International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

A NEW CLASS OF SINGLE BURST ERROR CORRECTING CODES BY USING PARALLEL DECODING

[1] B Eramma, [2] D Venu Gopal, [3] H Yirendra Nath, [4] G Sanjay Kumar, [5] K Vishnu Vardhan
[1] Assistant Professor [2,3,4,5] B.Tech Scholar,
[1,2,3,4,5] Department of Electronics and Communication Engineering
G Pullaiah College of Engineering and Technology, Nandikotkur Rd,near Venkayapalle,Pasupala Village,Kurnool,Andhra Pradesh 518002, India

ABSTRACT: With technology scaling, burst errors or clustered errors are becoming increasingly common in different types of memories. Multiple bit upsets due to particle strikes, write disturbance errors, and magnetic field coupling are a few of the mechanisms which cause clustered errors. In this project, A new class of single burst error correcting codes are presented which correct a single burst of any size *b*within a codeword. A code construction methodology is presented which enables us to construct the proposed scheme from existing codes, e.g.,Hamming codes. A new single step decoding methodology for the proposed class of codes is also presented which enables faster decoding. Different code constructions using Hamming codes, and BCH codes have been presented in this project and a comparison is made with existing schemes in terms of decoding complexity and data redundancy. The proposed scheme in all cases reduce the decoder complexity for little to no increase in data redundancy, specifically for higher burst error sizes. In this project, A new class of single burst error correcting codes are presented with a parallel decoding scheme. The proposed parallel decoding scheme enables high speed decoding. This is particularly useful for memories whose performance is sensitive to read or access latencies. A new construction methodology is presented which enables the proposed codes to be derived from already existing codes so that a single burst error can be corrected.

1.INTRODUCTION

Technology scaling has been leading to smaller and smaller device geometries over the years. This has given rise to a host of different problems with both the established memory technologies as well as the newer forms of emerging memory technologies. One such form of error is a burst error which is becoming more and more prevalent in several types of memories due to shrinking feature size. Consider the case of static random-access memory (SRAM). Soft errors caused by radiation poses a significant reliability concern for SRAMs [1]. With technology scaling, the susceptibility of SRAMs to soft-errors has significantly increased as well [2]. In current nano scale technology nodes, device geometries are small, and with technology scaling, devices are getting smaller. Thus, a particle strike might affect more than one cell causing a multiple bit upset (MBU) [3]. The smaller the device geometries, the larger the number of cells that are affected by a single particle strike. A *b*-bit burst error caused by such a particle strike can cause multiple bits to be flipped within the *b*-bit burst window.

Dynamic random-access memory (DRAM) also suffers from a similar problem [4]. The problem arises due to the small physical dimensions brought about by technology scaling. Although it enables to increase the memory capacity of a chip, it also enables the ease with which near-by or adjacent DRAM cells interact with each other. Thus, accessing a memory cell causes a disturbance in the neighboring memory cells causing their charge to leak

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

into the cell or away from it. With enough accesses, it is possible to flip the neighboring cell's currently held value.

Thus, at any given point a memory access can potentially cause a single *b*-bit burst error in the vicinity of the cell being accessed, where *b*is the size of the burst.

The underlying principle of error correcting codes addressing these issues is that the codes should be able to correct a single burst of error. It is possible that not all bits or symbols change within the burst window *b*. Thus, codes aimed towards addressing these issues should be able to correct all possible error combinations within a *b*-bit burst window regardless of the position of burst error.

In this project, a new class of single burst error correcting codes is presented with a parallel decoding scheme. The proposed parallel decoding scheme enables high speed decoding. This is particularly useful for memories whose performance is sensitive to read or access latencies. A new construction methodology is presented which enables the proposed codes to be derived from already existing codes so that a single burst error can be corrected. Preliminary results for the proposed scheme were presented in [5] and [6]. A key feature of the proposed class of codes is that it leads to significant reduction in area of the decoding circuit specifically as the burst size being corrected *b* increases.

