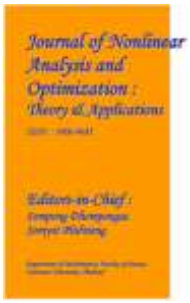
**J**ournal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 13, No. 1, (2022),
ISSN : **1906-9685**




# EFFICIENT MULTIPLE ERROR CORRECTION BY USINGTWO BIT OVERLAP CODES

**C. RAMAMOHAN**, ASSISTANT PROFESSOR rammohan.mohan966@gmail.com
**L. RANGA SWAMY**, ASSISTANT PROFESSOR ranar404@gmail.com
**M. SATEESH KUMAR**, ASSISTANT PROFESSOR steeshkumar9f@gmail.com
Department of ECE, Sri Venkateswara Institute of Technology, N.H 44,Hampapuram, Rapthadu, Anantapuramu, Andhra Pradesh 515722

**Abstract:** The most common way to prevent mistakes in memory is to use error correcting codes. For a long time, OLS codes have relied on linear block codes with single-error correction and double-error detection. Orthogonal Latin square (OLS) codes are one kind of one-step-majority-logic-decodable (OS-MLD) error correcting codes. The decoding process is quick and simple with these codes. In order to fix the radiation-induced soft errors in semiconductor memory, OLS codes are used to fix failures in multiple cells.Latin squares are used to extract OLS codes, which can be effectively implemented on reconfigurable architectures like FPGAs. In this research, we define the parity regulation matrices and suggest a way to reduce the decoding block by enlarging the real OLS code using Latin Square codes. This article details the steps to reduce the decoding block by increasing the real size of an orthogonal Latin square code (OLS) and how to build these codes using their parity control matrices. In order to address the generalisation, this research narrows the scope of the suggested method to codes that have better error correcting capabilities.

## I.INTRODUCTION

Several processes, including manufacturing flaws, ageing, and soft mistakes generated by radiation, may cause electronic circuits on the nanoscale to fail. As an example, a memory might fail due to data corruption caused by a soft mistake that changes the meaning of a word. Due to the importance of memory security in mission-critical applications, error correcting codes (ECCs) are often used. To identify and correct errors, these codes augment each word with a certain amount of parity bits. In both the write and read operations from memory, the parity checks are calculated. The encoding and decoding processes need extra logic hardware. The amount of bit mistakes that the code can fix is affected by the intricacy of the encoding and decoding logic circuits and the number of additional parity check bits per word. There are cases when the mistake rates are large and impact memory cells at random. For instance, in Spin-transfer or near-threshold caches.

For support of multiple bit error correction, STT-MRAMs (Torque Magneto-resistive Random Access Memory) are necessary.There is a correlation between the amount of bits of parity and the number of fixable defects for some widely used codes, such as the Bose ChaudhuriHocquenghem (BCH) codes. However, BCH codes' decoding complexity increases when bit-per-word corrections of more than one

http://doi.org/10.36893/JNAO.2022.V13I1.1973-1982

bit are required.

rises rapidly; this is true for the vast majority of error correction codes; and it presents a challenge to the maintenance of modest memory, despite the fact that decoding may become a limiting design feature.

In order to safeguard memories, researchers have spent the last hundred years studying various codes that can fix a few bits for every word of fast concurrent decoding. These codes include codes for Orthogonal Latin Square, Difference Set, and even Euclidean Geometry. Fast decoding is accomplished in all of these scenarios utilising Another Step Majority Logic Decoding. OS-MLD decodes each bit by enduring a few of equations that check for parity.When compared to syndrome decoding, which requires a huge number of syndrome patterns to be evaluated when correcting several bits at once and compares each bit to the appropriate error patterns, the whole system is much simpler. The most serious issue with OS-MLD code is that it is only compatible with a small subset of codes, each of which has limited word sizes and error correcting capabilities.

For two-bit error correction codes (i.e., Double Error Correction, or DEC) and word sizes larger than ten bits, for example, the only practical design choices are those supplied by OLS codes.
Such codes provide a large memory overhead due to the large number of parity check bits they include.

