Error Detecting and Correcting Code Using Orthogonal Latin Square Using Verilog HDL

Ch.Srujana
M.Tech [EDT]
srujanaxc@gmail.com
SR Engineering College, Warangal.

M.Sampath Reddy
Assoc. Professor, Department Of ECE
sampath_reddy_m@srecwarangal.ac.in
SR Engineering College, Warangal.

Abstract: Error correction codes (ECCs) are commonly used to protect memories against errors. 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 is used, the encoder and decoder circuits can also suffer errors. In this brief, a concurrent error detection technique for OLS codes 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. 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. 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.

Keywords—Concurrent error detection(CED), error correction codes (ECC), Latin squares, majority logic decoding (MLD), parity, memory.

I. Introduction

Reliability is a major concern in advanced electronic circuits. Errors caused for example by radiation become more common as technology scales. To ensure that those errors do not affect the circuit functionality a number of mitigation techniques can be used[1]. Among them, Error Correction Codes (ECC) are commonly used to protect memories and registers in electronic circuits[2]. The general idea for achieving error detection and correction is to add some redundancy which means to add some extra data to a message, which receiver can use to check uniformity of the delivered message, and to pick up data determined to be corrupt. Error-detection and correction scheme may be systematic or it may be non-systematic. In the system of the module non-systematic code, an encoded is achieved by transformation of the message which has least possibility of number of bits present in the message which is being converted. Another classification is the type of systematic module unique data is sent by the transmitter which is attached by a fixed number of parity data like check bits that obtained from the data bits. The receiver applies the same algorithm when only detection of the error is required to the received data bits which is then compared with its output with the receive check bits if the values does not match, there we conclude that an error has crept at some point in the process of transmission. Error correcting codes are regularly used in lower-layer communication, as well as for reliable storage in media such as CDs, DVDs, hard disks and RAM.

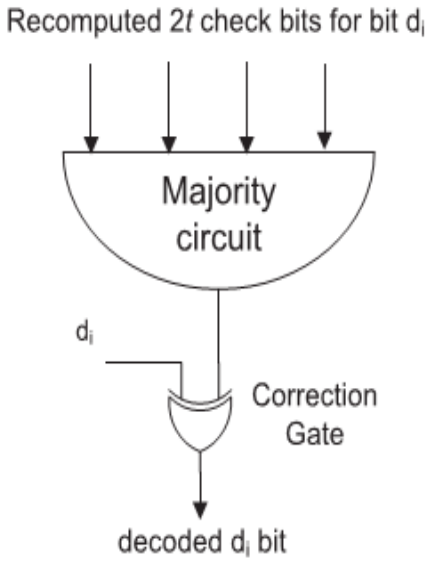

Fig.1. Illustration of OS-MLD decoding for OLS codes

Provision against soft errors that apparent they as the bit-flips in memory is the main motto of error detection and correction. Several techniques are used present to midi gate upsets in memories. For example, the Bose – Chaudhuri– Hocquenghem codes, Reed–Solomon codes, punctured difference set codes, and matrix codes has been used to contact with MCUs in memories. But the above codes mentioned requires more area, power, and delay overheads since the encoding and decoding circuits are more complex in

these complicated codes. Reed-Muller code is another protection code that is able to detect and correct additional error besides a Hamming code. But the major drawback of this protection code is the more area it requires and the power penalties. Hamming Codes are mostly used to correct Single Error Upsets (SEU's) in memory due to their ability to correct single errors through reduced area and performance overhead. Although it is brilliant for correction of single errors in a data word, but they cannot correct double bit errors caused by single event upset. An extension of the basic SEC-DED Hamming Code has been proposed to form a special class of codes known as Hsiao Codes to increase the speed, cost and reliability of the decoding logic. One more class of SEC-DED codes known as Single-error-correcting, Double error-detecting Single-byte-error-detecting SEC-DED-SBD codes be proposed to detect any number of errors disturbing a single byte. These codes are additional suitable than the conventional SEC-DED codes for protecting the byte-organized memories [4]. Though they operate through lesser overhead and are good for multiple error detection, they cannot correct multiple errors. There are additional codes such as the single -byte-error-correcting, double-byte-error detecting (SBC-DBD) codes, double-error-correcting, triple error-detecting (DEC-TED) codes that can correct multiple errors. The Single-error-correcting, Double-error-detecting and Double-adjacent-error-correcting (SEC-DED-DAEC) code provides a low cost ECC methodology to correct adjacent errors as proposed The only drawback through this code is the possibility of miss-correction for a small subset of many errors.