2.LITERATURE SURVEY

Soft errors in advanced computer systems by R. Baumann,

As the dimensions and operating voltages of computer electronics shrink to satisfy consumers' insatiable demand for higher density, greater functionality, and lower power consumption, sensitivity to radiation increases dramatically. In terrestrial applications, the predominant radiation issue is the soft error, whereby a single radiation event causes a data bit stored in a device to be corrupted until new data is written to that device. This article comprehensively analyzes soft-error sensitivity in modern systems and shows it to be application dependent. The discussion covers ground-level radiation mechanisms that have the most serious impact on circuit operation along with the effect of technology scaling on soft-error rates in memory and logic.

Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors by Yoongu Kim; Ross Daly; Jeremie Kim; Chris Fallin; Ji Hye Lee; Donghyuk Lee

Memory isolation is a key property of a reliable and secure computing system-an access to one memory address should not have unintended side effects on data stored in other addresses. However, as DRAM process technology scales down to smaller dimensions, it becomes more difficult to prevent DRAM cells from electrically interacting with each other. In this paper, we expose the vulnerability of commodity DRAM chips to disturbance errors. By reading from the same address in DRAM, we show that it is possible to corrupt data in nearby addresses. More specifically, activating the same row in DRAM corrupts data in nearby rows. We demonstrate this phenomenon on Intel and AMD systems using a malicious program that generates many DRAM accesses. We induce errors in most DRAM modules (110 out of 129) from three major DRAM manufacturers. From this we conclude that many deployed systems are likely to be at risk. We identify the root cause of disturbance errors as the repeated toggling of a DRAM row's wordline, which stresses inter-cell coupling effects that accelerate charge leakage from nearby rows. We provide an extensive characterization study of disturbance errors and their behavior using an FPGA-based testing platform. Among our key findings, we show that (i) it takes as few as 139K accesses to induce an error and (ii) up to one in every 1.7K cells is susceptible to

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

errors. After examining various potential ways of addressing the problem, we propose a low-overhead solution to prevent the errors.

Systematic b-adjacent symbol error correcting reed-solomon codes with parallel decoding by Abhishek Das; Nur A. Touba

With technology scaling, the probability of write disturbances affecting neighboring memory cells in nonvolatile memories is increasing. Multilevel cell (MLC) phase change memories (PCM) specifically suffer from such errors which affects multiple adjacent memory cells. Reed Solomon (RS) codes offer good error protection since they can correct multi-bit symbols at a time. But beyond single symbol error correction, the decoding complexity as well as the decoding latency is very high. This paper proposes a systematic b-adjacent symbol error correcting code based on Reed-Solomon codes with a low latency and low complexity parallel one step decoding scheme. A general code construction methodology is presented which can correct any errors within b-adjacent symbols. The proposed codes are compared to existing adjacent symbol error correcting Reed-Solomon codes, and it is shown that the proposed codes achieve better decoder latency. The proposed codes are also shown to achieve much better redundancy compared to symbol error correcting orthogonal Latin square (OLS) codes.

Siva Sreeramdas, S.Asif Hussain and "Dr.M.N.Giri Prasad proposed on Secure Transmission for Nano-Memories using EG-LDPC"

Memory cells have been protected from soft errors for more than a decade; due to the increase in soft error rate in logic circuits, the encoder and decoder circuitry around the memory blocks have become susceptible to soft errors as well and must also be protected. Here introducing a new approach to design faultsecure encoder and decoder circuitry for memory designs. The key novel contribution of this paper is identifying and defining a new class of error-correcting codes whose redundancy makes the design of fault-secure detectors (FSD) particularly simple and further quantify the importance of protecting encoder and decoder circuitry against transient errors. By using that Euclidean Geometry Low-Density Parity-Check (EG-LDPC) codes have the fault-secure detector capability. Using some of the smaller EG LDPC codes, can tolerate bit or nanowire defect rates of 10% and fault rates of 10-18 upsets/device/cycle, achieving a FIT rate at or below one for the entire memory system and a memory density of 1011 bit/cm2 with nanowire pitch of 10nm for memory blocks of 10 Mb or larger. Larger EG-LDPC codes can achieve even higher reliability and lower area overhead.

