$$\mathbf{Pr}(N = n, Q = q) = \mathbf{Pr}(Q = q \mid N = n) \cdot \mathbf{Pr}(N = n), \quad (3)$$

where the probability of a memory of size M having exactly n failures is given by:

$$\mathbf{Pr}(N = n) = \binom{M}{n} P_{cell}^{n} * (1 - P_{cell})^{M-n}, \quad (4)$$

where P_{cell} is the SRAM bit-cell failure probability. Since we know that memories with a certain bit-cell failure probability can result in memory samples with different numbers of failures, the probability of getting a die with quality q over the range of up-to n failures is:

$$\mathbf{Pr}(Q = q) = \sum_{i=1}^{n} \mathbf{Pr}(N = i, Q = q). \quad (5)$$

Equation (5) provides an estimate of the probability of the memory having a specific quality, as defined by the quality function. Therefore, (5) measures the yield of a memory that satisfies a given quality constraint.

The chosen quality function should be representative of an appropriate quality metric of the underlying application, and should show good correlation with the output quality of

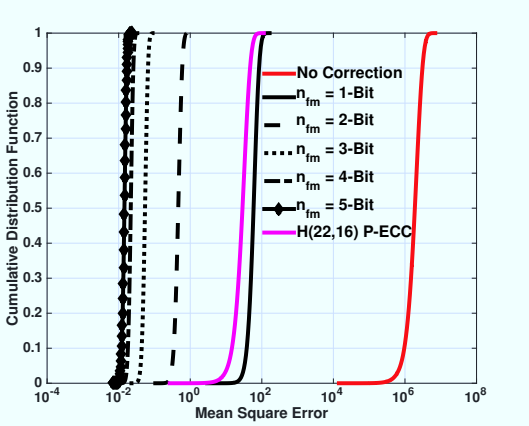


Figure 5: CDF versus measured MSE for a memory using i) no protection, ii) priority-based ECC, and iii) all segment size options of the proposed bit-shuffling scheme

the application under the impact of failures. If only applied locally to the memory, the quality function must still properly capture the impact of a fault at a given bit location on the output error magnitude. The mean square error (MSE), computed over the error magnitude of all words in the memory, often provides a good local estimate of the achievable final output quality. Using the local MSE as an indicator for the application dependent output quality metric has two main advantages: a) Both the locations of the failures and the corresponding MSE can be rapidly determined at test time, avoiding the need for executing a large number of test-time simulations to determine if a specific memory sample satisfies all quality constraints. b) The MSE still provides a means to differentiate between memories that have the same number of failures, but different output quality characteristics by virtue of the location of the failures. For the previous example of a 2's complement integer, the magnitude of the impact of a failure at bit position b is given by 2^b. The overall error magnitude of an $R \times W$ memory array storing 2's complement integers can then be defined by the MSE, as given by:

$$MSE = \frac{1}{R} \sum_{i=1}^{N_{\mathrm{failures}}} (2^{b_i})^2, \quad 0 \le b_i < W, \quad (6)$$

where b_i is the location of the i^{th} failure and N_{failures} is the number of failures in the specific memory sample.

Fig. 5 shows the cumulative density function (CDF) of the MSE for a 16kB memory with a cell failure probability of $5 \cdot 10^{-6}$ obtained by evaluating equations (3) to (6). The CDF was obtained for both an unprotected memory and a memory with a P-ECC scheme using an H(22,16) SECDED code for protecting the 16 MSBs as well as for memories utilizing the proposed bit-shuffling scheme with various segment sizes. For each number of failures, from 1 to 150, random bit-flips were injected considering in total T_{run}=10^7 Monte Carlo (MC) generated memory samples, and the resulting MSE was computed for each sample. The fault injection was performed by generating maps of random bit-flip locations for each failure count. The number of samples per failure count n is determined by $\mathbf{Pr}(N{=}n){\cdot}\mathrm{T_{run}}$. The results show that the proposed approach leads to a minimum 30× reduction in MSE that must be tolerated to achieve a given target yield, even for the n_{FM}=1 case. This can directly be translated into yield improvement, as for example, by setting the maximum target MSE to $MSE < 10^6$, the proposed scheme achieves 99.9999% yield for the n_{FM}=1 configuration, as opposed to a yield of <6% for the unprotected

