A New SEC-DED Error Correction Code Subclass for Adjacent MBU Tolerance in Embedded Memory

Adam Neale, *Student Member, IEEE,* and Manoj Sachdev, *Fellow, IEEE*

***Abstract*—The reliability concern associated with radiation induced soft errors in embedded memories increases as semiconductor technology scales deep into the sub-40 nm regime. As the memory bit-cell area is reduced, single event upsets (SEUs) that would have once corrupted only a single bit-cell are now capable of upsetting multiple adjacent memory bit-cells per particle strike. While these error types are beyond the error handling capabilities of the commonly used single error correction double error detection (SEC-DED) error correction codes (ECCs) in embedded memories, the overhead associated with moving to more sophisticated double error correction (DEC) codes is considered to be too costly. To address this, designers have begun leveraging selective bit placement to design SEC-DED codes capable of double adjacent error correction (DAEC) or triple adjacent error detection (TAED). These codes can be implemented for the same check-bit overhead as the conventional SEC-DED codes; however, no codes have been developed that use both DAEC and TAED together. In this work, a new ECC scheme is introduced that provides not only the basic SEC-DED coverage, but both DAEC and scalable adjacent error detection (xAED) with a reduction in miscorrection probability as well. Codes capable of up to 11-bits AED have been developed for both 16- and 32-bit standard memory word sizes, and a (39, 32) SEC-DED-DAEC-TAED code implementation that uses the same number of check-bits as a conventional 32 data-bit SEC-DED code is presented.**

***Index Terms*—Error correction codes (ECCs), multiple bit upsets (MBUs), memories, soft errors.**

I. Introduction

As semiconductor devices scale deep into the sub-40 nm regime, radiation induced soft errors are becoming a serious reliability challenge facing embedded memory designers [1]. Soft errors occur when a radiation particle, either an energetic neutron from a cosmic ray or an alpha particle from chip packaging material, strikes a register or memory cell's storage node causing an unintentional change in the logical data value of the storage element [2]. Depending on the criticality of the information being stored, this has the potential to lead to catastrophic system failures. Error correction codes (ECCs) have typically been used as a form of channel encoding to mitigate against these upsets by encoding the data with an additional set of check-bits [3]. The most commonly used ECCs for embedded memory applications are single error correction double error detection (SEC-DED) Hamming [4] and Hsiao codes [5]. These codes are capable of correcting one single-bit error or detecting two single-bit errors per memory word. The advantage of the SEC-DED codes is that they are simple to implement, and provide a minimal impact in terms of latency and area to the memory system. Recent studies have shown however, that while technology scaling leads to an increase in memory bit-cell density, it also increases the probability of upsetting multiple adjacent bits per radiation particle strike [1], [6]. These upsets, known as multiple cell upsets (MCUs), have typically been addressed through the use of bit-interleaving, where rather than storing a memory word's bits in physically adjacent memory cells, the bits of multiple memory words are interleaved throughout a memory row [3]. The physical separation of a memory word's bits can redistribute a multiple bit upset (MBU) within a single memory word into many single bit upsets (SBUs) across multiple words. Interleaving converts an upset beyond the error handling capabilities of an ECC into multiple smaller upsets that can be handled individually. There are limits however to the practical degree of interleaving. These are dictated by the memory's area, performance, and aspect ratio constraints. The physical separation of a memory word's bit-cells leads to an increase in signal routing complexity, which in turn increases the memory's area and latency. These penalties are exacerbated as technology scaling leads to larger MCUs, and a need for more interleaving; a typical static random access memory (SRAM) row consists of 4-8 interleaved memory words. Although, for small memories, register files, and content-addressable memories (CAMs), interleaving may not even be feasible and an alternative method for mitigating against these upsets is required.

Although more powerful ECCs such as Bose Chaudhuri Hocquenghem (BCH) [7], and Euclidean Geometry (EG) [8] codes have been developed to counteract MBUs, they do so at a significantly higher latency and area penalty as compared to the simpler SEC-DED codes. These codes require a large number of check-bits, which significantly increases the ECC's area overhead relative to the SEC-DED codes. Recently, designers have proposed selective cycle avoidance [6] and bit placement codes [9] that extend the standard SEC-DED codes to provide double adjacent error correction (DAEC) *or* triple adjacent error detection (TAED) codes for the same check-bit overhead as the standard SEC-DED codes. These codes are still limited however in the size of adjacent errors that they are capable of handling and can introduce a probability for misinterpreting non-adjacent errors as adjacent ones.

In this paper, a new ECC scheme is proposed that provides

Manuscript received October 1, 2012; revised November 13, 2012; accepted December 2, 2012. This work was supported by NSERC Discovery/PGS-D.

A. Neale, and M. Sachdev are with the Department of Electrical and Computer Engineering, 200 University Ave. W, University of Waterloo, Waterloo, ON, N2L 3G1, Canada (e-mail: ajneale@uwaterloo.ca; msachdev@uwaterloo.ca).

the basic SEC-DED coverage as well as both DAEC *and* scalable adjacent error detection (xAED) while reducing the adjacent/non-adjacent double-bit error miscorrection probability. This is accomplished by exploring the check-bit design space between the conventional SEC-DED code and the BCH double error correction (DEC) code. The degree of xAED and miscorrection probability are functions of the number of check-bits. The proposed code that uses the same number of check-bits as the conventional SEC-DED code provides both DAEC *and* TAED. This coding scheme provides a means for mitigating adjacent MBUs that is orthogonal to and independent of interleaving. If used in conjunction with interleaving, the provided error protection coverage will be the product of the ECC protection and the degree of interleaving.

The remainder of this paper is organized as follows Section II presents an overview of parity check matrices and how shortened binary linear block SEC-DED codes can be modified to provide adjacent error correction and detection. Section III describes the proposed ECC's structure and design procedure. Section IV provides an evaluation of the proposed codes. Finally, Section V concludes the paper.

II. Binary Linear Block Codes

SEC-DED codes are capable of correcting one single-bit error and detecting all possible combinations of double-bit errors. These codes fall into the category of binary linear block codes. The proposed class of SEC-DED-DAEC-xAED codes, or SDD-x for short, are a subset of the SEC-DED codes and can be characterized in the same manner. This section provides a rudimentary understanding of SEC-DED and linear block codes for the purpose of understanding the proposed class of codes. For a full treatment on linear block codes the reader is referred to [10] or [11].

