iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** III **Month of publication:** March 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Multiple Cell Errors Detection and Correction for OLS codes

P.Vijay Gopal[1], V.Sivaraghavaiah[2]
[1]Assistant Professor, Dept Of Electronics And Communication Engineering, Pragati Engineering College A.P, India
[2]M.Tech Scholar, Dept Of Electronics And Communication Engineering, Pragati Engineering College, A.P, India

Abstract-Error correction codes (ECCs) are commonly used to protect memories against errors. The proposed method we are we are detect the one or more errors and to correct single bit error. Among ECCs, orthogonal latin squares (OLS) codes have gained renewed interest for memory protection due to their modularity and the simplicity of the decoding algorithm that enables low delay implementations. An important issue is that when ECCs are used, the encoder and decoder circuits can also suffer errors. In this brief, a concurrent error detection technique for OLS codes encoders and syndrome computation is proposed and evaluated. The proposed method uses the properties of OLS codes to efficiently implement a parity prediction scheme that detects all errors that affect a single circuit node.

I. INTRODUCTION

Error correction codes (ECCs) have been used to protect memories for many years [1], [2]. There is a wide range of codes that are used or have been proposed for memory applications. Single error correction (SEC) codes that can correct one bit per word are commonly used. More advanced codes that can also correct double adjacent errors [3] or double errors in general have also been studied [4]. The use of more complex codes that can correct more errors is limited by their impact on delay and power, which can limit their applicability to memory designs [5]. To overcome those issues, the use of codes that are one step majority logic decodable (OS-MLD) has recently been proposed. OS-MLD codes can be decoded with low latency and are, therefore, used to protect memories [6]. Among the codes that are OS-MLD, a type of euclidean geometry (EG) code has been proposed to protect memories [7], [8]. The use of difference set code has also been recently analyzed in [9]. Another type of code that is OS-MLD is orthogonal latin squares (OLS) code [10]. The use of OLS codes has gained renewed interest for interconnections [11], memories [12], and caches [13]. This is due to their modularity such that the error correction capabilities can be easily adapted to the error rate [11] or to the mode of operation [13]. OLS codes typically require more parity bits than other codes to correct the same number of errors. However, their modularity and the simple and low delay decoding implementation (as OLS codes are OS-MLD), offset this disadvantage in many applications. An important issue is that the encoder and decoder circuits needed to use (ECCs) can also suffer errors. When an error affects the encoder, an incorrect word may be written into the memory. An error in the decoder can cause a correct word to be interpreted as erroneous or the other way around, an incorrect word to be interpreted as a correct word.

The protection of the encoders and decoders has been studied for different ECCs. For example, in [8] EG codes were studied. The protection of Reed–Solomon, Hamming, and BCH encoders and decoders has also been studied in [14] and [15], and more general techniques for systematic and cyclic codes have been proposed in [16] and [17]. Finally, the protection of encoders for SEC codes against soft errors was discussed in [18]. The ECC encoder computes the parity bits, and in most cases the decoder starts by checking the parity bits to detect errors. This is commonly referred to as syndrome computation. For some codes, it is possible to perform encoding and syndrome computation serially based on the properties of the code. However, when delay has to be low, parallel implementations are preferred. This is the case for OLS codes that are commonly used in high-speed applications. The reader is referred to [6] for a detailed discussion of ECC encoders and decoders. After syndrome computation, when errors are detected, the rest of the decoding is done to correct the errors. This means that generating and checking the parity bits are important parts of the encoder and decoder circuitry. Therefore, its protection is an important issue.

In this brief, the protection of the encoders and syndrome computation for OLS codes when used in SRAM memories and caches is considered. Based on the specific properties of these codes, it is shown that parity prediction is an effective technique to detect errors in the encoder and syndrome computation. This is not the case for most other block codes for which parity prediction cannot provide effective protection. Therefore, this is another advantage of OLS codes in addition to their modularity and simple decoding.