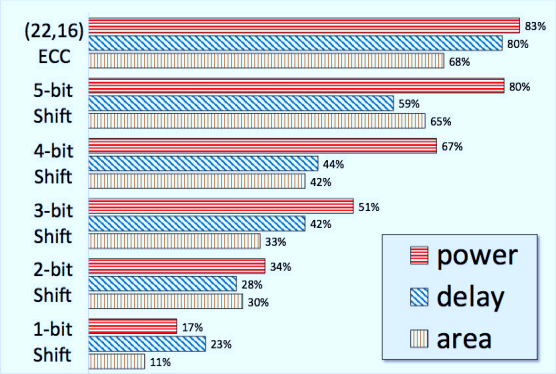


Figure 6: Overhead comparison of the proposed scheme using n_{FM}=1, ..., 5 and a H(22,16) P-ECC scheme with a H(39,32) SECDED ECC scheme

memory (based on no correction mechanism). In addition, we observe that the proposed scheme also outperforms P-ECC in most cases, providing a significantly lower MSE for a given yield target with n_{FM}=2, ..., 5.

5. EVALUATION METHODOLOGY AND RESULTS

Evaluation of the proposed scheme was performed in two parts. First, bit-shuffling, ECC, and P-ECC schemes were implemented for overhead comparison. Second, a simulation framework for evaluating the output quality was developed for running representative benchmarks on real-world datasets, and used to evaluate and compare the quality degradation due to bit-shuffling and P-ECC schemes [4].

5.1 Hardware Overhead Comparison

In order to quantify the reduction in overhead by using the proposed *error-mitigation* technique as compared to alternative *error-correction* solutions, a power, delay, and area comparison was carried out versus a classical H(39,32) SECDED ECC and an H(22,16) P-ECC scheme. The encoder/decoder blocks of the two ECC techniques and all segmenting options for a 32-bit wide memory (n_{FM}=1, ..., 5) were implemented in a 28 nm FD-SOI technology, synthesized with Synopsys Design Compiler, and placed and routed with Cadence SoC Encounter. The resulting layouts were evaluated for power consumption, using a value change dump (VCD) based flow in the Cadence Encounter Power System. Power and area overheads due to the addition of extra columns (parity bits and LUTs) were estimated based on SRAM macros available in this technology. Note that this comparison only takes into account the readout path, as for the considered applications, write operations are not on the critical path and are carried out much less frequently than reads. In addition, the LUTs are implemented as entire bit columns in the array to demonstrate the achievable saving through the most straightforward realization of the proposed technique. However, the LUT could be realized with, for example, a content-addressable memory (CAM) or register file, to provide much less overhead, especially in terms of write latency, which in the case of bit-shuffling, requires a read prior to a write.

Fig. 6 presents the comparison of the read power, read delay, and area of each of the aforementioned techniques, relative to the overhead required by the H(39,32) SECDED ECC. The proposed scheme provides an advantage over both ECC-based methods in all design aspects. In terms of read power overhead, the bit-shuffling techniques save 20%–83% depending on the bit-segment size, while the read delay and area overhead are reduced by 41%–77% and 32%–89%, respectively, as compared to the overheads required by SECDED ECC. When considering the P-ECC, the bit-shifting technique provides as much as 59%, 64%, and 57% reduction in read power, read delay, and area overhead, respectively.

Table 1: Evaluation Applications and Datasets

Class	Algorithm	Dataset	Metric
Regression	Elasticnet	Wine Quality [18]	R^2
Dimensionality Reduction	Principal Component Analysis (PCA)	Madelon [19]	Explained Variance
Classification	K-Nearest Neighbors (KNN)	Activity Recognition [20]	Score