Binary linear block codes are described by their number of data-bits, k, encoding and decoding check-bits, r, and the total number of stored code-bits, $n = k+r$. An (n, k) code projects the k-dimensional dataword onto a k-dimensional subspace of the n-dimensional codeword vector space of n-tuples. The code can be characterized by its $(r \times k)$ generator matrix $\mathbf{G}$ or its $(r \times n)$ parity check matrix $\mathbf{H}$. The $\mathbf{G}$-matrix is used for the encoding process and the $\mathbf{H}$-matrix for the decoding process. Either matrix is sufficient to fully describe the code. In this paper, codes are described using their $\mathbf{H}$-matrices. The vector $\mathbf{v}$ is a valid codeword provided that

$$\mathbf{v} \cdot \mathbf{H}^T = \mathbf{0} \tag{1}$$

where $\mathbf{H}^T$ is the transpose of $\mathbf{H}$ and $\mathbf{0}$ is the r-bit null vector.

An m-bit error $E(i_1, \ldots, i_m)$ inverts the bits $i_1, \ldots, i_m$ of a codeword $\mathbf{v}$. The error can be described by the error vector $\mathbf{e} = (e_1, \ldots, e_n)$. When $e_i = 1$ the i-th element of $\mathbf{e}$ is in error, whereas $e_i = 0$ indicates that no error has occurred at the i-th bit location of $\mathbf{e}$. The injection of the error vector $\mathbf{e}$ into the valid codeword $\mathbf{v}$ can be modeled by the bit-wise XOR of the two n-bit vectors to produce the resultant, or received, vector $\mathbf{u} = \mathbf{v} \oplus \mathbf{e}$.

The syndrome of an error E with the error vector $\mathbf{e}$ is calculated by passing the resultant vector $\mathbf{u}$ to the parity check matrix $\mathbf{H}$. This is modeled as

$$\mathbf{S}(E) = \mathbf{u} \cdot \mathbf{H}^T = (\mathbf{v} \oplus \mathbf{e}) \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T \tag{2}$$

where $\mathbf{S}(E)$ is the syndrome of the error E. This comes as a result of (1).

If $\mathbf{H}$ is considered to be a set of n column vectors,

$$\mathbf{H} = (h_1, \ldots, h_n) \tag{3}$$

then for the m-bit error $E(i_1, \ldots, i_m)$, the syndrome $\mathbf{S}(E)$ is given by

$$\mathbf{S}(E) = (h_{i_1} \oplus \cdots \oplus h_{i_m}) \tag{4}$$

which is the column-wise XOR result of the columns in $\mathbf{H}$ corresponding to the bit error locations in $\mathbf{e}$.

An error E is detectable provided that its corresponding syndrome vector $\mathbf{S}$(E) is non-zero. Additionally, the set of errors $E_1, \ldots, E_s$ is correctable provided the set of syndromes $\mathbf{S}(E_1), \ldots, \mathbf{S}(E_s)$ are unique. For an r-bit syndrome, there are $2^r - 1$ non-zero error patterns. The all-zero pattern is reserved to indicate that no error has occurred. An r-row $\mathbf{H}$-matrix consisting of all of the $2^r - 1$ unique non-zero columns is known as a complete Hamming code. For complete Hamming codes, $n = 2^r - 1$ and $k = 2^r - 1 - r$. The error handling capability of this code is limited to single error correction only. Since every syndrome directly corresponds to a single-bit error, any other error type would be masked as another single-bit error. To increase the error handling capabilities of the code (allowing for the correction and/or detection of other error types) some columns can be removed from the $\mathbf{H}$-matrix of the complete Hamming code. This results in what is known as a shortened code. By letting

$$k < 2^r - 1 - r \tag{5}$$

unused syndromes are available to map to different error types. This concept is leveraged to mitigate against MBUs. Common practice is to set k equal the width of standard memory word sizes, e.g., 8, 16, 32, 64 bits, etc... and then determine the required number of check-bits r to provide the desired level of error protection. Given a direct one-to-one correspondence between a particular error and a syndrome, that particular error can be corrected. If multiple error patterns map to the same syndrome however, then those errors can be detected but not corrected, since there is no way of distinguishing which particular error generated the syndrome. SEC-DED codes require that each of the $\mathbf{H}$-matrix column vectors are unique and non-zero. This provides the SEC coverage, while ensuring that the XOR result of any two columns are non-zero and not equal to any of the individual columns provides the DED coverage. The Hsiao SEC-DED code for instance, provides its DED capabilities by using only odd weighted columns for its $\mathbf{H}$-matrix. The XOR result of any two of these odd weighted columns vectors will result in a non-zero, even weighted column vector, which by definition is not contained within the odd weighted column $\mathbf{H}$-matrix. Careful arrangement of the $\mathbf{H}$-matrix column vectors through selective bit placement can allow for the error correction and/or detection of other error types such as adjacent MBUs.

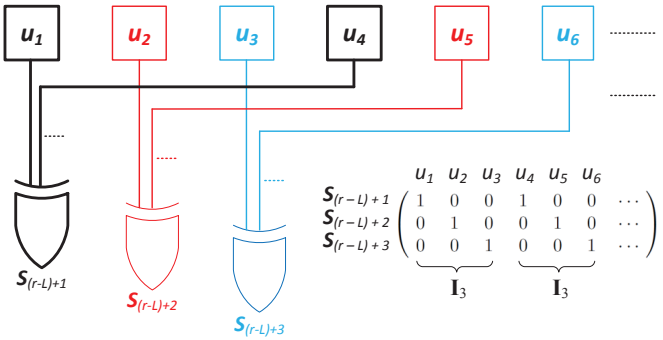


Fig. 1. By using the repeated $\mathbf{I}_L$ matrix structure, each of the code-bits used to generate the subset of the syndrome bits are separated by a physical distance of L bit-cells.

