Open access • Journal Article • DOI:10.1109/TC.2018.2890651

Two Bit Overlap: A Class of Double Error Correction One Step Majority Logic Decodable Codes — Source link

Pedro Reviriego, Shanshan Liu, Ori Rottenstreich, Fabrizio Lombardi

Institutions: Charles III University of Madrid, Northeastern University

Published on: 01 May 2019 - IEEE Transactions on Computers (IEEE)

Topics: Error detection and correction, Parity bit, BCH code, Code rate and Parity-check matrix

Reviriego, Pedro; Liu, Shanshan; Rottenstreich, Ori; Lombardi, Fabrizio. (2019). Two bit overlap: a class of double error correction one step majority logic decodable codes. *IEEE Transactions on Computers*, 68(5), pp. 798-803.

DOI: https://doi.org/10.1109/TC.2018.2890651

Two Bit Overlap: A Class of Double Error Correction One Step Majority Logic Decodable Codes

Pedro Reviriego, Shanshan Liu, Ori Rottenstreich and Fabrizio Lombardi

Abstract—Error Correction Codes (ECCs) are commonly used to protect memories against soft errors with an impact on memory area and delay. For large memories, the area overhead is mostly due to the additional cells needed to store the parity check bits. In terms of delay, the overhead is mostly needed to detect and correct errors when the data is read from the memory. Most ECCs that can correct more than one error have a complex decoding process and so are limited in high speed memory applications. One exception is One Step Majority Logic Decodable (OS-MLD) codes for which decoding can be done in parallel at high speed. Unfortunately, there are only a few OS-MLD codes that provide a limited choice in terms of block sizes, error correction capabilities and code rate. Therefore, there is considerable interest in a novel construction of OS-MLD codes to provide additional choices for protecting memories. In this paper, a new method to construct Double Error Correction (DEC) OS-MLD codes is presented. This method is based on the use of parity check matrices in which two bits have at most two parity check equations in common; the proposed method provides codes that require a smaller number of parity check bits than existing codes like Orthogonal Latin Square (OLS) codes. The drawback of the proposed Two Bit Overlap (TBO) codes is that they require slightly more complex decoding than OLS codes. Therefore, they provide an intermediate solution between OLS and non OS-MLD codes in terms of decoding delay and number of parity check bits. The proposed TBO codes have been implemented for some block sizes and compared to both OLS and BCH codes to illustrate the trade off in delay and memory overhead. Finally, this paper discusses the generalization of the proposed scheme to codes with larger error correction capabilities.

Index Terms—Memory, Error Correction Codes, Orthogonal Latin Square codes, One Step Majority Logic Decoding

1 Introduction

In the nanoscales, electronic circuits are prone to suffer failures due to a number of phenomena, such as manufacturing defects, aging or radiation induced soft errors [1],[2]. For example, in the case of memories, a soft error can modify the contents of a word causing data corruption, so possibly leading to a system failure. Therefore, in applications that need to operate reliably, memories are commonly protected with Error Correction Codes (ECCs) [3],[4] . These codes add a number of parity check bits to each word to detect and correct errors. The parity checks are computed as part of the write operation and checked when reading from memory. This requires additional logic circuitry for encoding and decoding. The number of additional parity check bits per word and the complexity of the encoding and decoding logic circuitry depend on the number of bit errors that the code can correct [5],[6]. In some scenarios, the error rates can be significant and affect memory cells randomly. For example, in near-threshold caches [7] or in Spin-transfer Torque Magnetoresistive RAMs (STT-MRAMs) [8], there is a need to support multiple bit error correction.

The number of parity bits scales nearly in a linear fashion with the number of bits that can be corrected; this relationship is found for some commonly used codes such as the Bose Chaudhuri Hocquenghem (BCH) codes [9]. However, the decoding complexity for BCH codes increases exponentially when more than one bit per word must be corrected [10]; this occurs for most ECCs and poses a problem for the protection of high speed memories because decoding can become a limiting design factor.

The use of codes that can correct several bits per word with fast parallel decoding to protect memories has been extensively investigated over the last decade; Euclidean Geometry (EG) codes [11], Difference Set (DS) codes [12] and Orthogonal Latin Square (OLS) codes [13],[14] have been optimized for memory protection. In all these cases, fast decoding is achieved by using One Step Majority Logic Decoding (OS-MLD) [9]. OS-MLD performs the decoding of each bit by taking the majority of a small number of parity check equations. This scheme is significantly simpler than for example syndrome decoding in which each bit is compared with the relevant error patterns and when multiple bits must be corrected, a large number of syndrome patterns must be considered. One of the most significant issues with OS-MLD is that only a few codes support it and for each of them, only a few word sizes and error correction capabilities are available.