II. ORTHOGONAL LATIN SQUARES CODES

OLS codes are based on the concept of Latin squares. A Latin square of size m is an m × m matrix that has permutations of the digits 0, 1,…, m − 1 in both its rows and columns [19]. Two Latin squares are orthogonal if when they are superimposed every ordered pair of elements appears only once. OLS codes are derived from OLS [10]. These codes have k = m2 data bits and 2tm check bits, where t is the number of errors that the code can correct. For a double error correction code t = 2, and, therefore, 4m check bits, are used. As mentioned in the introduction, one advantage of OLS codes is that their construction is modular. This means that to obtain a code that can correct t +1 errors, simply 2m check bits are added to the code that can correct t errors. This can be useful to implement adaptive error correction schemes, as discussed in [11] and [13]. The modular property also enables the selection of the error correction capability for a given word size. As mentioned before, OLS codes can be decoded using OS-MLD as each data bit participates in exactly 2t check bits and each other bit participates in at most one of those check bits. This enables a simple correction when the number of bits in error is t or less. The 2t check bits are recomputed and a majority vote is taken. If a value of one is obtained, the bit is in error and must be corrected. Otherwise the bit is correct. As long as the number of errors is t or less, the remaining t −1 errors can, in the worst case, affect t −1 check bits. Therefore, still a majority of t + 1 triggers the correction of an erroneous bit. In any case, the decoding starts by recomputing the parity check bits and checking against the stored parity check bits.

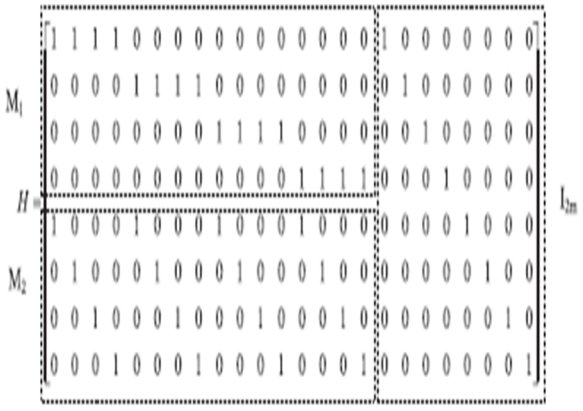


Fig 1 Parity check matrix for OLS code with k = 16 and t = 1.

The parity check matrix H for OLS codes is constructed from the OLS. As an example, the matrix for a code with k = 16 and 8 check bits that can correct single errors is shown in Fig. 1. As discussed earlier, due to the modular construction of OLS codes this matrix forms part of the H matrix for codes that can correct more errors. For example, to obtain a code that can correct two errors, eight additional rows are added to the H matrix. For an arbitrary value of k = m2, the H matrix for a SEC OLS code is constructed as follows:

$$H = \begin{bmatrix} M_1 & I_{2m} \\ M_2 & \end{bmatrix} \quad (1)$$

Where I2m is the identity matrix of size 2m and M1, M2 are matrices of size m × m2. The matrix M1 has m ones in each row. For the r th row, the ones are at positions (r − 1) × m + 1,(r − 1)× m + 2,…(r − 1) × m + m − 1, (r − 1) × m + m. The matrix M2 is constructed as follows:

$$M2 = [Im\ Im\ \ldots\ Im] \quad (2)$$

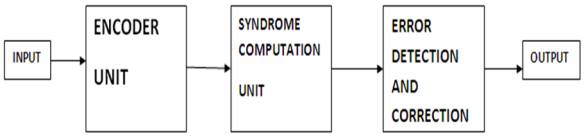
For m = 4, the matrices M1 and M2 can be clearly observed in Fig. 1. The encoding matrix G is just the H matrix on which the check bits are removed

$$G = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} \quad (3)$$

In summary, the encoder takes k = m2 data bits (di) and computes 2tm parity check bits (ci using a matrix G, which is derived from Latin squares and has the following properties. Each data bit participates exactly in 2t parity checks. A pair of data bits participates (both bits) in at most one of the parity checks. These properties are used in the next section to discuss the proposed technique.

Volume 3 Issue III, March 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

III. PROPOSED CONCURRENT ERROR DETECTION AND CORRECTION TECHNIQUE



Block diagram of proposed module

Before describing the proposed error detection techniques, the standard definition of self-checking circuits that are used in this section is presented. During normal, or fault-free, operation, a circuit receives only a subset of the input space, called the input code space, and produces a subset of the output space, called the output code space. The outputs that are not members of the output code space form the output error space. In general, a circuit may be designed to be self-checking only for an assumed fault set. In this brief, we consider the fault set **F** corresponding to the single stuck-at fault model [20]. A circuit is self-checking [20] if and only if it satisfies the following properties: 1) it is self-testing, and 2) fault-secure. A circuit is self-testing if, for each fault f in the fault set **F**, there is at least one input belonging to the input code space, for which the circuit provides an output belonging to the output error space.