II. Literature Survey

Most prior work in memory ECC has focused on low failure rates present at normal operating voltages, and has not focused on the problem of persistent failures in caches operating at ultra low voltage where defect rates are very high.

Error correction codes (ECCs) have been used to protect memories for many years. There are wide ranges of codes that used or proposed for the applications in the memory. The codes that can correct one bit per word called the Single error correction are commonly used known as SEC. More sophisticated studies are carried on the codes that correct the double adjacent errors or the double errors. Further the use of more complex codes that corrects more errors is limited by their impact on delay and power, which limits their applicability to the design of memory. To surmount the issues, the use of codes that are one step majority logic decodable (OS -MLD) has been proposed recently. OS-MLD codes can be decoded with low latency and so they are used for the protection of memories. Another type of code that is OS-MLD is orthogonal Latin squares called the OLS codes. While OLS codes require more redundancy than conventional ECC, the one-step majority encoding and decoding process is very fast and can be scaled up for handling large numbers of errors as opposed to BCH codes, which while providing the desired level of reliability requires multi-cycles for decoding. The post-manufacturing customization approach proposed in this paper can be used to reduce the number of check bits and hence the amount of redundancy required in the memory while still providing the desired level of reliability. Note that the proposed approach does not reduce the hardware requirements for the OLS ECC as the whole code needs to be implemented on-chip since the location of the defects is not known until post-manufacturing test is performed.

III. Orthogonal Latin Squares codes

The concept of Latin squares and their applications are well known [5]. 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[6]. For each value of m there can be more than one Latin square. When that is the case, two Latin squares are said to be orthogonal if when they are superimposed every ordered pair of elements appears only once. Orthogonal Latin Squares (OLS) codes are derived from Orthogonal Latin squares [7]. These codes have $k=m^2$ data bits and 2tm check bits where t is the number of errors that the codes can correct.OLS codes can be decoded using OS-MLD. OS-MLD is a simple procedure in which each bit is decoded by simply taking the majority value of the set of the recomputed parity check equations, in which it participates. This is shown in Fig. 1 for a given data bit di . The reasoning behind OS-MLD is that when an error occurs in bit di, the recomputed parity checks in which it participates will take a value of one [8].

Therefore, a majority of ones in those recomputed checks is an indication that the bit is in error and therefore needs to be corrected. However, it may also occur that errors in other bits different from di provoke a majority of ones that would cause mis-correction. For a few codes, their properties ensure that this mis-correction cannot occur, and therefore OS-MLD can be used. For a Double Error Correction (DEC) code t=2 and therefore 4m check bits are used. 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. The modular property enables the selection of the error correction capability for a given word size. As mentioned in the introduction, OLS codes can be decoded using One Step Majority Logic Decoding (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 this ensures the error correction as the remaining t-1 errors can, in the worst case affect t-1 check bits so that still a majority of t+1 triggers the correction of an erroneous bit. For an OLS code that can correct t errors using OS-MLD, t+1 errors can cause miss-corrections. This occurs for example if the errors affect t+1 parity bits in which bit di participates as this bit will be miss-corrected. The same occurs when the number of errors is larger than t+1. Each of the 2t check bits in which a data bit participates is taken from a group of m parity bits. Those groups are bits 1 to m, m+1 to 2m, 2m+1 to 3m and 3m+1 to 4m.

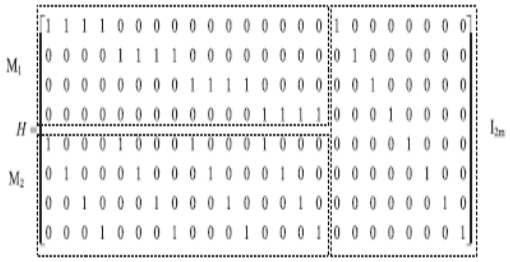

Fig. 2. Parity check matrix for OLS code having k and t as 16&1

The "H" matrix for OLS codes is build from their properties. The matrix is capable of correcting single type error. By the fact that in direction of the modular structure it might be able to correct many errors. They have check bits of number "2tm" in which, "t" stands for numeral of errors such that code corrects. If we wanted to correct a double bit then we have "2" as the value of t and thereby the check bits required are 4m.The H matrix , of Single Error Code "OLS" code is construct as :