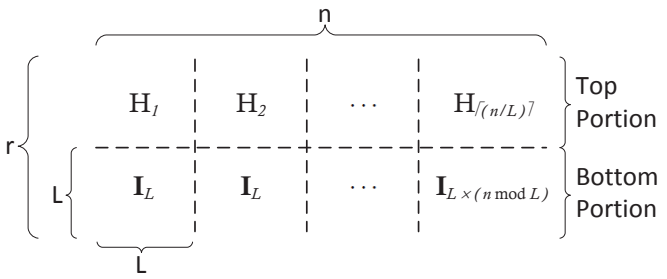


Fig. 2. Generalized **H**-matrix structure for the proposed class of ECCs.

A. Selective Bit Placement

Through judicious selection and arrangement of the **H**-matrix columns, the error protection coverage provided by shortened codes can be shaped to mitigate against certain error types. By arranging the **H**-matrix columns in such a way that the syndromes produced by error patterns affecting multiple adjacent bits are different from those that provide error correction, adjacent error detection can be achieved. By extending this idea such that the syndromes produced by multiple adjacent bit errors are distinct, the code can then provide adjacent error correction as well. This is shown in [9] where the authors provides examples of SEC-DAED and SEC-DED-TAED codes, and in [6] for SEC-DED-DAEC codes.

B. Repeated Identity Matrices

A technique used in both Fire [12] and Burton [13] codes is to construct a portion of the **H**-matrix by repeated horizontal concatenation of the $L \times L$ identity matrix, $\mathbf{I}_L$. This physically separates the bits used for each of these syndrome-bit values by a physical distance of L-bits, and essentially mimics the interleaving process within the **H**-matrix. This produces the matrix

$$\mathbf{H} = \left(\begin{array}{c} H \\ \hline \mathbf{I}_L \quad \mathbf{I}_L \quad \cdots \quad \mathbf{I}_L \quad \mathbf{I}_L \end{array} \right) \tag{6}$$

where the bottom L rows consist of the repeated $\mathbf{I}_L$ matrix, and the top $r - L$ rows, represented by H, are the rows of the **H**-matrix not included in the repeated $\mathbf{I}_L$ matrix structure. The $\mathbf{I}_L$ matrix structure is shown in Fig. 1 using the repeated $\mathbf{I}_3$ matrix. Notice that the bits used for each of the syndrome-bit values are separated by a physical distance of three bit-cells. For an L adjacent bit upset, only one bit in each of the L sub-syndrome bits will be affected. The **H**-matrix consisting of only the repeated $\mathbf{I}_L$ matrix is capable of detecting up to $2L - 1$ adjacent-bit errors.

By introducing adjacent error correction into an adjacent error detection scheme, the degree of adjacent error detection is reduced. This occurs due to the uniqueness requirement of all of the error correcting syndromes. When combined with an error correcting scheme, the sub-syndrome pattern generated by a $2L - 1$ bit error will be identical to that of a single-bit error. This will cause a miscorrection as opposed to the appropriate error detection action. When used in conjunction with adjacent error correction, the degree of adjacent error detection provided by the repeated $\mathbf{I}_L$ matrix structure is given by

$$xAED = 2L - (yAEC + 1) \tag{7}$$

where $xAED$ and $yAEC$ are the degrees of adjacent error detection and adjacent error correction respectively.

III. Proposed Codes

In the following section, a general design procedure for generating scalable SDD-x codes will first be discussed, and then a SDD-x code with TAED (SDD-T) will be presented using an actual 32 data-bit implementation. The proposed SDD-T code is a special case of the proposed class of SDD-x codes. The most advantageous feature of the SDD-T code is that it can be implemented using the same number of check-bits as the standard SEC-DED code. The **H**-matrices for each of these codes is constructed by concatenating a series of sub-matrices, H_i, designed using selective bit placement to the repeated $\mathbf{I}_L$ matrix structure.

A. Code Structure

The generalized **H**-matrix structure for the proposed class of ECCs is shown in Fig. 2. The matrix is partitioned into two separate portions. The bottom portion consists of the $L \times L$ identity matrix, $\mathbf{I}_L$, horizontally concatenated across n columns, with the $\lceil \frac{n}{L} \rceil$-th matrix truncated to size $L \times (n \mod L)$ to align with the n-bit codeword length. The top portion of the matrix consists of a set of $\lceil \frac{n}{L} \rceil$ unique sub-matrices, H_i, one associated with each bottom portion $\mathbf{I}_L$ matrix. For each H_i sub-matrix the odd column vectors are identical to one another, and likewise for the even column vectors. For example, for a four column H_i sub-matrix, the first, and third columns are identical to one another, and the same is true for the second and fourth columns. The four column H_i sub-matrix is described by

$$H_i = (h_{\mathrm{o}_i},\ h_{\mathrm{e}_i},\ h_{\mathrm{o}_i},\ h_{\mathrm{e}_i}) \tag{8}$$

where h_{o_i} represents the odd column vectors for the i-th top portion sub-matrix and h_{e_i} represents the even column vectors. When vertically concatenated with its respective bottom portion $\mathbf{I}_L$ matrix, these H_i matrices each form a new $(r \times L)$ sub-matrix each comprised of L unique columns.

B. Code Properties and Design Constraints

The proposed class of SDD-x codes have the following properties, they can:

1) Correct all single-bit errors,
2) Detect all double-bit errors,
3) Correct all adjacent double bit errors,
4) Detect all adjacent errors less than or equal to x bits,

To make this possible, the **H**-matrix must comply with the following constraints:

1) All columns must be non-zero,
2) All columns must be distinct,
3) The XOR result of any two columns must not equal any of the individual columns,
4) The XOR result of any two adjacent columns must be distinct,
5) The XOR result of any adjacent columns greater than two and less than or equal to x must produce a non-zero vector not equal to any of the error correcting syndromes,

Constraint 1 ensures that no single-bit errors match the error free, all-zero syndrome. Constraint 2 ensures that all single-bit errors are uniquely identifiable and thus are correctable. This provides the SEC functionality. Constraint 3 restricts the syndromes of all double-bit errors from matching those of the single-bit errors. This in conjunction with Constraint 2 provides the DED functionality. Constraint 4 ensures that the syndrome of all adjacent double-bit errors are distinct, and combined with Constraints 2 and 3 ensure that these syndromes are different from the single-bit error correcting syndromes. This provides the DAEC functionality. Finally, Constraint 5 ensures that any x or fewer adjacent bit errors produce a non-zero (and thus detectable) syndrome pattern that does not conflict with any of the other single-bit or double adjacent bit error correcting syndrome patterns. This is sufficient to provide the xAED functionality.