There is undoubtedly a strong potential, and thus in uncovering new code installs that stimulate OS-MLD as effective and creative design possibilities. With the use of OS-MLD, this work introduces a novel approach to building Double Error Correction codes. This novel framework relies on parity control matrices with constant-weight columns that overlap no more than once or twice.That makes it possible to create an enhanced OS-MLD, which is much more intricate than Orthogonal Latin. Square codes, although they still outperform non-OS-MLD codes like BCH codes in terms of decoding performance.

Memory security also makes use of OLS codes, which were established in prior generations as part of the plan to safeguard interconnect and cache. In Orthogonal Latin Square codes, each block consists of k bits for data and tm bits for parity. This is where an integer m and a number t represent the number of mistakes fixed. Both m and word dimensions are often powers of two when dealing with memories. One major benefit of OLS coding is how cheap and easy it is to decode them.This is why OS-MLD was able to decode OLS codes.As an example, consider OLS codes and difference set codes (DS codes).Orthogonal Latin Square codes have historically evolved to include the following features: 1) all data bits are connected to exactly 2t parity control bits; 2) all other data bits are connected to no more than like those parity control bits.

The proposed SEC-DED-DAEC codes are derived from DEC-OLS codes, which stand for Double Error Correction Orthogonal Latin Square codes. To begin, using the parity check matrix as a starting point, remove the m parity check bits that correspond to one of the Mi matrices. Each equation for the deleted parity review will use communication data bits within the reduced matrix that do not constitute a parity check. In addition, Figure 1 labels these m-bit groups as g1, g2, g3, and g4. And so, removing M1 does not result in an exchange of parity check bits. In the same way, bits 5-8 (g2), 9-12 (g3), and 13-16 (g4) all belong to different pairs. The updated matrix includes three parity tests that cover all data bits.Therefore, both single-bit and double-bit faults may be fixed with the development of a majority vote to decode the bits.

http://doi.org/10.36893/JNAO.2022.V13I1.1973-1982

### II.EXISTING METHOD

The limitations of these codes also serve to analyse and develop Two Bit Overlap (TBO) codes. The first part of the paper explains how to get DEC OS-MLD codes by looking at the properties of the matrices that were used to create them. Subsection 2 introduces the process for building the matrices, while Subsection 3 contains the parameters of the suggested codes.

Feature Matrix
A technique for constructing DEC One Step Majority
As mentioned before, in order to create Logic Decodable (OS-MLD) codes, it is necessary to build column matrices in such a way that:

1. Each row has precisely four columns.2. Each set of columns shares no more than one place with any other set.

The design described above is shown by Orthogonal Latin Squares with double error correction.Because every data bit is anticipated to participate in the parity checks for which it has one of these in the column, the whole matrix might be used to build parity checks. This way, each data bit corresponds to a column and each row to a parity check bit.

As mentioned before, the Orthogonal Square-Majority Logic Decodable property is simple because all bits in the input data undergo four parity tests, whereas the output bits undergo just one.In a similar vein, an error may still occur; however, no errors can occur until two parity checks on the initial bit generate two mistakes on the other bits.

Consider a building in which the matrices are given in such a way that:

1. There are exactly 7 columns each.2. With one in particular, each pair of columns has just two positions at most.

Later on, the code will be OS-MLD for DEC by taking the majority of at least five of the seven equations for the participating bit. The following are examples of likely scenarios:
1. The error free bit has a worst-case scenario of four parity check failures, which eliminates plurality and wrongdoing, thanks to a free error bit and two additional bits that are in error. 2. It will be repaired since at least five parity check faults on the wrong bit are created by two bits that aren't accurate. The increased difficulty in encoding and decoding compared to a DEC OLS code is one drawback of this design. This is because of two things: One, more reasoning is needed to assess parity tests due to the increased amount of matrices (seven per column instead of four) in the matrix.2. The majority vote is conducted after seven parity tests, but no more than four are allowed, which is much more complicated.
Nevertheless, as discussed in the paper's assessment section, decoding remains much easier than for a language that is not OS-MLD. To be competitive with existing DEC OLS codes, the suggested codes must use less parity check bits. Next, we'll demonstrate that this holds true for the proposed plan.