- *P. Reviriego was with Universidad Antonio de Nebrija at the time of writing this paper, he is currently with Universidad Carlos III de Madrid, Spain (email: revirieg@it.uc3m.es).*
- *S. Liu and F. Lombardi are with the Dept. of ECE, Northeastern University, Boston, USA (email: ssliu@coe.neu.edu; lombardi@coe.neu.edu).*
- *O. Rottenstreich is with the Department of Computer Science and the Department of Electrical Engineering, Technion, Israel (email: or@cs.technion.ac.il).*

For example, for two bit error correction codes (i.e. Double Error Correction (DEC)) and word sizes larger than 10 bits, the only viable design options are given by OLS codes. These codes require a large number of parity check bits and thus, they introduce a significant overhead in terms of memory size.

Therefore, there is a significant interest in finding new code constructions that support OS-MLD as efficient and novel design options [15]. In this paper, a new method to construct DEC codes that support OS-MLD is presented. This new construction relies on the use of parity check matrices whose columns overlap by at most two positions with ones and have a constant weight. This permits to design a modified OS-MLD that is slightly more complex than OLS codes but still having significantly simpler decoding than non OS-MLD codes (such as BCH codes).

The rest of the paper is organized as follows. Section 2 provides an overview of OS-MLD codes focusing on OLS codes that are commonly used to implement DEC for large word sizes. The proposed Two Bit Overlap (TBO) codes are presented in Section 3. Then, two of the proposed codes are evaluated in Section 4 and compared to existing OLS and BCH codes. In Section 5, a brief discussion of the potential extension of the TBO scheme to construct codes that can correct a larger number of errors is presented. Finally, the paper ends with Section 6 that summarizes the conclusion of this work and outlines some topics for future work.

2 One Step Majority Logic Decodable Codes

The use of Error Correction Codes (ECCs) for memory protection has some key differences with their use in communications. The most significant difference is that a memory word is read in parallel and the correct data is expected at the end of the clock cycle. In communication, data is typically received serially and can be decoded on a bit by bit basis. The need to complete the decoding process in one cycle for an entire word makes decoding very challenging. Traditionally, Single Error Correction (SEC) codes have been used to protect memories [3]. In such case, decoding can be done for each bit by just checking if the syndrome matches the corresponding column in the parity check matrix. However, when more than one bit needs to be corrected, such approach, known as syndrome decoding, needs to compare the syndromes of multiple bit errors that include that bit. This leads to a large increase in decoding complexity, specially for large word sizes [10].

To overcome the limitations of syndrome based decoding, One Step Majority Logic Decodable (OS-MLD) codes have been proposed [11] to protect memories. Some OS-MLD codes, such as the Orthogonal Latin Square (OLS) codes were proposed decades ago for memory protection. However, in most cases, they require a large number of parity check bits [13] . Recently, Euclidean Geometry (EG) or Difference Set (DS) codes have been proposed [11],[12],[16]. These codes reduce the number of parity check bits. The significant concern with EG and DS codes is that they support only a few block sizes and error correction capabilities [9]. For example, for Double Error Correction (DEC), EG codes only support (15, 7) and DS only (21, 11), where (n, k) refers to the codeword size n and the data block size k (thus the size of parity check block is n-k). Orthogonal Latin Square (OLS) codes provide a wider range of choices; the parameters of the DEC OLS codes that have k equal to a power of two are shown in Table 1 for k up to 1024.

TABLE 1
PARAMETERS OF SOME DOUBLE ERROR CORRECTION ORTHOGONAL LATIN SQUARE CODES

Data block size k	Parity check block size n-k
16	16
64	32
256	64
1024	128

Double error correction Orthogonal Latin Square codes are built such that their parity check matrices H have the following properties [13]:

1. Each column has a size of $4 \cdot \sqrt{k}$.
2. Each column that corresponds to a data bit has exactly four ones.
3. Each pair of columns only has at most a single position with a one in common (one bit overlap).

Based on these properties, a simple decoding scheme can be designed. In this scheme, for each data bit a majority vote of the four parity check equations that are involved is performed. If the result is one, then the bit is in error and therefore, is thus corrected. This procedure is usually referred to as one step majority logic decoding. It will correct all errors that affect data bits in the presence of single and double errors. If a data bit is in error, another error can only affect one of its parity checks and thus a majority will always occur and the bit will be corrected. Conversely, if the bit is not in error, errors on the other two bits can only affect at most two of its parity checks, so that there will be no majority to start a correction.