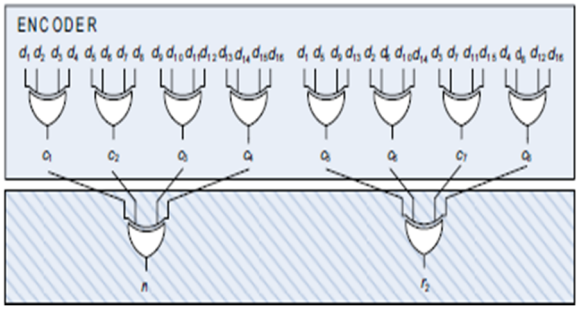


Fig 2 Proposed self-checking encoder for OLS code with k = 16 and t = 1.

A circuit is fault-secure if, for each fault f in the fault set **F** and for each input belonging to the input code space, the circuit provides the correct output, or an output belonging to the output error space. The fault-secure property guarantees that the circuit gives the correct response, or signals the presence of a fault that provides an output in the error space. Faults are always detected, since there is an input that produces an output that identifies the presence of the fault.

This property is related to the assumption that the interval between the occurrences of two faults is enough to permit to all the elements belonging to the input code space to appear as circuit inputs before the occurrence of the second fault. Thus, an output belonging to the output error space appears at the circuit output before the occurrence of the second fault. The technique that we propose is based on the use of parity prediction, which is one of the techniques commonly used to detect error in general logic circuits [21], [22]. In our case, the problem is substantially simpler, given the structure of the OLS codes. Forthe encoder, it is proposed that the parity of the computed check bits (ci) is compared against the parity of all the check equations. The parity of all the check equations is simply the equation obtained by computing the parity of the columns in G. For OLS codes, since each column in G has exactly 2t ones, the null equation is obtained (see, for example, Fig. 1). Therefore, the concurrent error detection (CED) scheme is simply to check

c1 ^ c2 ^ c3 ^ • • • ^c2tm = 0. (4)

This enables an efficient implementation that is not possible in other codes. For example, in a Hamming code a significant part of the columns in G has an odd weight and for some codes the number is even larger as they are designed to have odd weights [23]. The input code space of the OLS encoder corresponds to the input space, since the encoder can receive all the possible 2k input configurations. The output code space of the OLS encoder is composed by the outputs satisfying (4), while the output error space is the complement of the output code space. A fault that occurs in one of the gates composing the OLS encoder can change at most one of the ci check bits. When this change occurs, the OLS encoder provides an output that does not satisfy (4), i.e., an output belonging to the output error space. Hence, this guarantees the fault-secure property for this circuit. Additionally, since the encoder is

composed only by XOR gates, no logic masking is performed in the circuit. Therefore, when a fault is activated the error is propagated to the output. This ensures the self-testing property of the circuit. In order to check if the output of the OLS encoder belongs to the output code space or the output error space, a self-checking implementation of a parity checker is used [20]. The checker controls the parity of its inputs and is realized with a repetition code. The two outputs (r1, r2) are each equal to the parity of one of two disjoint subsets of the checker inputs (ci), as proposed in [24]. When a set of inputs with the correct parity is provided, the output code {r1, r2} takes the values 00 or 11. When the checker receives an erroneous set of inputs, the checker provides the output codes 01 or 10. Also, if a fault occurs in the checker, the outputs are 01 or 10. This guarantees the self-checking property of the parity checker [24]. The proposed encoder is illustrated in Fig. 2 for the code with k = 16 and t = 1. The proposed circuit can detect any error that affects an odd number of ci bits. For a general code, in most cases there is logic sharing among the computations of the ci bits [18]. This means that an error may propagate to more than one ci bit, and if the number of bits affected is even, then the error is not detected by the proposed scheme. To avoid this issue, the computation of each ci bit can be done separately. This, however, increases the circuit area of the encoder as no logic sharing is allowed. Another option is to control the logic in such a way that errors can only propagate to an odd number of outputs. This would also increase the cost compared to an unrestricted implementation. Additionally, even if the error propagates to an odd number of outputs, the delay of each path can be different. This may cause registering of only some of the output errors at the clock edge. For OLS codes, as discussed in the previous section a pair of data bits shares at most one parity check. This guarantees that there is no logic sharing among the computation of the ci bits. Therefore, the proposed technique detects all errors that affect a single circuit node. For the syndrome computation, the parity prediction can be implemented by checking that the following two equations take the same value