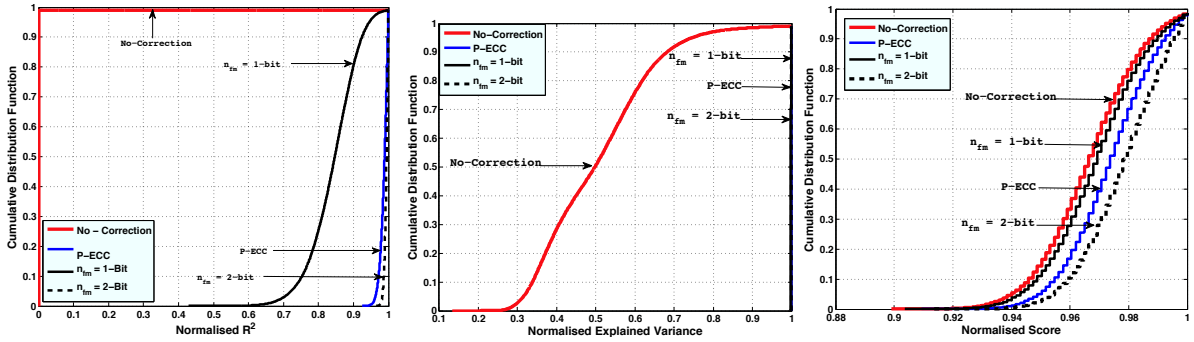
5.2 Impact on Quality

In order to evaluate the efficacy of the proposed scheme in limiting the quality loss at the application level under potential memory failures, we applied it to various data mining and classification applications. Although our proposed scheme is developed keeping in mind its applicability to a wide range of error-resilient applications, for the sake of brevity, we will limit our discussion of the impact to only three very widely-used algorithms in these domains. In particular, we consider: i) *Elasticnet* - used for data regression, ii) *Principal Component Analysis* (PCA) - used for dimensionality reduction, and iii) *K-Nearest Neighbors* (KNN) - used for data classification. These popular algorithms have yet to be studied extensively in the context of fault-mitigation, especially in the case of the P-ECC schemes.

All benchmark algorithms under consideration require a set of training data, which is processed initially by each algorithm for developing the actual model. The developed model is then validated by executing a set of test data and evaluating a quality metric suitable for each target application, as summarized in Table 1. For the Elasticnet benchmark, we used a wine taste preference dataset and use R^2 as an indicator o fthe goodness of fit [18]. For the PCA benchmark, a synthetic dataset from the 2003 NIPS feature selection challenge was used [19]. *Explained variance* is the measured quality metric. The classification (KNN) benchmark performs human activity recognition based on accelerometer readings from the dataset [20]. The benchmarks were developed using the open-source Scikit-Learn framework [21] and the datasets were obtained from the UCI machine learning repository [22].

A software simulation framework was developed, for storing and processing the input training dataset of each benchmark and evaluating the resulting output quality. For each benchmark, the dataset is partitioned into training and testing inputs (0.8:0.2 ratio), and a functional model of a 16 KB memory is used to inject bit-flips, according to the random fault maps based on the assumed P_{cell}, as described in Section 4. In particular for each number of failures N=1, ..., N_{max}, we determined the probability that a memory sample will have N bit-cell failures using (4) for a given P_{cell}, where 99% of the memories have no more than N_{max} failures. For each N, we generated 500 MC samples of random fault maps and executed the benchmarks on the functional faulty memory to determine the output quality.

Fig. 7 plots the resulting cumulative distribution function (CDF) of the specific quality metric for each benchmark for a 16kB memory with a P_{cell} of 10^{-3}. The plots assume that the small number of samples with more than one error per word are discarded, such that H(39,32) ECC provides error-free operation, and therefore, a perfect normalized metric of



(a) Elasticnet (R^2). (b) PCA (Explained Variance) (c) KNN (Score)

Figure 7: Measured CDF of quality for three different applications under the impact of memory failures with a.)no protection b.) P-ECC c.) bit-shuffling

1 for each benchmark in fault-free cases. The CDF indicates the yield as a function of the quality obtained when executing the specific benchmarks. The CDF of the metric is then plotted for the use of: i) no error protection; ii) bit-shuffling with n_{FM}=1; iii) bit-shuffling with n_{FM}=2; and iv) H(22,16) P-ECC on the MSB bits. For each benchmark, the n_{FM}=2 bit-shuffling scheme already provides better error-mitigation than the P-ECC, and therefore, higher values of n_{FM} are not plotted. In the case of the Elasticnet benchmark, while without any correction, the R^2 metric is extremely low for virtually all samples, even the addition of a single bit LUT provides very good results, clearly surpassing the P-ECC technique. As shown previously, this also comes with significantly less overhead.