As an example, the parity check matrix H for the (32, 16) DEC OLS code is shown in Figure 1, in which the left 16 columns refer to the data bits and the other 16 columns to the parity bits. The OS-MLD decoding of the first data bit is shown in Figure 2, where each parity equation corresponds to one row of the parity check matrix in Figure 1. The required logic circuit is simpler than checking all possible two bit error patterns for a bit. This feature is more advantageous when the codeword size is large, because that number of double bit error patterns is proportional to n.

$$\left[\begin{array}{c|c}
1111000000000000 & 1000000000000000 \\
0000111100000000 & 0100000000000000 \\
0000000011110000 & 0010000000000000 \\
0000000000001111 & 0001000000000000 \\
1000100010001000 & 0000100000000000 \\
0100010001000100 & 0000010000000000 \\
0010001000100010 & 0000001000000000 \\
0001000100010001 & 0000000100000000 \\
1000010000100001 & 0000000010000000 \\
0100100000010010 & 0000000001000000 \\
0010000110000100 & 0000000000100000 \\
0001001001001000 & 0000000000010000 \\
1000001000010100 & 0000000000001000 \\
0100000100101000 & 0000000000000100 \\
0010100001000001 & 0000000000000010 \\
0001010010000010 & 0000000000000001
\end{array}\right]$$

Figure 1. Parity check matrix H of the (32, 16) DEC OLS code.

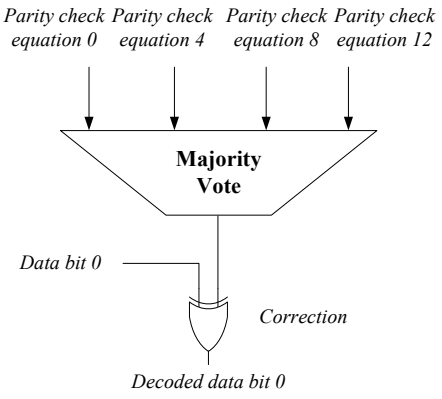

Figure 2. Majority logic decoding of data bit 0 in the (32, 16) DEC OLS code.

As discussed in the introduction, the main drawback of OLS codes is that they require a large number of parity check bits. For example, for k = 256, a DEC OLS code requires 64 parity check bits compared to the 18 bits required by a DEC BCH code. Therefore, code constructions that support OS-MLD while requiring a smaller number of parity check bits are of significant interest. In the next section, a new construction for such codes is presented.

3 Two Bit Overlap Codes

This section presents the proposed Two Bit Overlap (TBO) codes; the parameters of these codes are also analyzed and established. The first subsection describes the features of the matrices that are used to construct the codes and shows how DEC OS-MLD codes can be obtained from them. The second subsection presents the method to construct the matrices, while the third subsection summarizes the parameters of the proposed codes.

3.1 Matrix Features

As discussed in the previous section, a method for constructing Double Error Correction (DEC) One Step Majority Logic Decodable (OS-MLD) codes requires to design matrices with columns such that:

1. Each column has exactly four ones.
2. Each pair of columns only has at most a position with a one in common.

Orthogonal Latin Squares with double error correction are a particular case of the above construction. Then this matrix can be used to form the parity checks such that each data bit corresponds to a column and each row to a parity check bit and each data bit participates in the parity checks for which it has a one in the column.

As explained previously, the OS-MLD property is simple, because each data bit participates in four parity checks and the other bits share at most one parity check with it. Therefore, a majority of the four is obtained when the bit is in error and another bit is also in error. Conversely, when the bit is not in error, two errors on other bits can affect at most two of the parity checks of the initial bit and thus no miscorrection will occur.

Consider a construction in which the matrices are given such that:

1. Each column has exactly seven ones.
2. Each pair of columns only has at most two positions with a one in common.

Then, the code will be OS-MLD for DEC by taking a majority of at least five on the seven equations for the participating bit. The possible cases are as follows:

1. An error free bit and two other bits in error give a worst case of four parity check errors on the error free bit so there is no majority and no miscorrection.
2. A bit in error and another bit in error cause at least five parity check errors on the erroneous bit, so it will be corrected.

A disadvantage of this construction is that the encoding and decoding are more complex than for a DEC OLS code. This is due to two features:

1. The matrix has a larger number of ones (seven per column instead of four) and thus, more logic is needed to compute the parity checks.
2. The majority voting is done over seven parity checks and not over four which is again more complex.