3.EXISTING METHOD

OLS codes are based on the concept of Latin squares. A Latin square of size m is an $m \times m$ matrix that has permutations of the digits $0, 1, \ldots, m-1$ in both its rows and columns. Two Latin squares are orthogonal if when they are superimposed every ordered pair of elements appears only once. OLS codes are derived from OLS. These codes have $k = m2$ data bits and $2tm$ check bits, where t is the number of errors that the code can correct. For a double error correction code $t = 2$, and, therefore, $4m$ check bits, are used. As mentioned in the introduction, one advantage of OLS codes is that their construction is modular. This means that to obtain a code that can correct $t+1$ errors, simply $2m$ check bits are added to the code that can correct t errors. This can be useful to implement adaptive error correction schemes. The modular property also enables the selection of the error correction capability for a given word size. As mentioned before, OLS codes can be

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

decoded using OS-MLD as each data bit participates in exactly $2t$ check bits and each other bit participates in at most one of those check bits. This enables a simple correction when the number of bits in error is t or less. The $2t$ check bits are recomputed and a majority vote is taken. If a value of one is obtained, the bit is in error and must be corrected. Otherwise the bit is correct. As long as the number of errors is t or less, the remaining $t-1$ errors can, in the worst case, affect $t-1$ check bits.

$$H=\begin{bmatrix}
1&0&0&0&0&0&0&1&0&0&1&0&1&0&1&1&1&0&0&0&1&0&0\\
0&1&0&0&0&0&0&0&1&0&1&1&0&1&1&0&0&1&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&1&1&0&1&0&0&1\\
0&0&0&1&0&0&0&1&0&0&1&0&0&0&1&0&1&0&1&1&0&0&0\\
0&0&0&0&1&0&0&0&1&0&0&1&1&1&1&0&0&0&0&1&1&0&0\\
0&0&0&0&0&1&0&0&1&0&0&0&1&1&0&0&0&1&0&1&0&0&1\\
0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&1&0&1&1&1&0&1&1
\end{bmatrix}$$

Parity check matrix for OLS code with $k = 16$ and $t = 1$.

(1) Therefore, still a majority of $t + 1$ triggers the correction of an erroneous bit. In any case, the decoding starts by recomputing the parity check bits and checking against the stored parity check bits. The parity check matrix H for OLS codes is constructed from the OLS. As an example, the matrix for a code with $k = 16$ and 8 check bits that can correct single errors is shown in Fig. 1. The modular construction of OLS codes this matrix forms part of the H matrix for codes that can correct more errors. For example, to obtain a code that can correct two errors, eight additional rows are added to the H matrix. For an arbitrary value of $k = m2$, the H matrix for a SEC OLS code is constructed as follows:

$$H = \begin{bmatrix} M_1 & \\ & I_{2m} \\ M_2 & \end{bmatrix}$$

where $I2m$ is the identity matrix of size $2m$ and $M1$, $M2$ are matrices of size $m \times m2$. The matrix $M1$ has m ones in each row. For the r th row, the ones are at positions $(r-1) \times m + 1, (r-1) \times m + 2, \ldots (r-1) \times m + m - 1, (r-1) \times m + m$. The matrix $M2$ is constructed as follows:

$M2 = [Im\ Im \ldots Im]$.

(2)

For $m = 4$, the matrices $M1$ and $M2$ can be clearly observed in Fig. 1. The encoding matrix G is just the H matrix on which the check bits are removed

$$G = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}.$$

. (3)

In summary, the encoder takes $k = m2$ data bits (di) and computes $2tm$ parity check bits (ci) using a matrix G, which is derived from Latin squares and has the following properties.

1) Each data bit participates exactly in $2t$ parity checks.

2) A pair of data bits participates (both bits) in at most one of the parity checks.

These properties are used in the next section to discuss the proposed technique.

4.IMPLEMENTATION OF PROPOSED ARCHITECTURE

In coding theory, burst error-correcting codes employ methods of correcting burst errors, which are errors that occur in many consecutive bits rather than occurring in bits independently of each other.

Many codes have been designed to correct random errors. Sometimes, however, channels may introduce errors which are localized in a short interval. Such errors occur in a burst (called burst errors) because they occur in many consecutive bits. Examples of burst errors can be found extensively in storage mediums. These errors may be due to physical damage such as scratch on a disc or a stroke of lightning in case of wireless channels. They are not independent; they tend to be spatially concentrated. If one bit has an error, it is

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

likely that the adjacent bits could also be corrupted. The methods used to correct random errors are inefficient to correct burst errors.

Cyclic codes are defined as follows: think of the q symbols as elements in fq. Now, we can think of words as polynomials over fq where the individual symbols of a word correspond to the different coefficients of the polynomial. To define a cyclic code, we pick a fixed polynomial, called generator polynomial. The code words of this cyclic code are all the polynomials that are divisible by this generator polynomial.

The key idea of the proposed class of codes is to use identity submatrices in the upper portion of the parity check matrix so that the burst error magnitudes or error pattern can be directly computed from the syndrome bits or symbols itself. The lower portion of the parity check matrix is then constructed using a base code so that the following conditions are satisfied.

1. All *b*-adjacent syndromes generated by XORing *b*-adjacent columns should be unique.
2. All syndromes for all possible combinations of columns within *b*-adjacent columns should be unique.
3. If multiples of a column are used to XOR instead of the original column in the above two cases, all syndromes thus generated should be unique.

Condition 1 ensures that no *b*-adjacent errors are mis-corrected. Condition 2 ensures that any number of errors within the *b*-adjacent columns are not mis corrected. The unique syndromes exactly identify which *b*-adjacent columns contain the errors. For non-binary codes like Reed-Solomon codes which correct on a symbol basis, each column can also have different multiples. These multiples can be identified from the upper *b*-rows of the parity check matrix. But to avoid mis-correction for different magnitudes of errors, condition 3 needs to be satisfied as well. Thus, the lower part of the parity check matrix is constructed in such a manner that all the conditions are satisfied. Also, an identity sub matrix of size atleast *b*is appended at the end of the parity check matrix. If a *r*x *r* identity sub matrix is used instead of a *b*x *b*, then the code is systematic by design. The general structure of a systematic parity check matrix, also called the H-matrix, of the proposed scheme .

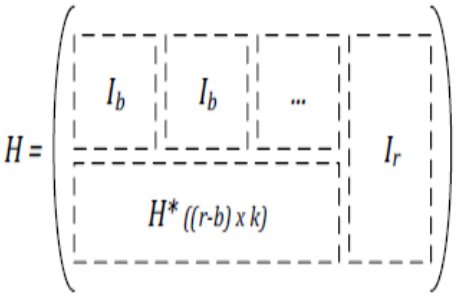

Fig 4.1 General Structure of a systematic parity check matrix of the proposed scheme.

The parity check matrix of the proposed scheme is similar to the codes in [23] and Fire codes [24]with the upper portion being identity matrices. The key difference in the construction of the proposed codes is that the above-mentioned codes use GF(2*b*) symbols. Thus, each identity sub-matrix has an element from GF(2*b*) as its correspond-ing lower sub-matrix. The proposed scheme instead uses an algorithmic construction based on certain conditions to construct the final parity check matrix using other existing codes, BCH codes.

ENCODING PROCEDURE

For a systematic code, the encoding procedure is a few XOR operations of the message bits or symbols which is then appended to the original message to form the code-word. If a code is non-systematic, then apart from the original encoding procedure of the base code used to construct the code, a few XOR operations are additionally required to compute the

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

parity check bits or symbols related to the upper portion of the parity check matrix. These parity check bits or symbols can either be append-ed at the end of the codeword or they can be stored separately in a given word. If stored separately, they are here-by called as Separate Parity. The two different forms of parity storage have been shown in Fig. 2.

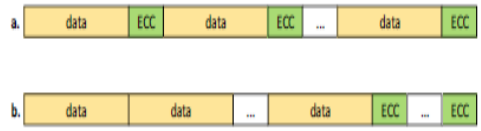

Fig. 2. (a) ECC bits stored alongside data bits. (b) ECC bits stored sepa-rately in memory (Separate Parity).

DECODING PROCEDURE

The general decoding procedure involves the two pro-cedures of computing the error pattern and the error locations. For the case of the proposed scheme, the first step is to compute the syndromes using the parity check matrix. The syndrome is computed by multiplying the parity check matrix with the received codeword. This is a simple XOR operation between all the data bits or symbols which have a corresponding 1 in the parity check matrix on a per row basis. The structure of the parity check matrix of the proposed scheme is such that the error pattern is directly represented by the upper *b* syndrome bits or symbols, where *b* is the size of the burst error being corrected. The rest of the syndrome bits or symbols are then used to compute the location of the burst error.