6. CONCLUSION

A fault mitigation scheme that departs from the conventional 100% error detection and correction mechanisms, and instead reduces the impact of errors on output quality by utilizing the relaxed reliability constraints enabled by error resilient applications is presented. By exploiting the fact that all popular binary representations attribute different significance to each bit location, the proposed scheme shifts the stored bits such that only less significant bits are affected by any fault. By doing so, the loss in output quality is kept low, as quantified in terms of suitable metrics and shown for a variety of data mining and classification algorithms. Our results in a 28 nm process node indicate that the adoption of the proposed scheme, instead of traditional ECC, can reduce the read power, read delay, and area overhead by as much as 83%, 77% and 89%, respectively. Overall, the proposed mechanism provides a low-cost alternative to traditional fault-tolerant schemes and can be used to exploit the properties of a variety of error-resilient applications for allowing operation at scaled voltages and advanced technology nodes.

7. ACKNOWLEDGEMENT

The authors acknowledge financial support for this research from Huawei.

8. REFERENCES

[1] "ITRS - 2013 edition," 2013. [Online]. Available: http://www.itrs.net

[2] S. Mukhopadhyay *et al.*, "Modeling of Failure Probability and Statistical Design of SRAM Array for Yield Enhancement in Nanoscaled CMOS," *IEEE TCAD*, 2005.

[3] S. Bhunia *et al.*, *Low-Power Variation-Tolerant Design in Nanometer Silicon.* Springer, 2010.

[4] Y. Emre *et al.*, "Techniques for compensating memory errors in JPEG2000," *IEEE Trans. VLSI Syst.*, vol. 21.

[5] P. Gupta *et al.*, "Underdesigned and opportunistic computing in presence of hardware variability," *IEEE TCAD*, vol. 32, no. 1, pp. 8–23, 2013.

[6] J. Lucas *et al.*, "Sparkk: Quality-scalable approximate storage in DRAM," in *The Memory Forum*, June 2014.

[7] J. Henkel *et al.*, "Multi-layer dependability: From microarchitecture to application level," in *DAC'14*, 2014.

[8] V. K. Chippa *et al.*, "Analysis and characterization of inherent application resilience for approximate computing," in *DAC '13*, 2013.

[9] A. Sampson *et al.*, "Approximate storage in solid-state memories," in *ISM '13*, 2013, pp. 25–36.

[10] V. Kleeberger *et al.*, "A cross-layer technology-based study of how memory errors impact system resilience," *IEEE Micro*, 2013.

[11] G. Karakonstantis *et al.*, "On the exploitation of the inherent error resilience of wireless systems under unreliable silicon," in *DAC'12*, Jun. 2012, pp. 510–515.

[12] I. Lee *et al.*, "Priority based ECC for embedded SRAM memories in H.264 system," *S. P. Systems*, vol. 73.

[13] T. Date *et al.*, "Robust importance sampling for efficient SRAM yield analysis," in *IEEE ISQED*.

[14] M. Gottscho *et al.*, "Power/capacity scaling: Energy savings with simple fault-tolerant caches," in *DAC'14*, 2014.

[15] Z. Shi-Ting *et al.*, "Minimizing Total Area of Low-Voltage SRAM Arrays Through Joint Optimization of Cell Size, Redundancy, and ECC," in *ICCD'10*.

[16] F. Frustaci *et al.*, "13.8 a 32kb sram for error-free and error-tolerant applications with dynamic energy-quality management in 28nm cmos," in *ISSCC'14*.

[17] D. Rossi *et al.*, "Error Correcting Code Analysis for Cache Memory High Reliability and Performance," in *DATE'11*.

[18] P. Cortez *et al.*, "Modeling wine preferences by data mining from physicochemical properties," *D. S. Systems*.

[19] I. Guyon *et al.*, "Competitive baseline methods set new standards for the NIPS 2003 feature selection benchmark."

[20] P. Casale *et al.*, "Personalization and user verification in wearable systems using biometric walking patterns," *Personal and Ubiquitous Computing*.

[21] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Mach. Learning Research*, vol. 12, 2011.