C. Code Design Procedure

Provided the number of information data-bits to be encoded, k, and the required degree of error protection coverage (e.g., SDD-x for the case of the proposed codes), one of the proposed **H**-matrix can be constructed. This first requires the determination of it design parameters. This includes the size of its repeated $\mathbf{I}_L$ matrix, required number of check-bits, r, and total number of code-bits, $n = r + k$. Once these have been determined, the columns of **H** can be selected, and arranged according to the constraints listed in Section III-B. This design procedure is described in the subsequent subsections.

1) Required $\boldsymbol{I}_L$ Matrix Size: By rearranging Equation 7 and letting $yAEC = 2$, the required $\mathbf{I}_L$ matrix size L, depends on the specified degree of adjacent error detection, $xAED$. This is given by

$$L = \left\lceil \frac{xAED + 3}{2} \right\rceil \tag{9}$$

For the proposed class of codes, TAED ($xAED = 3$) requires the $\mathbf{I}_3$ matrix. Codes have been implemented providing up to $xAED = 11$ bits of adjacent error detection. These codes require the repeated $\mathbf{I}_7$ matrix.

2) Required Number of Checkbits: The columns in the top portion of the **H**-matrix must be selected such that each complete column (the vertical concatenation of the top and bottom portion columns) complies with the constraints detailed in Section III-B. Knowledge of the repeated $\mathbf{I}_L$ bottom portion matrix can be leveraged when choosing the top portion columns. Since the $\mathbf{I}_L$ matrix is repeated across all of the bottom portion **H**-matrix columns, the top portion of the **H**-matrix effectively consists of a set of $\lceil n/L \rceil$ H_i sub-matrices each containing a maximum of L columns. Since the $\mathbf{I}_L$ matrix is repeated, the i-th column of each top portion sub-matrix will have an identical bottom portion $\mathbf{I}_L$ sub-column beneath it. To ensure the overall column uniqueness constraint, the top portion of the **H**-matrix can be divided into L bins (one for each column of the $\mathbf{I}_L$ matrix) each containing at most $\lceil n/L \rceil$ distinct columns. To express these $\lceil n/L \rceil$ binary columns, at least 2^{r-L} bits are required, where $r - L$ is the number of rows, or check-bits, used in the top portion of the matrix. Therefore, the required number of check-bits, r, must satisfy the following relation

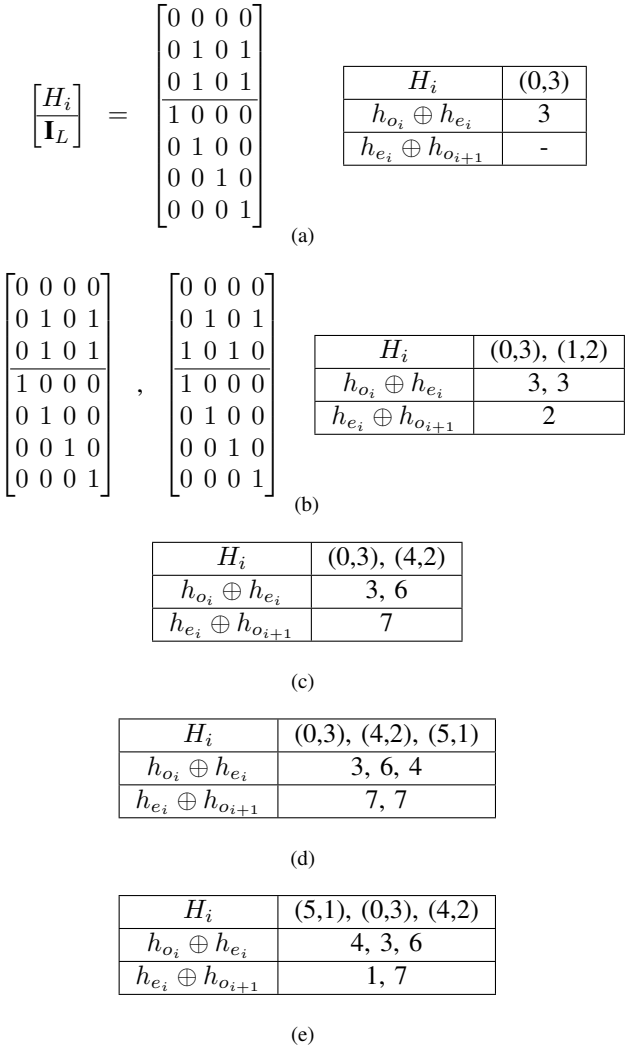
$$2^{r-L} \geq \left\lceil \frac{n}{L} \right\rceil \tag{10}$$

This will ensure that there are a sufficient number of check-bits such that each complete column is unique.

3) Column Vector Selection Procedure: In addition to the distinct column constraint (Constraint 2), the XOR result of all adjacent column pairs must be distinct from one another and from all individual columns (Constraints 3 and 4). The repeated $\mathbf{I}_L$ matrix will ensure the XOR uniqueness result of any two adjacent columns from any individual column. This is because the XOR result of any two adjacent bottom portion columns will have a weight of two, whereas the individual bottom portion column only have a weight of one. To ensure the XOR uniqueness result of all two adjacent column pairs requires a careful selection of the top portion columns.

For an n-bit codeword, rather than attempting to find an ordered set of n unique columns with $n - 1$ unique adjacent column XOR pairs in one attempt, the problem can be subdivided using a two-phase solution approach. Within the first phase, columns satisfying the uniqueness constraints are selected for each individual H_i sub-matrix. These sub-matrices are then arranged in the second phase to satisfy the overall uniqueness constraints for the entire **H**-matrix. This procedure is described in Fig. 3 using 3×4 H_i sub-matrices concatenated with $\mathbf{I}_4$ matrices as an example. For clarity, decimal equivalent values are used for the column values of each H_i matrix. For each of these column values, the top row represents the most significant bit, and the bottom row represents the least significant bit.