Consider a codeword of length *n*, *c*= (*c0*, *c1*, … *cn-1*), with the information length being *k* and number of check bits or symbols being *r*. Thus, the codeword length *n*is given by *n*= *k*+ *r*. The H-matrix is a matrix of *r* rows and *n* columns. The total number of syndrome bits or symbols is also given by *r*. Thus, the syndrome bits are computed by multiplying the H-matrix with the received codeword as shown in equation (1). The multiplication operation is simply an AND operation in case of binary bits. For non-binary *m*-bit symbols, multiplication is done over GF(2*m*). The addition operation is a XOR operation for both binary and non-binary cases.

$$\begin{pmatrix} S_0 \\ S_1 \\ \vdots \\ S_{r-1} \end{pmatrix} = \begin{pmatrix} h_{0,0} & \cdots & h_{0,n-1} \\ \vdots & \ddots & \vdots \\ h_{r-1,0} & \cdots & h_{r-1,n-1} \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ \vdots \\ C_{n-1} \end{pmatrix} \quad (1)$$

Next, we consider a burst error of size *b* starting from location *i*in the codeword. The error vector represents all the bits or symbols in the codeword that have changed. Thus, the error vector can be given by *e*= (0, 0, … *ei*, *ei+1*, … *ei+b-1*, 0, … 0). If we add the error vector to the received codeword, we get the original codeword back. If the received codeword is error free, the error vector is all zeros. Correspondingly, for a non-erroneous codeword, all the syndrome bits or symbols are also zero. Thus, computing the syndrome from an erroneous codeword is equivalent to computing the syndrome bits or symbols from its error vector. Also, let *x*= *i*mod *b*. The computed syndrome bits or symbols whena codeword with error vector *e*is multi-plied by the proposed H-matrix is given by equation (2).

For simplicity we assume *i*is a multiple of *b*. Then (*S0*, *S1*, …, *Sb-1*) = (*ei*, *ei+1*, …, *ei+b-1*). Also, for simplicity we consider the binary case with *k* data bits and *r* check bits. As can be seen from the computed syndrome bits, the error pattern is directly represented by the upper *b* syndrome bits. The lower (*r-b*) syndrome bits is a function of both the individual error magnitudes as well as the particular row of the H-matrix. Now we know, that for a burst error starting from location *i*, the error magnitudes are equal to the upper *b* syndrome bits. Therefore, for the locations

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

that are in error i.e. *i*to *i*+*b*-1, equation (3)is satisfied. For all other locations, equation (3)is not true. Thus, the error location computation is done on a group of *b* bits and for exactly the group of bits in error, equation (3)is satisfied. This is the basis for the proposed decoding procedure. The decoding works by considering each group of *b*-adjacent columns as a single large symbol and decoding on a per symbol basis. Any data bit *di* then is part of *b* symbols of *b*-bit each. An example has been shown for a 4-bit burst error correcting code in Fig. 3, the data but *di*is part of 4-bit symbols *Bi-3*, *Bi-2*, *Bi-1*and *Bi*. A *b*-bit burst error simply means an error in one of the *b*-bit symbols, and thus can be computed through equation (3). If for any of the *b*-bit symbols that the data bit is a part of equation (3) is satisfied, that means the data bit is in error. The error pattern of the data bit is simply the syndrome value $S\alpha$ where α is the row amongst the upper *b*-rows for which the corresponding column is a 1. Thus, the location of error for any data bit is the AND of all *b*-bit symbols of which it is a part. A data bit will not be in error only if it does not satisfy equation (3) for all *b*-bit symbols of which it is a part.