However, the decoding is still significantly simpler than for a non OS-MLD code as detailed in the evaluation section of this paper. To have advantage over existing DEC OLS codes, the proposed codes need to have a lower number of parity check bits. This will be the case for the proposed scheme as shown next.

3.2 Polynomial based Matrix Construction

Matrices with the properties specified in the previous subsection can be formed as follows. We associate each bit with index b with a polynomial P_b of degree two, such that each of its three coefficients belong to $\left\lfloor 0, \sqrt[3]{k}-1 \right\rfloor$, where k is the number of data bits in the codeword. The coefficients are selected such that $P_b(x) = \sum_{i=0}^{2} a_i \cdot x^i$ satisfies $P_b\left(\sqrt[3]{k}\right) = \sum_{i=0}^{2} a_i \cdot \left(\sqrt[3]{k}\right)^i = b$. Note that there is a single option for the selection of a_0, a_1, a_2. For instance, a_0 equals b mod $\sqrt[3]{k}$, $a_1 = \left((b-a_0)/\sqrt[3]{k}\right)$

mod $\sqrt[3]{k}$ and $a_2 = \left(\left(b - a_0 - a_1 \cdot \sqrt[3]{k}\right)/\left(\sqrt[3]{k}\right)^2\right) \bmod \sqrt[3]{k}$. This is illustrated with the following example. Let consider $k = 7^3 = 343$ and $b = 51$. Then the coefficients of the polynomial are $a_o = 51 \; mod \; 7 = 2$; $a_1 = (51-2)/7 \; mod \; 7 = 0$ *and* $a_2 = (51 - 2 - 0 \cdot 7)/7^2 \; mod \; 7 = 1$ so that $P_{51}(x) = 2 + x^2$ which satisfies $P_{51}\left(\sqrt[3]{k}\right) = P_{51}(7) = 2 + 7^2 = 51$.

We then characterize each bit b by the seven values for $x \in [0,6]$ of the polynomial $P_b(x)$, such that calculations are performed over modulo $\sqrt[3]{k}$.

In our example, for $b = 51$ those values would be {2,3,6,4,4,6,3}. We describe each such value in $\sqrt[3]{k}$ bits such that a single bit that corresponds to the value, is equal to one while all others are zero. Similarly, we describe the ordered list of values in a binary vector of length $7 \cdot \sqrt[3]{k}$. In our example, this list would be {0010000,0001000,0000001,0000100,0000100, 0000001,0001000}. This vector will be the column of the data bit b in the parity check matrix and has exactly seven ones (satisfying the first condition).

Next, it will be shown that when $\sqrt[3]{k}$ is a prime number greater or equal than seven, then the second condition stated in the previous subsection is also met. This condition states that two columns of the matrix can have at most two ones in common. The proof is based on Lagrange´s theorem in number theory that states that a polynomial of degree $n \geq 1$ modulo, a prime number p with integer coefficients not divisible by p has at most n distinct roots [17]. This is the case for the polynomials used in the code construction as the coefficients are modulo $\sqrt[3]{k}$ that is a prime and not divisible by $\sqrt[3]{k}$ and they are evaluated over $x \in [0,6]$ which are also smaller than $\sqrt[3]{k}$. Now, suppose that two bits b and c have more than two ones in common in the parity check matrix. This means that the two polynomials $P_b(x)$, $P_c(x)$, each of degree two collide in at least three distinct values. This means that $P_b(x)$-$P_c(x)$ which is also a polynomial of degree two, has three roots which according to Lagrange´s theorem is not possible. Therefore, this implies that P_b is exactly equal to P_c. Accordingly, $b = c$ and the two bits are the same which contradicts the assumption that these bits were different.

Therefore, the construction just described will generate matrices with columns that have exactly seven ones and that share at most two ones. Let us summarize the construction procedure:

1. Select a block size k such that $\sqrt[3]{k}$ is a prime larger than six.
2. For each of the k bits assign an index b = 0,1,2,….,k-1.
3. Compute the polynomial $P_b(x) = \sum_{i=0}^{2} a_i \cdot x^i$ that satisfies $P_b\left(\sqrt[3]{k}\right) = b$ with coefficients that belong to $[0, \sqrt[3]{k} - 1]$.
4. Compute the values of the polynomial $P_b(x)$ modulo $\sqrt[3]{k}$ for $x \in [0,6]$.
5. Assign a $\sqrt[3]{k}$ bit array to each of the values obtained in 4 such that the bit that corresponds to the value is '1' and all the other bits are '0'.
6. The column of the parity check matrix for that bit is formed by the concatenation of the seven arrays obtained in step 5.

TABLE 2
PARAMETERS OF THE PROPOSED DEC CODES AND OLS CODES

Type	Prime number p	Data block size k	Parity check block size n-k
OLS	-	256	64
TBO	7	343	49
Shortened TBO	7	256	48
OLS	-	1024	128
TBO	11	1331	77
Shortened TBO	11	1024	75
TBO	13	2197	91

The codes obtained for a given prime number p, have the following parameters: $k = p^3$ and $n - k = 7 \cdot p$. This compares favorably to OLS codes that have $k = p^2$ and $n - k = 4 \cdot p$, when p is large.

3.3 TBO Codes

The parameters of the TBO codes obtained with the polynomial construction are summarized on Table 2; this table also shows the parameters of the DEC OLS codes of similar word lengths; only few word sizes are supported starting at k = 343. This could be used to protect 256 bit words by shortening its H matrix. After removing 87 columns that share most positions for "1" on each row from the H matrix, one parity bit can be saved, thus the proposed code requires 48 parity check bits compared to 64 for a DEC OLS code. The next block size, k = 1331 can be used to protect words of 1024 bits (as applicable to some configurations for cache memory). In this case, the proposed code can be shortened by saving 2 parity bits and it requires 75 parity bits compared to 128 for a DEC OLS code.

4 EVALUATION

The proposed TBO codes shortened to 256/1024 data bits have been implemented in HDL and mapped to a 65nm library from TSMC using Design Compiler. Then, protected memories of varying sizes have been implemented as examples to evaluate the overheads of the schemes. Existing DEC OLS codes and DEC BCH codes have also been implemented to show the benefits of the proposed codes.

The settings of synthesis were selected to put "maximum" effort on optimizing area overhead, power, and delay to show the benefit in area/power that can be achieved by the fault tolerant memories, and the "minimum" latency that can be achieved by the encoders and decoders.

Overheads in terms of area and power consumption introduced by the ECCs to the unprotected memory include two components: the redundant memory overhead, and the encoder/decoder overhead. For a small memory

TABLE 3
AREA SYNTHESIS RESULTS (μm^2) COMPARISON FOR UNPROTECTED MEMORIES AND PROTECTED BY DIFFERENT OPTIONS

	SRAM size	Unprotected	OLS	TBO	BCH
	Enc/Dec	N.A.	1.4E4	2.6E4	3.2E5
	1K words	5.3E5	6.8E5	6.6E5	8.9E5
k=256	2K words	10.8E5	13.6E5	13.1E5	14.8E5
	4K words	21.6E5	27.1E5	25.9E5	26.3E5
	8K words	43.2E5	54.1E5	51.6E5	49.5E5
	Enc/Dec	N.A.	5.6E4	1.0E5	4.0E5
	1K words	2.1E6	24.6E5	23.9E5	63.8E5
k=1024	2K words	3.9E6	44.6E5	43.0E5	81.9E5
	4K words	7.8E6	88.7E5	85.1E5	12.2E6
	8K words	15.7E6	17.7E6	16.9E6	20.2E6

TABLE 4
POWER SYNTHESIS RESULTS (mW) COMPARISON FOR UNPROTECTED MEMORIES AND PROTECTED BY DIFFERENT OPTIONS

	SRAM size	Unprotected	OLS	TBO	BCH
	Enc/Dec	N.A.	8.7	11.0	35.1
	1K words	22.2	39.5	37.4	58.8
k=256	2K words	31.3	49.9	47.9	68.2
	4K words	56.0	83.7	82.2	83.5
	8K words	102.5	133.5	125.8	123.0
	Enc/Dec	N.A.	26.1	32.0	112.0
	1K words	73.1	110.5	108.3	186.7
k=1024	2K words	115.4	155.9	152.9	229.9
	4K words	206.1	258.0	253.3	322.5
	8K words	353.6	423.9	411.5	473.2

size, the overhead introduced by the different ECCs is mostly due to the encoder/decoder overhead. However, for larger memory sizes, the encoder/decoder overhead is smaller relative to the increase in memory size; hence, the redundant memory overhead (that is related to the number of parity bits) tends to dominate. By combining these two components of the overhead, different classes of ECCs may have lower area/power overhead when protecting different memory sizes. To compare the overhead introduced by different ECCs, the area and power consumption for unprotected memories and memories employing different coding options are given in Table 3 and Table 4. These tables include also the area and power dissipation of the encoder and decoder with no memory array so that the complexity of the encoding and decoding circuitry can be compared directly. It can be observed that the BCH codes need significantly larger area and power for encoding and decoding than both OLS and TBO codes. Instead, the proposed codes require less than 2x the area of the OLS codes and less than 1.3x the power. To illustrate the results for the entire system for different memory sizes, the relative overheads over the unprotected memory in terms of percentage are shown in Figure 3 to Figure 6.

Figure 3 shows the optimized synthesis results for the area overhead introduced by each ECC over memory size; the proposed TBO codes have the smallest area overhead in the range from 1K to 4K memory words, and then they are better than OLS codes, but worse than BCH codes. As discussed previously, OLS and TBO codes (which are both OS-MLD codes) have a lower decoding complexity than BCH codes (non OS-MLD codes). BCH codes have the smallest number of parity bits that can reduce the redundant memory bits. Therefore, in this case, the savings for the encoder/decoder make OS-MLD codes better than BCH codes when the size of the memory is smaller than or equal to 4K words. Savings in the redundant memory overhead for the BCH codes offset the disadvantage of a complex decoder and begin to have a compelling role when the memory size increases to 8K words.

Compared to OLS codes, the advantage of a smaller number of parity bits required by the proposed TBO codes also offsets its larger decoder circuit. Results for k = 1024 bit are shown in Figure 4, and in this case, the TBO codes have the lowest area overhead for memories with 1K to 8K words.

In terms of power consumption, results are shown in Figure 5 for k = 256 and in Figure 6 for k = 1024; they have the same trends as for the area overhead, i.e., the proposed codes are best for 1K to 4K words with 256 bits and 1K to 8K words with 1024 bits, respectively.

Additionally, these coding schemes have been compared for smaller memory sizes to determine the largest size for which OLS codes have a lower area/power over-

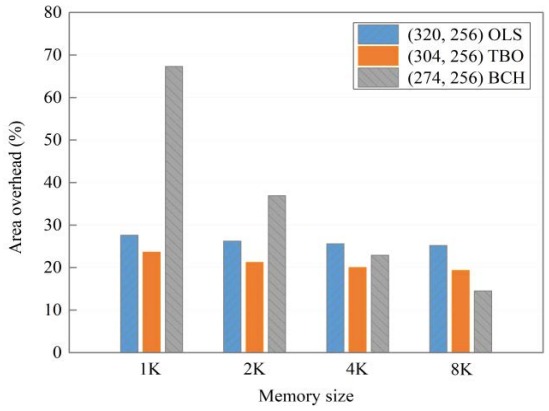

Figure 3 Area comparison for different size (in words) of SRAMs with k=256 employing different ECCs.

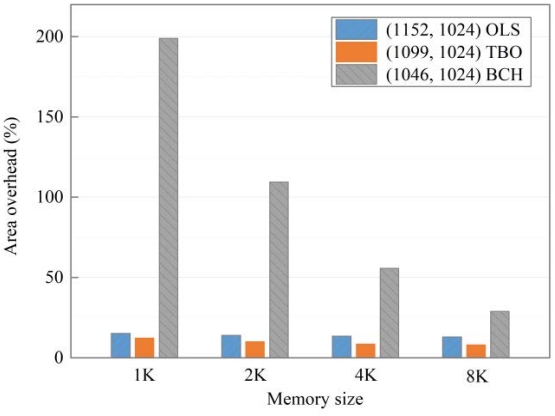

Figure 4 Area comparison for different size (in words) of SRAMs with k=1024 employing different ECCs

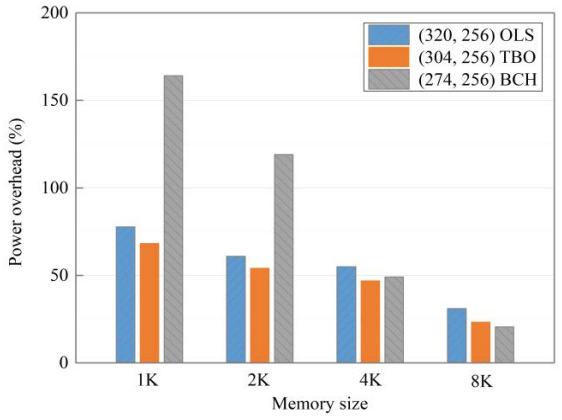

Figure 5 Power comparison for different size (in words) of SRAMs with k=256 employing different ECCs.

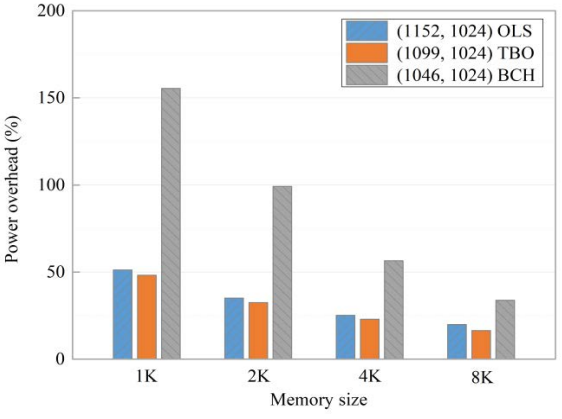

Figure 6 Power comparison for different size (in words) of SRAMs with k=1024 employing different ECCs.

head than the TBO codes. This size is 128 words in both cases of 256 and 1024 bits. This shows that the proposed TBO scheme reduces the memory overhead compared to OLS codes for practical memory sizes.

Results for the delay required by the encoders and decoders for a delay optimized synthesis are presented in Table 5. The OLS codes have the fastest encoding speed, while the BCH codes have the slowest encoding speed. This occurs because the largest number of "1" on the rows of the H matrix for OLS codes is smaller than the numbers for the other codes. In the case of decoding latency, OLS codes also show better results than the other codes, because they have the lowest complexity for the decoding algorithm. However, when the proposed TBO codes are slower than the OLS codes (for example 20.2% slower for k = 256), then they are still significantly faster than the BCH codes (for example by a 46.6% saving when k = 256).

Overall, the results show that for memories up to a few thousand words, the proposed TBO codes can provide a fast decoding latency with a reduced memory overhead. For larger memories, the proposed codes can still reduce the delay, but they will incur in a larger area overhead compared to BCH codes, but still lower than for OLS codes. These features can be useful for applications in which OLS codes have some delay margin over requirements that non OS-MLD codes (such as BCH) cannot meet. When selecting an ECC for a specific application, designers can check the delay first. If non OS-MLD codes cannot meet the delay restrictions, the proposed codes can be used to reduce the memory overhead introduced by having a lower number of parity check bits than OLS codes.

TABLE 5
DELAY SYNTHESIS RESULTS (*ns*) COMPARISON FOR ENCODERS AND DECODERS

Type		OLS	TBO	BCH
k =256	Encoder	0.54	0.66	0.76
	Decoder	0.99	1.19	2.23
k=1024	Encoder	0.65	0.77	1.00
	Decoder	1.11	1.38	2.63

5 GENERALIZATION TO LARGER NUMBER OF ERRORS

The proposed TBO construction can be generalized to obtain codes that correct up to t bit errors. Consider a construction in which the matrices are such that:

1. Each column has exactly w ones.
2. Each pair of columns only has at most two positions with a one in common.

Then, for such a code to be able to correct t errors using OS-MLD with a threshold for correction of m the following conditions are needed:

1. To avoid erroneous corrections on data bits $m > 2 \cdot t$.
2. To ensure correction of bits in error $w - 2 \cdot (t-1) \geq m$.

From which $w \geq m + 2 \cdot (t-1)$. Therefore, setting $m = 2 \cdot t + 1$ and $w = 4 \cdot t - 1$, an OS-MLD code that can correct t errors is obtained.

The parameters for such codes in terms of the number of elements in the vote w and the majority threshold m are given in Table 6. The table also shows the corresponding values for OLS codes; as t increases, the majority voting for TBO codes becomes significantly more complex. The proposed polynomial based construction presented for DEC TBO codes could potentially be extended to the general case of t bit correcting codes; however, the word sizes would be even larger than those obtained in the DEC case, so making them of interest only for the protec-

TABLE 6
MAJORITY LOGIC DECODING PARAMETERS OF TBO CODES

Type	Error correction capability t	Number of elements in the vote w	Majority threshold m
TBO	2	7	5
OLS	2	4	3
TBO	3	11	7
OLS	3	6	4
TBO	4	15	9
OLS	4	8	5
TBO	5	19	11
OLS	5	10	6

tion of very wide memories. The construction and evaluation of TBO codes with larger error correction capabilities is beyond the scope of this paper and left for future work.

6 Conclusion and Future Work

This paper has presented a new construction for Double Error Correction (DEC) One Step Majority Logic Decodable (OS-MLD) codes based on Two Bit Overlap (TBO). The codes obtained provide a trade-off between the number of parity check bits and the decoding complexity. They are significantly simpler and faster to decode than non OS-MLD codes such as BCH codes but slightly more complex than existing DEC OS-MLD codes (such as Orthogonal Latin Square (OLS) codes). Also, they require a smaller number of parity check bits than OLS codes but more than BCH codes. Therefore, they provide additional choices to memory designers in terms of decoding speed and memory overheads.