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

a. In the above, I^2m is the identity matrix of size 2 m.
b.M_1, M_2 is the matrices of given size m ×m_2.
"The matrix M_1 have m ones in respective rows. For the rth row, the 1's are at the position
$(r-1)\times m+1,(r-1)\times m+2,\ldots\ldots\ldots\ldots(r-1)\times m+m-1,(r-1)\times m+m$".
The matrix M_2 is structured as

$$M_2=[I_m, I_m, \ldots I_m]$$

For the given value 4 for m, the matrices M_1 and M_2 can be evidently experiential in Fig.2 H Matrix in the check bits we remove is evidently the G Matrix

$$G=\begin{bmatrix}M_1\\ M_2\end{bmatrix} \qquad (2)$$

On concluding the above mentioned, it is evident that the encoder is intriguing m^2 data bits and computing 2tm parity check bits by using G matrix .

IV. Proposed Method

The proposed method is based on the observation that by construction, the groups formed by the m parity bits in each Mi matrix have at most a one in every column of H. For the example in Fig. 2, those groups correspond to bits (or rows) 1–4 (M_1), 5–8 (M_2), 9–12 (M_3), and 13–16 (M_4). Therefore, any combination of four bits from one of those groups will at most share a one with the existing columns in H. For example, the combination formed by bits 1, 2, 3, and 4 shares only bit 1 with columns 1, 2, 3,and 4. This is the condition needed to enable OS-MLD. Therefore, combinations of four bits taken all from one of those groups can be used to add data bit columns to the H matrix. For the code with k=16 and t =2 shown in Fig. 2, we have m=4. Hence, one combination can be formed in each group by setting all the positions in the group to one. This is shown in Fig. 3, where the columns added are highlighted. In this case, the data bit block is extended from k=16 to k=20 bits. The proposed method first divides the parity check bits in groups of m bits given by the M_i matrices. Then, the second step is for each group to find the combinations of 2t bits such that any pair of them share at most one bit. This second step can be seen as that of constructing an OS-MLD code with m parity check bits. Obviously, to keep the OS-MLD property for the extended code, the combinations formed for each group have to share at most one bit with the combinations formed in the other $2t-1$ groups. This is not an issue as by construction, different groups do not share any bit. When m is small finding, such combinations is easy. For example, in the case considered in Fig. 3, there is only one possible combination per group.

When m is larger, several combinations can be formed in each group. This occurs, for example, when m=8. In this case, the OLS code has a data block size k =64. With eight positions in each group, now two combinations of four of them that share at most one position can be formed. This means that the extended code will have eight (4×2) additional data bits. As the size of the OLS code becomes larger, the number of combinations in a group also grows. For the case m=16 and k =256, each group has 16 elements. Interestingly enough, there are 20 combinations of four elements that share at most one element. In fact, those combinations are obtained using the extended OLS code shown in Fig. 3 in each of the groups. Therefore, in this case, 4×20=80 data bits can be added in the extended code. The method can be applied to the general case of an OLS code with k =m^2 that can correct t errors. Such a code has 2tm parity bits that as before are divided in groups of m bits. The code can be extended by selecting combinations of 2t parity bits taken from each of the groups. These combinations can be added to the code as long as any pair of the new combinations share at most one bit. When m is small, a set of such combination with maximum size can be easily found.

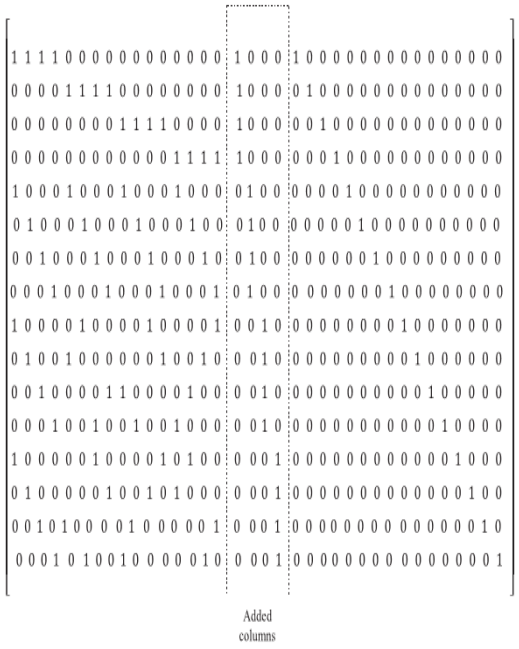

Fig. 3. Parity check matrix H for the extended OLS code with k=20 and t =2

However, as m grows, finding such a set is far from trivial (as mentioned before, solving that problem is equivalent to designing an OS-MLD code with m parity bits that can correct t errors). An upper bound on the number of possible combinations can be derived by observing that any pair of bits can appear only in one combination. Because each combination has 2t bits, there are (2t 2) pairs in each combination. The number of possible pairs in each group of m bits is m^2. Therefore, the number of combinations NG in a group of m bits has to be such that