$$\begin{pmatrix} S_0 \\ S_1 \\ \vdots \\ S_x \\ S_{x+1} \\ \vdots \\ S_{b-1} \\ S_b \\ \vdots \\ S_{r-1} \end{pmatrix} = \begin{pmatrix} e_{i+((b-x) \bmod b)} \\ e_{i+((b-x+1) \bmod b)} \\ \vdots \\ e_i \\ e_{i+1} \\ \vdots \\ e_{i+((b-x-1) \bmod b)} \\ h_{b,0}e_i + \cdots + h_{b,b-1}e_{i+b-1} \\ \vdots \\ h_{r-1,0}e_i + \cdots + h_{r-1,b-1}e_{i+b-1} \end{pmatrix}$$

$$S_\beta + h_{\beta,0}S_0 + h_{\beta,1}S_1 + \cdots + h_{\beta,b-1}S_{b-1} = 0$$
$$\forall\, b \le \beta \le (r-1)$$

5.SIMULATION RESULT:

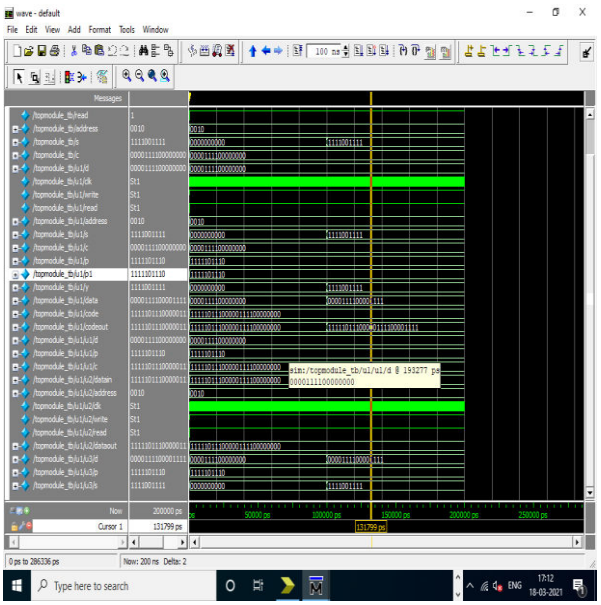

Fig 5.1 Simulation Result

Fig 5.1show the simulation result of proposed system. Where we can detect the error and correct the error

6.CONCLUSION AND FUTURE SCOPE

This paper presents a new class of single burst error correcting codes with parallel decoding. This class of codes are constructed by augmenting the parity check matrix of existing codes. A new decoding methodology is also presented which enables a one-step decoding logic enabling fast parallel decoding. These codes are particularly useful for memories whose performance is sensitive to access latencies. Comparisons with existing schemes show that the proposed class of codes achieve significant reduction in decoder area especially for high burst sizes with minimal addition to data redundancy. The results show that the proposed scheme also achieves better decoder latency compared to existing schemes for higher burst sizes.

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

Existing schemes can address smaller burst sizes much more efficiently than the proposed scheme. But with technology scaling as the number of bits that can flip due to a burst error increases, the proposed scheme provides an efficient solution in terms of decoder area and latency to correct the higher size burst error while enabling better throughput. Thus, as the dominant types of soft-errors are shifting with technology scaling towards localized clustered errors for different memory types, the proposed class of codes provides an efficient low complexity mechanism to tolerate such errors without any significant addition to data redundancy.

References

[1] R. Baumann, "Soft errors in advanced computer systems," in IEEE Design & Test of Computers, vol. 22, no. 3, pp. 258-266, May-Jun 2005.

[2] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo and T. Toba, "Impact of scaling on neutron-induced soft error in SRAMs from a 250 mm to a 22 nm design rule," in IEEE Transactions on Electron Devices, vol. 57, no. 7, pp. 1527-1538, Jul. 2010.

[3] D. Radaelli, H. Puchner, S. Wong and S. Daniel, "Investigation of multi-bit upsets in a 150 nm technology SRAM device," in IEEE Transactions on Nuclear Science, vol. 52, no. 6, pp. 2433-2437, Dec. 2005.

[4] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. LaiandO. Mutlu, "Flipping bits in memory without access-ing them: An experimental study of dram disturbance errors", in Proc. of ACM/IEEE International Symposium on Computer Archi-tecture (ISCA), pp. 361-372, 2014.

[5] A. Das and N.A. Touba, "Systematic b-Adjacent Symbol Error Correcting Reed-Solomon Codes with Parallel Decoding" in Proc. of IEEE VLSI Test Symposium, pp. 1-6, 2018.

[6] A. Das and N.A. Touba, "Low Complexity Burst Error Correct-ing Codes to Correct MBUs in SRAMs" in Proc. ofACMGreat Lakes Symposium on VLSI(GLSVLSI), pp. 219-224, 2018.

[7] H. O. Burton, "Some asymptotically optimal burst-correction codes and their relation to single-error-correcting reed-solom codes," in IEEE Transactions on Information Theory, vol. 17, no. 1, pp. 92–95, Jan. 1971.

[8] S. Baeg, S. Wen and R. Wong, "SRAM Interleaving Distance Selection with a Soft Error Failure Model," in IEEE Transactions on Nuclear Science, vol. 56, no. 4, pp. 2111-2118, Aug. 2009.

[9] R. Datta and N.A. Touba, "Generating Burst-Error Correcting Codes from Orthogonal Latin Square Codes - A Graph Theoret-
ic Approach," in Proc. of IEEE Symposium on Defect and Fault Tol-erance, pp. 367-373, 2011.

[10] P. Reviriego, S. Liu, J.A. Maestro, S. Lee, N.A. Touba and R. Datta, "Implementing Triple Adjacent Error Correction in Dou-ble Error Correction Orthogonal Latin Square Codes," in Proc. of IEEE Symposium on Defect and Fault Tolerance, pp. 167-171, 2013.

[11] P. Reviriego, M. Flanagan, S.-F. Liu and J. Maestro, "Multiple cell upset correction in memories using difference set codes," in IEEE Transactions on Circuits and Systems-I: Regular Papers, vol. 59, no. 11, pp. 2592–2599, Nov. 2012.

[12] P. Reviriego, S. Pontarelli, A. Evans and J. A. Maestro, "A Class of SEC-DED-DAEC Codes Derived from Orthogonal Latin Square Codes", in IEEE Transactions on Very Large Scale Integration (VLSI) Sys-tems, vol. 23, no. 5, pp. 968-972, May 2015.

[13] J. Kim, N. Hardavellas, K. Mai, B. Falsafi and J. Hoe, "Multi-bit error tolerant caches using two-dimensional error coding," in Proc. of IEEE/ACM

International Symposium on Microarchitecture (MICRO), pp. 197–209, 2007.

[14] C. Argyrides, D. Pradhan and T. Kocak, "Matrix codes for reliable and cost efficient memory chips," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 19, no. 3, pp. 420-428, Mar. 2011.

[15] A. Dutta and N.A. Touba "Multiple Bit Upset Tolerant Memory Using a Selective Cycle Avoidance Based SEC-DED-DAEC Code," in Proc. of IEEE VLSI Test Symposium, pp. 349-354, 2007.

[16] S. Shamshiri and K. T. Cheng, "Error-locality-aware linear coding to correct multi-bit upsets in SRAMs," in Proc. of IEEE International Test Conference (ITC), Paper 7.1, 2010.

[17] A. Neale and M. Sachdev, "A new SEC-DED error correction code subclass for adjacent MBU tolerance in embedded memory," in IEEE Transactions on Device and Materials Reliability, vol. 13, no. 1, pp. 223–230, Mar. 2013.

[18] L. S. Adalid, P. Reviriego, P. Gil, S. Pontarelli and J. A. Maestro, "MCU Tolerance in SRAMs Through Low-Redundancy Triple Adja-cent Error Correction," in IEEE Transactions on Very Large Scale Integra-tion (VLSI) Systems, vol. 23, no. 10, pp. 2332-2336, Oct. 2015.

[19] J. Li, P. Reviriego, L. Xiao and R. Zhang, "Efficient Implementations of 4-Bit Burst Error Correction for Memories", in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, no. 12, pp. 2037-2041, Dec. 2018.

[20] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar and S. Lu, "Reducing cache power with low-cost, multi-bit error-correcting codes," in Proc. of ACM/IEEE International Symposium on Computer Architecture (ISCA), pp. 83–93, 2010.

[21] K. Namba, S. Pontarelli, M. Ottavi and F. Lombardi, "A Single-Bit and Double-Adjacent Error Correcting Parallel Decoder for Multiple-Bit Error Correcting BCH Codes," in IEEE Transactions On Device And Materials Reliability, vol. 14, no. 2, pp. 664-671, Jun. 2014.