***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 $[0,\sqrt[3]{k}\text{-}1]$, where $k$ is the number of data bits in the code word. The coefficients are selected such that $P_b(x) = \sum_{i=0}^{2} a_i . x^i$ satisfies $P_b(\sqrt[3]{k}) = \sum_{i=0}^{2} a_i.(\sqrt[3]{k})i = b$. Note that there is a single option for the selection of $a_0$, $a_1$, $a_2$. For instance, $a_0$ equals $b$ mod $\sqrt[3]{k}, a_1 = ((b - a_0)/\sqrt[3]{k}) \bmod \sqrt[3]{k}$ and $a_2 = ((b - a_0 - a_1.\sqrt[3]{k})/(\sqrt[3]{k}))$. This is illustrated with the following example. Let consider $k=7^3 = 343 and\ b = 51$.

http://doi.org/10.36893/JNAO.2022.V13I1.1973-1982

Then the coefficients of the polynomial are$a_0 = 51 \ mod \ 7 = 2; a_1 = \frac{51-2}{7} mod \ 7 = 0 \ and \ \frac{a2(51-2-0.7)}{} mod \ 7 = 1$ so that $P\ (x) = 2 + x^2$

$7^2$ 51

which satisfies$P_{51}(\sqrt[3]{k}) = P_{51}(7) = 2 + 7^2 = 51.$

We then describe every b bit by the seven polynomial values such that modulo$\sqrt[3]{k}$ calculations are done. In our example, such values will be {2, 3, 6, 4, 4, 6, 3} with bb = 51. Each one of these values$\sqrt[3]{k}$ is represented. Bits in such a way that a single bit corresponding to a value is equivalent to one, while all the others are zeros. Likewise, the ordered sequence of values is described into a binary vector length

$\sqrt[3]{k}$.

That list would be {0010000, 0001000,0000001,0000100,0000100, 0000001, 0001000} in our example. In the parity search matrix, this vector will be the column of the data bit b and has exactly seven (satisfying the first condition). 3k 37k.

Further, when a prime number is greater than or equal to seven, it will be shown that the second condition specified in the previous subsection is also met. This condition states that two matrix columns must have two columns in general at most. The statements are based on Lagranges theorem in number theory , which states that a polynomial of degree n>1 modulo, a prime number p with integer coefficientswhich are not divisible by p, has at most n individual roots.

For the polynomials used in the construction of the code, this is the case because the coefficients are modulo that is a prime and not divisible by and are calculated over which they are also smaller than . Now, assume that two bits b andc in the parity check matrix have more than two bits in general. This indicates that at least three different values overlap between the two polynomials Pb(x), Pc(x), each degree two.Which polynomial with degree 2 , has three roots that are not possible as per Lagranges theorem. eThis means, therefore that Pb is completely quavalent to Pc. consequently, B=c

and the two bits are same which eliminates the hypothesis that these bits were different.

The building just mentioned would therefore producematrices with columns which have exactly seven columns and that share two at most. Let us summarize the mechanism of construction:

1. Choose a block size *k* such that $\sqrt[3]{k}$ is a prime more than six.2. For every position of*k* bits allocate an index $b$ = 0, 1, 2... *k*-1.3. Compute the polynomial$P_b(x) =$

$\sum_{i=0}^{2} a_i . x^i$which satisfies$P_b(\sqrt[3]{k})$=bwith coefficients that belong to

$[0,\sqrt[3]{k} - 1]$.4. Compute the values of the polynomial $\sqrt[3]{k}$ for x $\in$ [0,6].5. Assign a $\sqrt[3]{k}$bit array to each of the values implies that Pb(x)-Pc(x), which is a

http://doi.org/10.36893/JNAO.2022.V13I1.1973-1982

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.

For such a specified prime number p, the codes acquired get the following parameters: and n-k=7. P. Which compares correlates to codes which have OLS & when p is big, n-k=4.p.