$r1 = s1 \oplus s2 \oplus s3 \oplus \cdots \oplus s2tm$ (5)

$r2 = c1 \oplus c2 \oplus c3 \oplus \cdots \oplus c2tm$ (6)

where si are the computed syndrome bits. The proposed circuit is shown in Fig. 3 for the code with k = 16 and t = 1. For syndrome computation, the input code space is only a subset of the possible 2k+2tm input configurations as only up to t errors are considered. This subset is given by the valid OLS code words and the non-valid OLS code words that are at a Hamming distance of t or less from a valid codeword. Those correspond to the input configurations in which there are no errors or at most t errors on the di inputs such that the errors can be corrected. The output code space of the OLS syndrome computation is composed by the outputs given by (5) and (6) satisfying r1 = r2, while the output error space is the complement of the output code space. The fault-secure property for the syndrome computation is easily demonstrated for the faults in **F** by observing that the circuits that compute r1 and r2 do not share any gate and both circuits are only composed of XOR gates. Therefore, a single fault could propagate to only one of the outputs, producing an output on the output error space. To prove the self-testing property for the syndrome computation, suppose that a fault occurs in one of the gates computing (5). If the input configuration is a valid OLS codeword, all the syndrome bits are 0, detecting all stuck-at-1 faults in the XOR gates computing (5). Instead, if the input is a non-OLS codeword that is affected by a t or less errors, some syndrome bits are 1, allowing the detection of a stuck-at-0 faults in the XOR gates computing (5). Finally, suppose that a fault occurs in one of the gates computing (6). Since any combination of the 2tm check bits is allowed, any fault can be activated and the error propagated to the output r2. For OLS codes, the cost of the encoder and syndrome computation in terms of the number of two-input XOR gates can be easily calculated (note that for the calculations an l−input XOR gate is assumed to be equivalent to l − 1 two-input XOR gates).

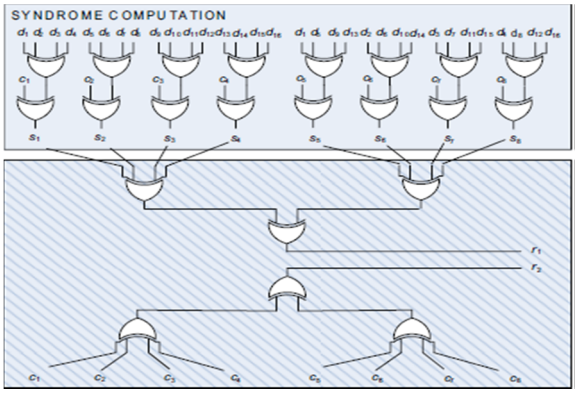


Fig 3 Proposed self-checking syndrome computation for OLS code with k = 16 and t = 1.

Volume 3 Issue III, March 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

For a code with k = m2 and that can correct t errors, there are 2tm parity check bits and the computation of each of them requires m − 1 two input XOR gates. Therefore, the encoder requires 2tm (m − 1) two-input XOR gates. For the syndrome computation, an additional XOR gate is needed for each parity check bit, giving a total of 2tm2 two-input XOR gates. The proposed method requires 2tm−1 two-input XOR gates for the encoder and 4tm-2 two-input XOR gates for the syndrome computation. This means that the overhead required to implement our method for the encoder is