First, notice that the matrix shown in Fig. 3(a) is a 3×4 H_i sub-matrix of the form described in Equation 8 concatenated with the $\mathbf{I}_4$ matrix, and this matrix conforms with all of the constraints listed in Section III-B. For this matrix, only the two $r - L$ bit column vectors $h_{o_i} = 0$ and $h_{e_i} = 3$ need to be considered to meet the constraint requirements. The first



H_i	(0,3)
$h_{o_i} \oplus h_{e_i}$	3
$h_{e_i} \oplus h_{o_{i+1}}$	-

H_i	(0,3), (1,2)
$h_{o_i} \oplus h_{e_i}$	3, 3
$h_{e_i} \oplus h_{o_{i+1}}$	2

H_i	(0,3), (4,2)
$h_{o_i} \oplus h_{e_i}$	3, 6
$h_{e_i} \oplus h_{o_{i+1}}$	7

H_i	(0,3), (4,2), (5,1)
$h_{o_i} \oplus h_{e_i}$	3, 6, 4
$h_{e_i} \oplus h_{o_{i+1}}$	7, 7

H_i	(5,1), (0,3), (4,2)
$h_{o_i} \oplus h_{e_i}$	4, 3, 6
$h_{e_i} \oplus h_{o_{i+1}}$	1, 7

Fig. 3. Column vector selection procedure.

phase of the solution method consists of finding a set of $\lceil n/L \rceil$ distinct $H_i = (h_{o_i}, h_{e_i})$ column vector pairs with a distinct set of $h_{o_i} \oplus h_{e_i}$ values. In the case of the $H_i = (0, 3)$ matrix, this gives a value of 3. In Fig. 3(b) a second H_i sub-matrix $(1,\ 2)$ is added. The $(0,\ 3)$, $(1,\ 2)$ matrices each have individually unique columns, but they produce the same $h_{o_i} \oplus h_{e_i}$ values. This causes the **H**-matrix to fail Constraint 4. By replacing the H_i pair $(1,\ 2)$ with $(4,\ 2)$, as is shown in Fig. 3(c), these matrices produce distinct $h_{o_i} \oplus h_{e_i}$ values. Once the set of column vector pairs have been successfully selected, this marks the end the Phase 1 solution.

The second phase of the solution method consists of arranging the H_i matrices in a manner such that the XOR result of the two adjacent columns in each pair of adjacent H_i sub-matrices are distinct. For the case of an even $\mathbf{I}_L$ matrix size, this requires that the set of $h_{e_i} \oplus h_{o_{i+1}}$ values form a distinct set. In Fig. 3(d), the column vector pairs $(0,\ 3)$, $(4,\ 2)$, and $(5,\ 1)$ are individually unique, and provide a valid Phase 1 solution; however, the adjacent columns for adjacent H_i sub-matrices fail to satisfy Constraint 4. This can be remedied by rearranging the H_i matrices as is shown in Fig. 3(e), to produce a valid Phase 2 solution. A complete solution consists of an ordered set of $\lceil n/L \rceil$ H_i sub-matrices that satisfy all of the constraints listed in Section III-B. This is shown in Table I for the proposed (39, 32) $\mathbf{I}_3$ SDD-T code.

This solution method reduces the required number of columns to be considered during the design process from n down to a maximum of $2 \times \lceil n/L \rceil$. For example, for a (39, 32) code using the $\mathbf{I}_3$ matrix, only $2 \times \lceil 39/3 \rceil = 26$ values need to be considered, while for a (42, 32) code using the $\mathbf{I}_6$ matrix, this value is reduced to $2 \times \lceil 42/6 \rceil = 14$. By reducing the number of columns to consider, the search space for candidate **H**-matrices is reduced as well, thereby easing the **H**-matrix construction procedure.

D. Construction Algorithm

The two-phase solution method has been implemented in Matlab for constructing valid **H**-matrices. The algorithm, as detailed in Fig. 4, takes the codeword length, $\mathbf{I}_L$ size, and maximum iteration counters for each phase of the solution as inputs, and returns a set of valid **H**-matrices as output. The algorithm initially selects a set of candidate H_i column vector pairs, $H(o, e)$, where o and e are top portion column vectors, either randomly, or by minimum weight. This is considered to be Phase 0. Each of these pairs represent the odd and even column vectors in each of the H_i sub-matrices. To ensure overall **H**-matrix column uniqueness, all of the o elements in each pair are distinct, and likewise for the e elements. The XOR result of each H_i column vector pair is then calculated as $c = o \oplus e$. For the given set of column vectors, the first solution phase attempts to find sets of distinct c elements. Distinct sets are stored and passed to Phase 2. For sets with duplicate c elements, the e elements are randomly swapped, and the c elements are recalculated. This process is iterated p1Max number of times for p0Max sets of column vector pairs. Provided a valid set of Phase 1 solutions, Phase 2 attempts to order the list of H_i column vector pairs in such a manner that the XOR result, d, of the adjacent columns of any two adjacent H_i column vector pairs are distinct. For an odd $\mathbf{I}_L$ size this requires the set of $H(i).o \oplus H(i+1).o$ be unique, whereas for an even $\mathbf{I}_L$ size, the results of $H(i).e \oplus H(i+1).o$ must be unique. Each valid Phase 1 solution undergoes p2Max iteration attempts. All valid Phase 2 solutions make up a set of valid **H**-matrices for the given set of input parameters.

E. Example (39, 32) $\boldsymbol{I}_3$ SDD-T Code

A search result for a (39, 32) SDD-T code using the proposed design procedure is shown in Fig. 5. The **H**-matrix uses the repeated $\mathbf{I}_3$ matrix and $r = 7$ check-bits to encode/decode $k = 32$-bit datawords resulting in $n = 39$-bit codewords. The matrix complies with all of the design constraints listed in Section III-B, and the decimal equivalent values for each column vector, adjacent column vector XOR pair, and H_i sub-matrix column vector pair is shown in Table I.

By inspection of Table I it is clear that all of the individual columns are distinct and non-zero, and the XOR results of all of the adjacent column vector pairs are unique, non-zero, and distinct from the set of individual column vectors. The use of the repeated $\mathbf{I}_3$ matrix ensures that the XOR result of

$$\mathbf{H} = \left(\begin{array}{c|c|c} H_1 & \cdots & H_{13} \\ \hline \mathbf{I}_3 & \cdots & \mathbf{I}_3 \end{array}\right) = \left(\begin{array}{cccccccccccccc} 010 & 010 & 010 & 000 & 000 & 111 & 000 & 101 & 010 & 101 & 000 & 101 & 101 \\ 111 & 000 & 000 & 101 & 010 & 000 & 101 & 010 & 000 & 010 & 101 & 010 & 101 \\ 000 & 010 & 111 & 101 & 010 & 000 & 111 & 101 & 101 & 000 & 000 & 111 & 010 \\ 000 & 010 & 101 & 000 & 101 & 111 & 101 & 111 & 000 & 000 & 111 & 010 & 010 \\ \hline 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \\ 010 & 010 & 010 & 010 & 010 & 010 & 010 & 010 & 010 & 010 & 010 & 010 & 010 \\ 001 & 001 & 001 & 001 & 001 & 001 & 001 & 001 & 001 & 001 & 001 & 001 & 001 \end{array}\right)$$

Fig. 5. Example of a (39, 32) $\mathbf{I}_3$ SDD-T code using the proposed design procedure.

TABLE I
PROPOSED (39, 32) $\mathbf{I}_3$ SDD-T CODE'S UNIQUENESS TABLE

Individual Columns	36, 98, 33, 4, 90, 1, 28, 82, 25, 52, 2, 49, 12, 50, 9, 76, 74, 73, 60, 18, 57, 92, 42, 89, 20, 66, 17, 68, 34, 65, 44, 10, 41, 84, 58, 81, 100, 26, 97
Adjacent Column Pairs	70, 67, 37, 94, 91, 29, 78, 75, 45, 54, 51, 61, 62, 59, 69, 6, 3, 117, 46, 43, 101, 118, 115, 77, 86, 83, 85, 102, 99, 109, 38, 35, 125, 110, 107, 53, 126, 123
H_i Sub-matrix Column Pairs	(4, 12), (0, 11), (3, 10), (6, 0), (1, 6), (9, 9), (7, 2), (11, 5), (2, 8), (8, 4), (5, 1), (10, 7), (12, 3)

any 3-adjacent columns is not equal to any of the individual columns or XOR result of any of the adjacent column pairs. Compliance with these constraints are sufficient to provide the SDD-T coverage for 32-bit datawords.

IV. EVALUATION

H-matrices for the proposed scheme have been implemented and verified for functionality using a custom test bench designed in Matlab for typical memory word sizes of 16 and 32 data-bits. Relevant high-level performance metrics have been extracted directly from the codes' **H**-matrices and compared with Hsiao SEC-DED [5], Dutta SEC-DED-DAEC [6], and BCH DEC [7] codes. The codes are compared in terms of their provided error correction and detection capabilities, required number of check-bits, encoder/decoder 2-input XOR logic gate count, and miscorrection probabilities for non-adjacent double bit errors. The values reported for each of the proposed codes is the **H**-matrix implementation found to have the minimum miscorrection probability for the minimum encoder/decoder 2-input XOR logic gate count. Comparisons for the 16-data-bit implementations are shown in Fig. 6 and Fig. 7, and 16 and 32 data-bit performance metrics are summarized in Table II.

A. Error Correction and Detection Capability Analysis

In terms of error correction, the Hsiao code can correct all single-bit errors. Both the Dutta SEC-DED-DAEC and the proposed codes extend this to providing double adjacent-bit error correction, while the BCH DEC code provides double-bit error correction for all adjacent and non-adjacent double bit errors alike. In terms of error detection, the proposed code is the only one capable of detecting errors of size greater than two bits. A summary of the adjacent error detection coverage for each of the 16-data-bit implementations is shown in Fig. 6.

While the Hsiao and Dutta SEC-DED-DAEC codes (6 check-bits) and the BCH DEC code (10 check-bits) each provide 2 bits of adjacent error detection, the proposed codes offer between 5 and 11 bits of $xAED$ as the $\mathbf{I}_L$ matrix is scaled from $\mathbf{I}_4$ up to $\mathbf{I}_7$. As the $\mathbf{I}_L$ matrix size (and number of check-bits) increases, the degree of $xAED$ increases by 2 bits/check-bit. For the cost of one additional check-bit over the (22, 16) SEC-DED-DAEC code, the proposed (23, 16) $\mathbf{I}_4$ code provides 5 bits of AED, while for the same number of check-bits as the (26, 16) BCH DEC code, the proposed (26, 16) $\mathbf{I}_7$ code provides 11 bits of AED. This provided degree of error protection is identical for each of the 32-data-bit implementations, although one additional check-bit is required for each implementation to compensate for the doubling of the dataword size.

B. Encoder/Decoder XOR Logic Gate Count

The area required for the encoder-decoder circuits is proportional to the XOR gate count of the syndrome generation logic. The number of 2-input XOR logic gates can be computed directly from the **H**-matrix and is equal to the total matrix weight minus its number of rows. The 2-input XOR logic gate count for the 16-data-bit implementations is shown in Fig. 7.

Notice that the required number of 2-input XOR gates decreases as the size of the $\mathbf{I}_L$ matrix is increased for the proposed codes. For one additional check-bit, the proposed (23, 16) $\mathbf{I}_4$ code uses 12.5% fewer XOR gates than the (22, 16) Hsiao and Dutta codes. While for the same number of check-bits, the proposed (26, 16) $\mathbf{I}_7$ code uses 62.8% fewer 2-input XOR gates than the (26, 16) BCH DEC code.