$$\binom{m}{2} \geq \binom{2t}{2} \times NG \qquad (3)$$

which can be simplified as

$$\frac{m^2-m}{2t-1} \geq N_G \qquad (4)$$

One particular case for which a simple solution can be found is when m=2t ×l. In this case, an OLS code with a smaller data block size (l^2) can be used in each group. One example for t =2 is when m=16 (k=256) for which the OLS code with block size $k=4^2$ can be used in each group as explained before. Similarly, for t =2, when k=1024 (m=32) the OLS code of size k $=8^2$ can be used in each group.

V. RESULTS

The Xilinx ISE software used to perform compilation and synthesis. A test bench is created to execute the simulation. The encoder is stimulated using the above software. The simulation results for the computation of Parity check matrix for OLS code with k=16, t=1 and the parity check matrix H for extended OLS code with k=20 and t=2 is shown in Appendix, where d is input and r_1, r_2 are outputs.

VI. CONCLUSION

In this brief, a CED technique for OLS codes encoders and syndrome computation is 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. 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. This was achieved by performing the checking in parallel with the writing of the data in the case of the encoder and in parallel with the majority voting and error correction in the case of the decoder. In a general case, the proposed scheme required a much larger overhead as most ECCs did not have the properties of OLS codes. This limited the applicability of the proposed CED scheme to OLS codes. The availability of low overhead error detection techniques for the encoder and syndrome computation is an additional reason to consider the use of OLS codes in high-speed memories and caches.

REFERENCES

[1] M. Nicolaidis, “Design for Soft Error Mitigation”, IEEE Trans. on Device And Materials Reliability, Vol. 5, No. 3, pp 405-418, Sept. 2005.
[2] C. L. Chen and M. Y. Hsiao, “Error-correcting codes for semiconductor memory applications: a state-of-the-art review”, IBM Journal of Research and Development, vol. 28, no. 2, pp. 124-134, 1984.
[3] M.Y. Hsiao, “A Class of Optimal Minimum Odd-weight-column SEC-DED Codes”, IBM Journal of Research and Development, vol. 14 pp. 395-401, July 1970,.
[4]R. Naseer and J. Draper, “DEC ECC design to improve memory reliability in sub-100 nm technologies,” in Proc. IEEE ICECS, Sep. 2008,pp. 586–589
[5] J. Dénes and A. D. Keedwell, “Latin squares and their applications” Academic Press, 1974.
[6] J. Dénes and A. D. Keedwell,Latin Squares and Their Applications San Francisco, CA, USA: Academic, 1974
[7] M.Y. Hsiao, D.C. Bossen, and R.T. Chien, ‘‘Orthogonal Latin Square Codes,’’ IBM Journal of Research and Development, vol. 14, no. 4, pp. 390-394, 1970.
[8] S. Lin and D. J. Costello,Error Control Coding, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

APPENDIX

Simulation Data:

In Fig.4.'d' is input and r_1 and r_2 are outputs. The value of d[0:15] is 1011000000100000 with r1 and r2 as outputs. The results infer that when r_1=r_2 no error has occurred. Hence input is same as output. In Fig.5.'d' is input and r_1 and r_2 are outputs. The value of d[0:19] is 10100111001011010110 with r_1 and r_2 as outputs. The results infer that when r_1=r_2 no error has occurred. Hence input is same as output. The proposed technique of extending from k=16 to k=20 bits 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. The technique was evaluated for different bit sizes, which showed that for large bits the overhead is small. In a general case, the proposed scheme required a much larger overhead as most ECCs did not have the properties of OLS codes. This limited the applicability of the proposed CED scheme to OLS codes.

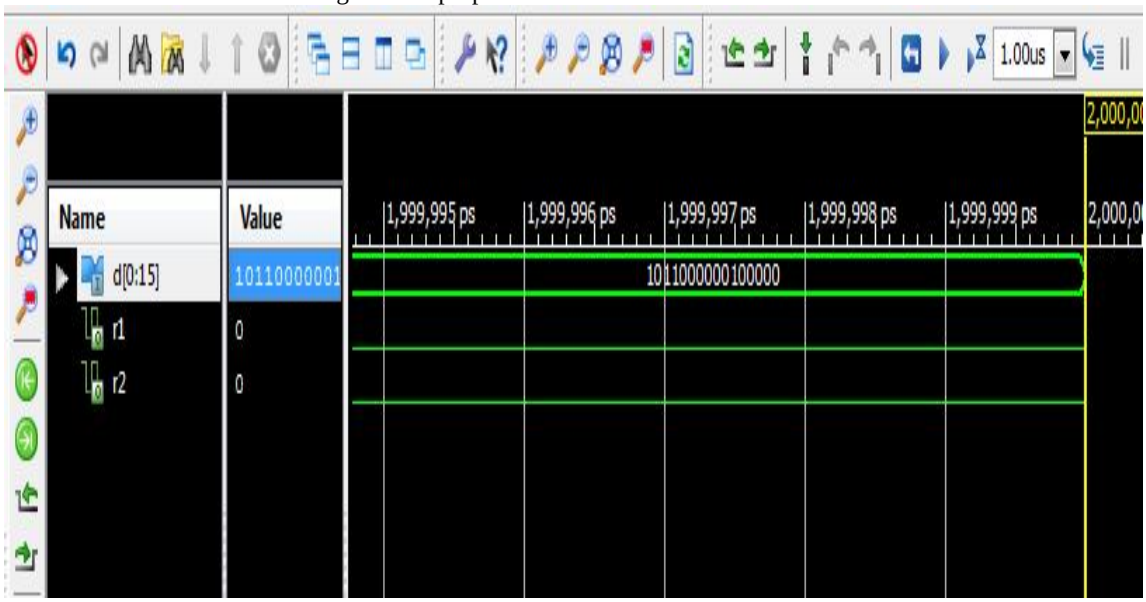

Fig. 4. For 16-bit OLS

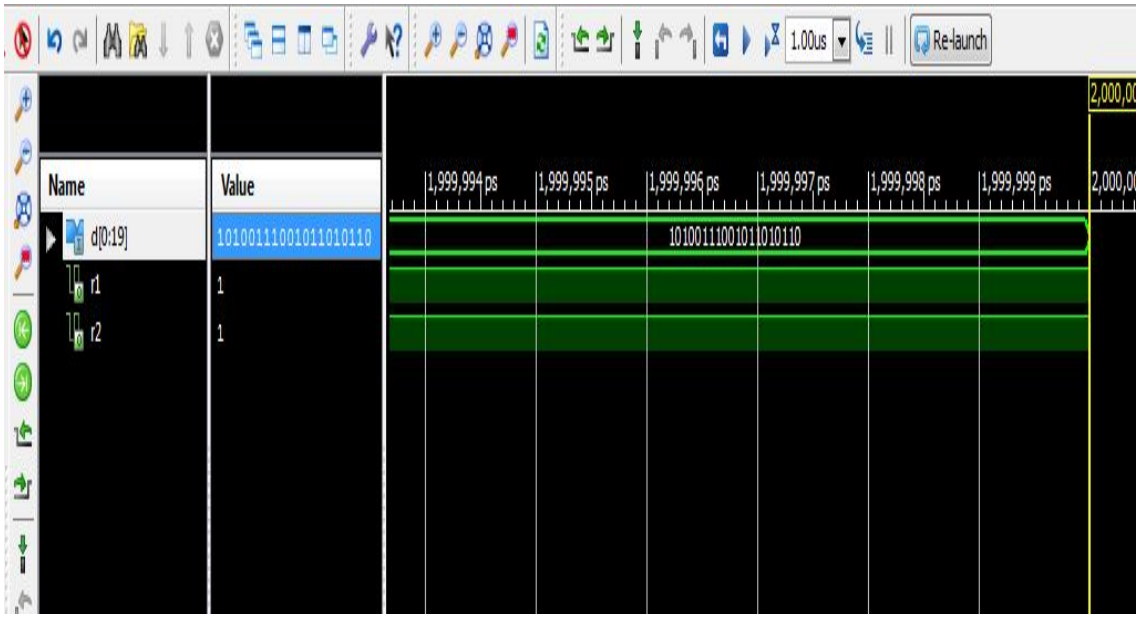

Fig. 5. For 20-bit OLS