$$O_{encoder} = \frac{(2m-1)}{(2tm(m-1))} \qquad (7)$$

and for the syndrome computation is

$$O_{syndrome} = \frac{(4tm-1)}{(2tm^2)} \qquad (8)$$

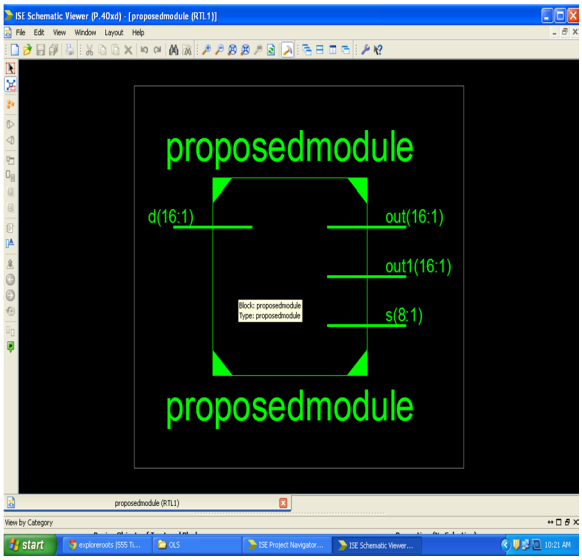
A. Detection Of Errors

In the propose method we are detect the errors by using syndrome computation bits. Here we are checking the syndrome bits
All syndrome bits from S1 to S8 are zeros in our output do not have any errors. Otherwise we are checking in syndrome computation bits having one or more number of one's in our output having errors.

B. Error Correction

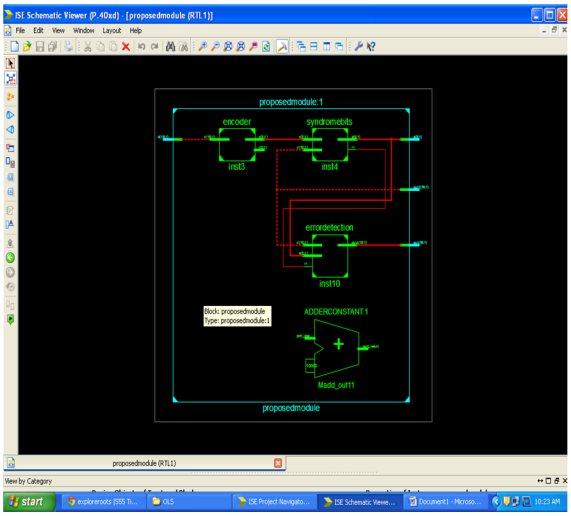
In the proposed technique we are correct the single bit error. For example S1, S5 syndrome computation bits are one's remaining all syndrome bits are zeros in this case the d1 value is error to correct the d1 value to invert the d1 value.
For example S1, S6 syndrome computation bits are one's remaining all syndrome bits are zeros in this case the d2 value is error to correct the d2 value to invert the d2 value.
For example S1, S7 syndrome computation bits are one's remaining all syndrome bits are zeros in this case the d3 value is error to correct the d3 value to invert the d3 value.
For example S1, S8 syndrome computation bits are one's remaining all syndrome bits are zeros in this case the d4 value is error to correct the d4 value to invert the d4 value.
For example S2, S5 syndrome computation bits are one's remaining all syndrome bits are zeros in this case the d5 value is error to correct the d5 value to invert the d5 value.
For example S2, S6 syndrome computation bits are one's remaining all syndrome bits are zeros in this case the d6 value is error to correct the d6 value to invert the d6 value.
The remaining bits are also same procedure to correct the error bit.