C. Miscorrection Probability

A shortcoming of the Dutta SEC-DED-DAEC code and the proposed SDD-x codes is that to minimize the required number of check-bits, some of the adjacent bit error syndromes are shared with non-adjacent bit error syndromes. It is therefore possible for non-adjacent double-bit errors to be misinterpreted by the decoder as adjacent double-bit error, and thus result in a miscorrection. Although potentially detrimental to the system, this drawback is tolerated since the probability of non-adjacent double-bit errors is substantially lower than the probability of adjacent double-bit errors [6]. An effort is still made however to limit the number of sharable syndromes between these error types. The miscorrection probability of a non-adjacent double error being masked as an adjacent double error is given by

$$\text{Miscorrection Probability} = \frac{\text{Sharable Syndromes}}{\binom{n}{2} - (n-1)} \qquad (11)$$

Input: n, L, p0Max, p1Max, p2Max
Output: setOf(p2Solutions)

Object: $H(i)$ Column vector pair, $H(o, e)$
o = Column vector 1; e = Column vector 2
$c = o \oplus e$
End Object

Object: List of $H(i)$ column vector pairs, H
$length$ = ceiling(n/L)
if L is Odd **then**
$d = H(i).o \oplus H(i+1).o$ ▷ Odd $\mathbf{I}_L$ Size
else
$d = H(i).e \oplus H(i+1).o$ ▷ Even $\mathbf{I}_L$ Size
end if
End Object

BEGIN CONSTRUCTION ALGORITHM
for $p0$ = 1 **to** p0Max **do** ▷ Phase 0
Create list of $H.length$ elements $H(o, e)$ via:
1. Random Fill, 2. Minimum Weight,
3. Modification of an existing list
Calculate setOf($H(i).c$)

for $p1$ = 1 **to** p1Max **do** ▷ Phase 1
if setOf($H(i).c$).isUnique AND
!setOf(p1Solutions).contains(H) **then**
Add H to setOf(p1Solutions)
Randomly swap a # of $H(i).e$ elements
else
for each duplicate in setOf($H(i).c$) **do**
Randomly swap $dup - 1$ $H(i).e$ elements
end for
end if
Recalculate setOf($H(i).c$)
end for ▷ End Phase 1
end for ▷ End Phase 0

if !setOf(p1Solutions).isEmpty **then** ▷ Phase 2
for each entry in setOf(p1Solutions) **do**
Calculate setof($H.d$)

for $p2$ = 1 **to** p2Max **do**
if setOf($H.d$).isUnique AND
!setOf(p2Solutions).contains(H) **then**
Add H to setOf(p2Solutions)
Randomly swap a # of $H(i)$'s
else
for each duplicate in setOf($H.d$) **do**
Randomly swap $dup - 1$ $H(i)$'s
end for
end if
Recalculate setOf($H.d$)
end for
end for
end if ▷ End Phase 2
END CONSTRUCTION ALGORITHM

Fig. 4. **H**-matrix construction algorithm.

where the numerator is the number of sharable syndromes between the two error types, and the denominator is the number of possible non-adjacent double-bit errors for the given n-bit codeword size (i.e., the total number of potential double-bit error combinations minus the number of double adjacent bit error patterns).

While the main purpose of increasing the number of check-bits is to increases the xAED functionality, as an additional benefit the miscorrection probability of the proposed codes is **also** reduced for each additional check-bit over the SEC-DED-DAEC codes. As the number of check-bits increases, the number of possible non-zero syndromes increases while

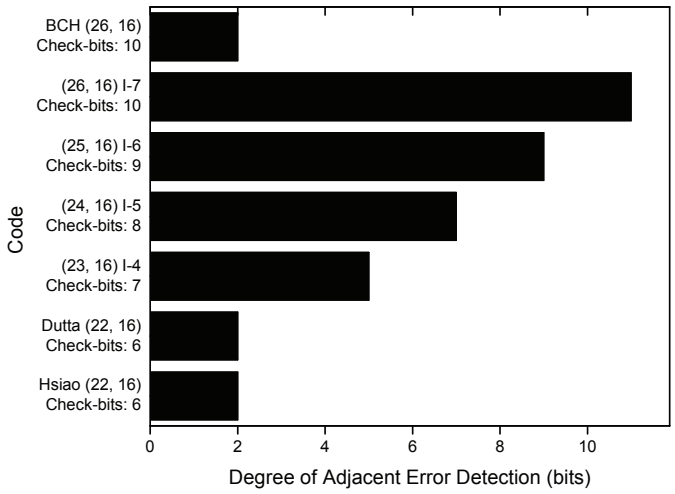


Fig. 6. Degree of adjacent error detection for 16-data-bit codes.

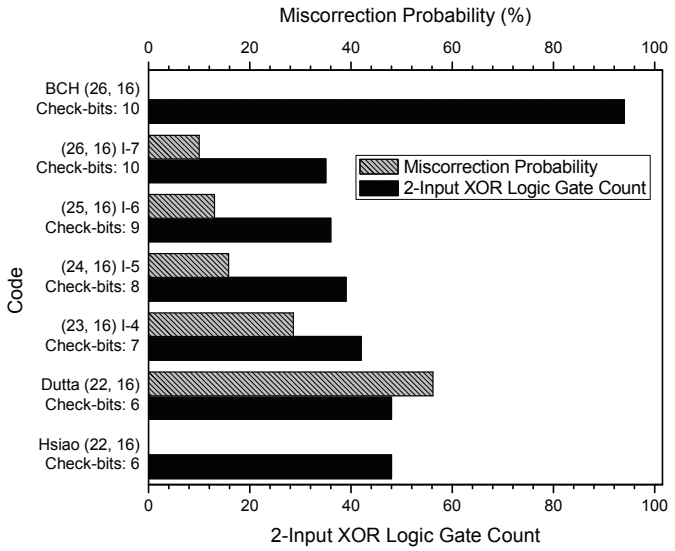


Fig. 7. Syndrome generation 2-input XOR logic gate count and miscorrection probabilities for 16-data-bit codes.

the number of error correcting syndromes for the code remains fixed. This is shown in Fig. 7. For the (22, 16) Dutta code there is a 56.2% miscorrection probability, while this probability is reduced to 10.0% for the (26, 16) $\mathbf{I}_7$ proposed code. The miscorrection probability of the Hsiao SEC-DED and DEC BCH codes are intrinsically zero since the Hsiao SEC-DED code does not perform any double error correction, and the BCH DEC code is capable of correcting *all* adjacent and non-adjacent double-bit errors alike.