***TBO Codes***

A summary of the TBO code requirements obtained using the polynomial setup is provided. Just a small number of word sizes starting at k = 343 are supported, but you can also check the specifications of DEC OLS codes with comparable word lengths in this table. Reducing the size of the H matrix might make this useful for 256-bit word security.By removing the 87 columns that take up no more than one spot from each H matrix, we can keep the single parity bit. corresponding to a DEC OLS code's 64 bits, the suggested code necessitates 48 bits for parity checking. For certain cache memory configurations, the following block size, k = 1331, is suitable for protecting words with a length of 1024 bits. To simplify the code and allow 75 parity bits instead of 128 for DEC OLS code, we may keep 2 parity bits.

## III. PROPOSED METHOD

Communication often involves the acquisition of data in a serial fashion, which allows for bit-by-bit decoding. Decoding is challenging since it requires finishing the procedure for a whole word in a single step. The usage of Single Error Correction (SEC) codes has been standard practice for memory preservation. It is possible to decode each bit in this scenario by comparing the syndrome to its matching column in the parity check matrix.

However when it is necessary to fix more than one bit, such

an approach called as syndrome decoding, wants to evaluate the various bit error syndromes that contain the bit. Which

1. The size of each column. 7. k 2. Every column referring to a bit of data has exactly seven columns.3. Every set of columns only has two positions as a maximum, with one in default (two bit overlap).

***Construction of the Parity Check Matrix:***

Codes derived from Latin squares are known as orthogonal Latin square codes. Latin squares of order m are square arrays of dimensions m* m, where each row and column may contain any number from the set {0, 1, 2... m-1}, and each element can only appear once in each row or column. The equation k= m2 holds true for safe data of length k bits unless the OLS code is being derived from Latin squares with dimension m. On the other hand, the t-error for OLS code modification uses 2tm parity bits.

$M1$ 1

leads sig cant increase in the to a nifi complexities of

http://doi.org/10.36893/JNAO.2022.V13I1.1973-1982

$$H = \begin{matrix} I \\ I \\ L \end{matrix} \quad \begin{matrix} M2 & I_{2tm} \\ . & I \\ . & \\ M3 & \end{matrix} \quad I$$

decoding, particularly for large word sizes. Because syndrome-based decoding has its limits, One Step Majority Logic Decodable (OS-MLD) codes have been created to safeguard memory. Orthogonal Latin Square codes and other OS-MLD algorithms were proposed for memory security decades ago. But there are situations when parity checking requires a lot of bits. By using these codes, we can reduce the amount of bits needed for parity checks. The fact that DS and EG codes only handle a small subset of block sizes and lack error correcting capabilities is their biggest drawback. As an example, the only values that EG codes can handle for Double Error Correction (DEC) are (n, k), where n is the size of the code word and k is the size of the data block.

Double error correction Orthogonal Latin Square codes are designed to have following characteristics for the parity check matrices H:

| | | |
|---|---|---|
| $n^2 * n^2$ | $1 * n^2$ | $n^2 * n^2$ Identity matrix |

In M1, m is the number of 1s in each row, and in M2, Im are identity matrices with dimensions m × m. The remaining submatrices, denoted as M3, M4,..., M2t, are constructed from pairwise orthogonal Latin squares of rank m.

A basic decoding method may be constructed using these characteristics. In this technique, the seven parity check equations are used to execute a majority vote for each data bit. If the output is 1, it means that the bit was incorrect and has to be fixed. Commonly, this process is called one-step majority logic decoding.

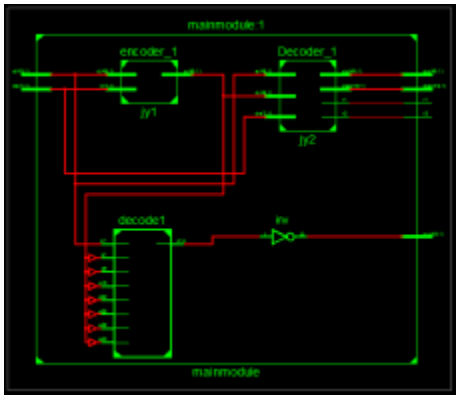
It fixes both single-bit and double-bit faults that impact data bits. On the other side, if the bit is accurate, mistakes with the other two bits can only impact two of its parity checks, meaning there won't be enough of a majority to initiate a repair.

The data bits are located in the left n2 * n2 columns of the parity check matrix H for the Double Error Correction Orthogonal Latin Square coding, as shown in Figure 1. Compared to testing every potential two-bit error pattern for a bit, the necessary logic circuit is simpler. Since the number of double-bit error patterns is directly proportional to n, this characteristic becomes more favourable as the code word size increases.

Figure 1: Parity check matrix *H* of the (n, 1, k) DEC OLS code.

Since the equations for calculating the code words can be directly acquired from each row, the design of the encoder for an OLS code is fairly straightforward. To create the parity bits, bits from each row are sent into XOR gates. Parity check matrices for Orthogonal Latin Square codes use the same number of rows as their number of bits. In the picture, we can see the decoder of an OLS code of size 1* n2 doing one-step majority

logic decoding.




RTL Schematic

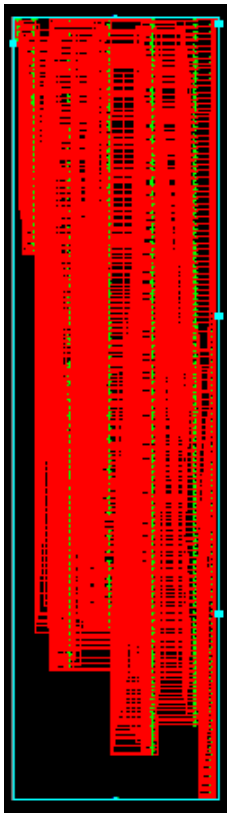
Because n is a prime number in this arrangement, we settled on 7.The decoding bits are 1*49, and the check matrix is a 49*49 identity matrix. The process of getting a code word's bits involves recalculating the bit and all of the parity check equations in which it is involved, followed by a majority vote among these equations. If a mistake has happened, the value of the syndrome bit or the parity check equation will be 1, otherwise it will be 0.

It is clear that an error has changed the bit's state and the bit has to be fixed if most of the syndrome bits become 1. With the right amount of inputs, modulo-2 adders or XOR gates may recalculate the parity check bits. One way to fix the problem at the end of the majority logic circuit is to use a 2-input XOR gate. This gate takes the bit as one input and uses the output of the majority logic circuit as the second input.

## RESULTS AND DISCUSSION

To correct multi-bit errors orthogonal Latin square codes arehighly efficient. Based on this work, the encoder and decoder of OLS codes can be implemented for larger data blocks. By comparing with the existing method, in proposedmethod the area as well as delay will be reduced.

http://doi.org/10.36893/JNAO.2022.V13I1.1973-1982



Technology Schematic

Simulation results:

Evaluation of Area, Delay and Power Results:

| | **Area** | **Delay** | **Power** |
|---|---|---|---|
| Existing | 563 | 9.162ns | 0.906 |
| proposed | 425 | 7.639ns | 0.906 |

## CONCLUSION

When compared to other multi-bit error correcting codes, orthogonal Latin square codes are much more efficient, take up far less space, and need far simpler circuitry for both the encoders and the decoders. This study lays the groundwork for using OLS codes, which have a highly flexible and modular encoder and decoder, to handle bigger data blocks and a range of data lengths. Confirmation of the encoder and decoder's low complexity and latency was provided by the implementation findings for an FPGA platform.

## REFERENCES

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

[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, Pari-chute: Generalized Turbo code-Based Error Correction for Near Thresh-old 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.

http://doi.org/10.36893/JNAO.2022.V13I1.1973-1982

[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 Nano science (FNANO07), Snowbird, UT, 2007.

[12] P. Reverie, 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.

http://doi.org/10.36893/JNAO.2022.V13I1.1973-1982