IV. SIMULATION RESULTS

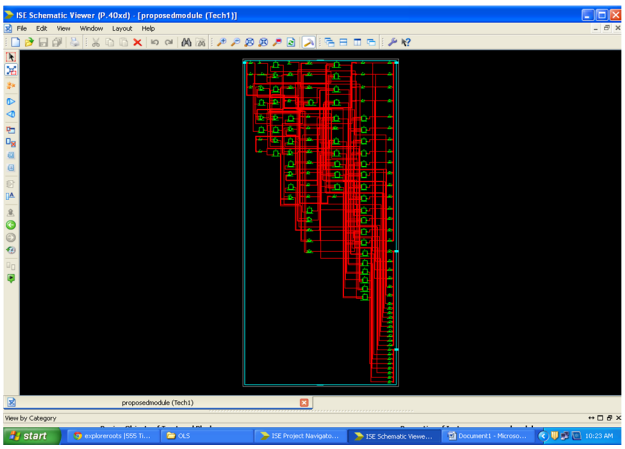


Volume 3 Issue III, March 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

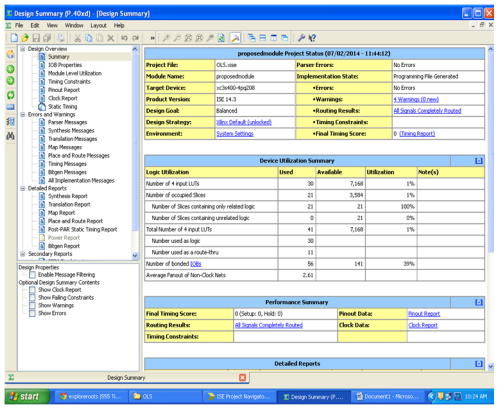
Block diagram



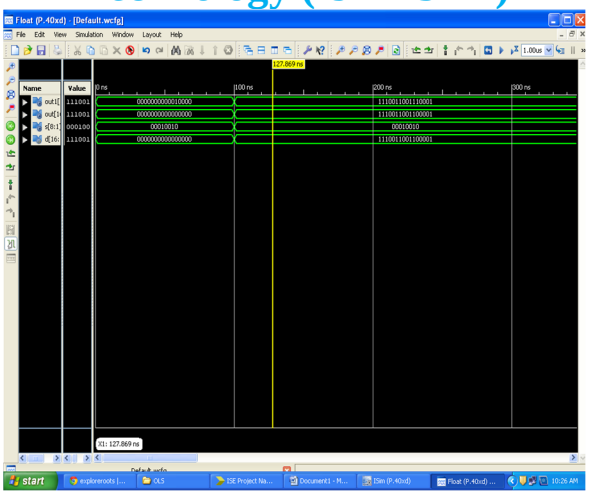
RTL schematic



Technology schematic



Design Summary



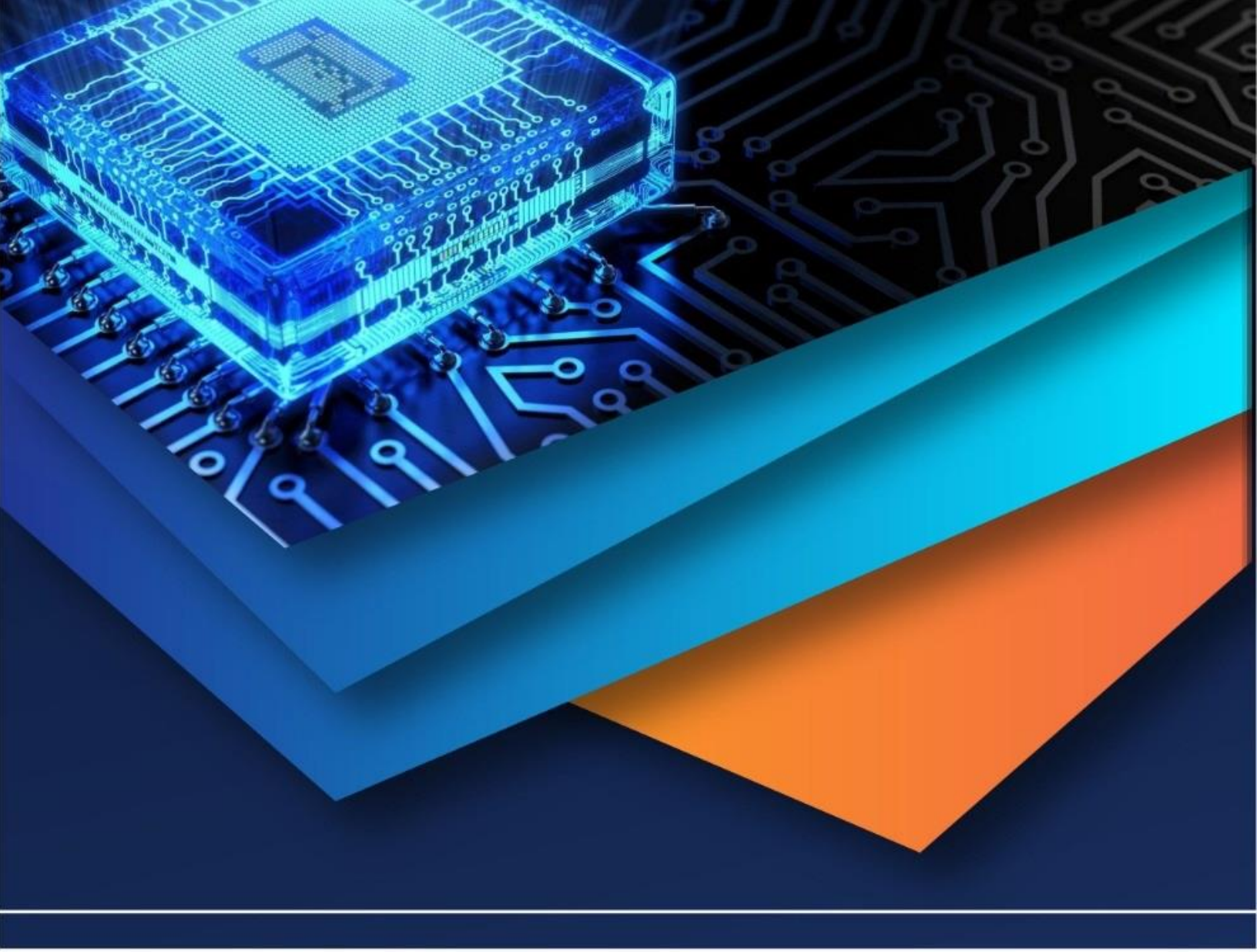
Simulation Result

V. CONCLUSION

In this brief, a CED technique for OLS codes encoders and syndrome computation was proposed. The proposed technique took advantage of the properties of OLS codes to design a parity prediction scheme that could be efficiently implemented and detects all errors that affect a single circuit node. Here the proposed scheme to detect the one or more errors and to correct the single bit errors by using Orthogonal Latin square error correcting method. The technique was evaluated for different word sizes, which showed that for large words the overhead is small. This is interesting as large word sizes are used, for example, in caches for which OLS codes have been recently proposed. The proposed error checking scheme required a significant delay; however, its impact on access time could be minimized.

REFERENCES

[1] 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.
[2] E. Fujiwara, Code Design for Dependable Systems: Theory and Practical Application. New York: Wiley, 2006.
[3] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in Proc. IEEE VLSI Test Symp., May 2007, pp. 349–354.
[4] R. Naseer and J. Draper, "DEC ECC design to improve memory reliability in sub-100nm technologies," in Proc. IEEE Int. Conf. Electron., Circuits, Syst., Sep. 2008, pp. 586–589.
[5] G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Fault tolerant solid state mass memory for space applications," IEEE Trans. Aerosp. Electron. Syst., vol. 41, no. 4, pp. 1353–1372, Oct. 2005.