D. Summary of 16- and 32-Data-bit Implementation Performance Metrics

The 32-data-bit implementation metrics for the proposed codes are shown in Table II and show similar trends to the 16-data-bit implementations (also included). For the 32-data-bit implementations, $\mathbf{I}_3$ codes were found that use the same number of check-bits as the Hsiao and Dutta codes, and provide 3 bits of AED. An $\mathbf{I}_3$ solution was not found for the

TABLE II
16- AND 32-DATA-BIT SDD-x CODE METRICS

16 Data-bit Codes					
(n, k) **I**-Size	Code	Check-bits	XOR Gates	Mis. Prob.	xAED
(22, 16)	Hsiao	6	48	0.000	2
(22, 16)	Dutta	6	48	0.562	2
(23, 16) $\mathbf{I}_4$	Proposed	7	42	0.286	5
(24, 16) $\mathbf{I}_5$	Proposed	8	39	0.158	7
(25, 16) $\mathbf{I}_6$	Proposed	9	36	0.130	9
(26, 16) $\mathbf{I}_7$	Proposed	10	35	0.100	11
(26, 16)	DEC BCH	10	94	0.000	2
32 Data-bit Codes					
(n, k) **I**-Size	Code	Check-bits	XOR Gates	Mis. Prob.	xAED
(39, 32)	Hsiao	7	96	0.000	2
(39, 32)	Dutta	7	96	0.539	2
(39, 32) $\mathbf{I}_3$	Proposed	7	98	0.498	3
(40, 32) $\mathbf{I}_4$	Proposed	8	88	0.248	5
(41, 32) $\mathbf{I}_5$	Proposed	9	84	0.214	7
(42, 32) $\mathbf{I}_6$	Proposed	10	80	0.090	9
(44, 32)	DEC BCH	12	200	0.000	2

16-data-bit implementations.

V. CONCLUSION

The proposed class of ECCs presented in this paper explore the check-bit design space between the conventional SEC-DED and BCH DEC codes to provide a new type of SEC-DED code capable of both double adjacent error correction and a scalable amount of adjacent error detection. The degree of this adjacent error detection depends on the amount of check-bits used to implement the code. A special case of the proposed code offers both triple adjacent error detection and double adjacent error correction for the same number of check-bits as the conventional SEC-DED code. The emphasis on adjacent error correction and detection is significant for aggressively scaled, sub-40 nm bulk CMOS technologies in which MBUs are becoming more prevalent. While bit interleaving has traditionally been used to mitigate adjacent MBUs, this coding scheme offers an orthogonal approach to enhance this protection, and an attractive solution for when the practicality of bit interleaving is either limiting or not feasible. Performance evaluations have been provided for both 16- and 32-data-bit implementations, and a (39, 32) SEC-DED-DAEC-TAED code example is shown.

REFERENCES

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

[2] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 3, pp. 305 – 316, Sept. 2005.

[3] E. Fujiwara, *Code Design for Dependable Systems - Theory and Practical Applications*. John Wiley and Sons, Inc., 2006.

[4] R. Hamming, "Error correcting and error detecting codes," *Bell Sys. Tech. Journal*, vol. 29, pp. 147–160, April 1950.

[5] M. Y. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes," *IBM Journal of Research and Development*, vol. 14, no. 4, pp. 395 –401, July 1970.

[6] A. Dutta and N. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in *VLSI Test Symposium, 2007. 25th IEEE*, May 2007, pp. 349 –354.

[7] R. Naseer and J. Draper, "Parallel double error correcting code design to mitigate multi-bit upsets in SRAM s," in *Solid-State Circuits Conference, 2008. ESSCIRC 2008. 34th European*, Sept. 2008, pp. 222 –225.

[8] H. Naeimi and A. DeHon, "Fault secure encoder and decoder for nanomemory applications," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 4, pp. 473 –486, April 2009.

[9] A. Sanchez-Macian, P. Reviriego, and J. Maestro, "Enhanced detection of double and triple adjacent errors in Hamming codes through selective bit placement," *Device and Materials Reliability, IEEE Transactions on*, vol. 12, no. 2, pp. 357 –362, June 2012.

[10] S. Lin and J. D. Costello, *Error Control Coding: Fundamentals and Applications*, 2nd ed. Prentice Hall, 2004.

[11] W. Peterson and J. E. Weldon, *Error-Correcting Codes*, 2nd ed. MIT Press, 1972.

[12] W. Adi, "Fast burst error-correction scheme with Fire code," *Computers, IEEE Transactions on*, vol. C-33, no. 7, pp. 613 –618, July 1984.

[13] H. Burton, "Some asymptotically optimal burst-correcting codes and their relation to single-error-correcting Reed-Solomon codes," *Information Theory, IEEE Transactions on*, vol. 17, no. 1, pp. 92 – 95, Jan 1971.

Adam Neale (S'08) received the B.A.Sc. and M.A.Sc. degrees in electrical engineering from the University of Waterloo, Waterloo, Canada, in 2008 and 2010, respectively. He is currently pursuing the Ph.D. degree in electrical engineering also at the University of Waterloo. His research interests are in the area of robust/high performance/low power SRAM design, and ECC circuits. He is an NSERC scholar, holds a Waterloo Institute of Nanotechnology Fellowship, and is a captain of the University of Waterloo varsity men's track and field team.

Manoj Sachdev (M'87-SM97-F'11) received the B.E. degree (with honors) in electronics and communication engineering from University of Roorkee, Roorkee, India, and the Ph.D. degree from Brunel University, Brunel, U.K.

He was with Semiconductor Complex Limited, Chandigarh, India, from 1984 till 1989 where he designed CMOS Integrated Circuits. From 1989 till 1992, he worked in the ASIC division of SGS-Thomson at Agrate (Milan). In 1992, he joined Philips Research Laboratories, Eindhoven, The Netherlands, where he researched on various aspects of VLSI testing and manufacturing. He has been a Professor in the Department of Electrical and Computer Engineering at the University of Waterloo, ON, Canada since 1998. At present, he serves as the department chair and holds the University of Waterloo research chair. His research interests include low power and high performance digital circuit design, mixed-signal circuit design, test and manufacturing issues of integrated circuits. He has contributed to five books, two book chapters, and has authored over 170 technical articles in conferences and journals. He holds more than 30 granted and pending U.S. patents on various aspects of VLSI circuit design, reliability, and test.