As applicable to also most OS-MLD codes, the proposed TBO codes have limited choices in terms of word sizes, two practical configurations for memory protection of 256 and 1024 bit words are used by shortening the codes. These two options have been implemented and compared to DEC OLS and BCH codes. The results show that the proposed TBO codes achieve the smallest area overhead and power consumption for memories with 1K to 4K words in the case of 256 bits and for memories with 1K to 8K words in the case of 1024 bits. For larger memories, the overhead of the proposed codes are higher than BCH codes but still lower than OLS codes. In terms of delay, the proposed codes are slower than OLS codes but significantly faster than BCH codes. Therefore, the proposed codes incur in a reduced area overhead and power consumption for memories with tight requirements in speed that cannot currently be met by BCH codes.

The work presented in this paper is being investigated and extended for future research for designing TBO codes that can correct more than two bit errors. Another area for future work is to find alternative matrix constructions that provide additional choices in term of block sizes and number of parity check bits for DEC codes.

References

[1] N. Kanekawa, E. H. Ibe, T. Suga and Y. Uematsu, “Dependability in electronic systems: mitigation of hardware failures, soft errors, and electro-magnetic disturbances," (Springer Verlag, New York, USA, 2010)

[2] S. Dolev, Y.A. Haviv, “Self-stabilizing microprocessor: analyzing and overcoming soft errors,” IEEE Transactions on Computers, vol.55, no.2, pp.385-399, Apr. 2006.

[3] C. L. Chen and M. Y. Hsiao, “Error-correcting codes for semiconductor memory applications: a state-of-the-art review," IBM J. Res. Develop., vol. 28, no. 2, pp. 124-134, Mar. 1984.

[4] S.C. Krishnan, R. Panigrahy, S. Parthasarathy, “Error-correcting codes for ternary content addressable memories,” IEEE Transactions on Computers, vol.58, no.2, pp.275-279, Feb. 2009.

[5] E. Fujiwara, “Code design for dependable systems: theory and practical application," John Wiley & Sons, Inc., Hoboken, New Jersey, 2006.

[6] J. Li, P. Reviriego, L. Xiao and C. Argyrides, "Extending 3-bit Burst Error Correction Codes with Quadruple Adjacent Error Correction", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 2, pp. 221-229, Feb. 2018.

[7] T. Miller, R. Thomas, J. Dinan, B. Adcock, and R. Teodorescu, “Parichute: Generalized Turbocode-Based Error Correction for Near Threshold Caches”, in MICRO, pp. 351–362, 2010.

[8] B. Del Bel, J. Kim, C. H. Kim, and S. S. Sapatnekar, “Improving STT-MRAM density through multibit error correction”, in Proceedings of the conference on Design, Automation & Test in Europe (DATE '14), 2014.

[9] S. Lin and D. J. Costello, “Error Control Coding, 2nd ed. Englewood Cliffs," NJ, USA: Prentice Hall, 2004.

[10] R. Naseer and J. Draper, “DEC ECC design to improve memory reliability in sub 100 nm technologies," In Proc. IEEE ICECS, pp. 586-589, 2008.

[11] S. Ghosh and P. D. Lincoln, “Dynamic low-density parity check codes for fault-tolerant nano-scale memory" in Proc. Foundations of Nanoscience (FNANO07), Snowbird, UT, 2007.

[12] P. Reviriego, M. Flanagan, S. Liu, J.A. Maestro, “Multiple Cell Upset Correction in Memories Using Difference Set Codes," IEEE Transactions on Circuits and Systems I, vol. 59, no. 11, November 2012, pp. 2592-2599.

[13] M. Y. Hsiao, D. C. Bossen, and R. T. Chien, “Orthogonal Latin square codes," IBM J. Res. Develop., vol. 14, no. 4, pp. 390–394, Jul. 1970.

[14] K. Namba and F. Lombardi, “Concurrent Error Detection of Binary and Nonbinary OLS Parallel Decoders" IEEE Trans, on Device and Materials Reliability, vol. 14, no. 1, pp. 112-120, March 2014.

[15] S. Liu, J. Li, P. Reviriego, M. Ottavi, "A Double Error Correction Code for 32-bit Data Words with Efficient Decoding", IEEE Transactions on Device and Materials Reliability, vol. 18, no. 1, pp. 125-127, March 2018.

[16] H. Naeimi and A. DeHon, “Fault secure encoder and decoder for nanomemory applications,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 4, pp. 473–486, Apr. 2009.

[17] R. Friedberg, “An Adventurer's Guide to Number Theory”, New York, Mc Graw-Hill, 1968.