[6] S. Lin and D. J. Costello, Error Control Coding, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004.
[7] S. Ghosh and P. D. Lincoln, "Dynamic low-density parity check codes for fault-tolerant nano-scale memory," in Proc. Found. Nanosci., 2007, pp. 1–5.
[8] 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.
[9] S. Liu, P. Reviriego, and J. A. Maestro, "Efficient majority logic fault detection with difference-set codes for memory applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 1, pp. 148–156, Jan. 2012.
[10] 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.
[11] S. E. Lee, Y. S. Yang, G. S. Choi, W. Wu, and R. Iyer, "Low-power, resilient interconnection with Orthogonal Latin Squares," IEEE Design Test Comput., vol. 28, no. 2, pp. 30–39, Mar.–Apr. 2011.
[12] R. Datta and N. A. Touba, "Generating burst-error correcting codes from orthogonal latin square codes–a graph theoretic approach," in Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst., Oct. 2011, pp. 367–373.
[13] A. R. Alameldeen, Z. Chishti, C. Wilkerson, W. Wu, and S.-L. Lu, "Adaptive cache design to enable reliable low-voltage operation," IEEE Trans. Comput., vol. 60, no. 1, pp. 50–63, Jan. 2011.
[14] G. C. Cardarilli, S. Pontarelli, M. Re, and A. Salsano, "Concurrent error detection in Reed-Solomon encoders and decoders," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 7, pp. 842–846, Jul. 2007.
[15] I. M. Boyarinov, "Self-checking circuits and decoding algorithms for binary hamming and BCH codes and Reed-Solomon codes over GF(2m)," Prob. Inf. Transmiss., vol. 44, no. 2, pp. 99–111, 2008.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY