INTERNATIONAL JOURNAL FOR TRENDS IN ENGINEERING & TECHNOLOGY
VOLUME 6 ISSUE 1 – JUNE 2015 - ISSN: 2349 - 9303

Novel SEC-DAEC-DNAEC Correction Codes Derived From Orthogonal Latin Square Codes

Stephie Hennah. J
Department of VLSI Design
Karunya University
Karunya Nagar, Coimbatore-641114, India
stephiehennah@gmail.com

Mr. Augustine Fletcher M.E.,
Department of VLSI Design
Karunya University
Karunya Nagar, Coimbatiore-641114, India
fletcher@karunya.edu

Abstract— Memories are more prone to errors. Redundancy is one technique to get rid of the errors in memories ie, to add some extra data to a message, which the receiver can use to check correctness of the data at the receiver side, and to get the data that is said to be corrupt. In order to ensure that memory contents are error free, single error correction double error detection (SEC-DED) codes are used, however, with recent advancement in technology, soft errors often affect more than one memory bit. But SEC-DED codes will not be able to correct multiple errors, and so interleaving is one common method that has been adopted. Interleaving has always affected memory design and cannot be used in memories at all circumstances. SEC-DED-DAEC codes when used has higher rates of decoder complexity and delay. Another important issue is that most of the time the methods that are carried out in new SEC-DED-DAEC codes may miscorrect some double nonadjacent bit errors. Apart from these methods OLS codes can be used, that can correct memories and caches. Their error correction capabilities can be adapted to the error rate or to the mode of operation. In this brief, a new class of SEC-DED DAEC DNAEC codes is derived from orthogonal Latin squares codes is presented. The new codes significantly reduce the decoding complexity and delay. Furthermore it has the advantage of reduced check bits and also has improved reliability of correcting even the errors that are not adjacently occurring.

Index Terms— Single Error Correction (SEC), Double Adjacent Error Correction (DAEC), Memory, Orthogonal Latin squares (OLS)

I. INTRODUCTION

For all these years errors in memories were not of concern since they were minimally affecting the correctness of the system. But later as the technology increased the complexity eventually increased and the correctness of the data became the increasingly significant factor for a memory.

As the trend in semiconductor memory design continues toward higher chip density and larger storage capacity, ECCs are becoming a more cost-effective means of maintaining a high level of system reliability. A memory system can be made fault tolerant with the application of an error correcting code; i.e., the mean time between failures of a properly designed memory system can be significantly increased with ECC. In this context, a system fails only when the errors exceed the error-correcting capability of the code. Also, in order to optimize data integrity, the ECC should have the capability of detecting the most likely of the errors that are not correctable.

Radiation from high-energy neutrons and an alpha particle can cause a single-event upset (SEU) that may alter the state of the system resulting in a soft error. There are two forms of errors formed. One is the Soft error and other is the Hard error. Hard errors are caused by struck-at faults or by permanent physical damage to the device. Soft errors are temporary errors such as an alpha particle induced error that can go away when the next write operation occurs.

An error correcting code is used to correct both soft errors and hard errors. A data can have a single error and multiple errors. Several methods have been used for deriving an error correcting code through heuristic search that can detect and correct the most likely double bit errors in a memory while minimizing the miscorrection probability of the non-adjacent double bits error.

The remainder of the paper is structured as follows. Section 2 related works in the error correction codes. Section 3 presents the brief introduction about the OLS codes. Section 4 is all about the OS-MLD. Section 5 is the construction of error correction code and double adjacent and non-adjacent error correction codes. Section 6 is the simulation analysis and section 7 concludes the work.

II. RELATED WORK

Error-correcting codes used in early computer memory systems were of the class of single-error-correcting with double error-detecting (SEC-DED) codes. A SEC-DED code is

capable of correcting one error and detecting two errors in the received data. The double-error detecting capability serves to guard against data loss. Conventional error correcting code (ECC) schemes used in memories and caches cannot correct double bit errors caused by a single event upset (SEU). A SEC-DED code is capable of correcting one error and detecting all possible double errors.

M. Y. Hsiao in [2] the class of codes described in this paper is used for single-error correction and double error detection (SEC-OED). It is equivalent to the Hamming SEC-OED code in the sense that for a specified number k of data bits, the same number of check bits r is used. The minimum odd-weight-column code is suitable for applications to computer memories or parallel systems. A computation indicates that this code is better in performance, cost and reliability than are conventional Hamming SEC-OED codes. This class of codes has better error detecting capability for triple and quadruple errors than do the conventional modified Hamming codes.

A. Dutta et' Al in [3] they present a state-of-the-art review of error-correcting codes for computer semiconductor memory applications. The construction of four classes of error-correcting codes appropriate for semiconductor memory designs is described, and for each class of codes the number of check bits required for commonly used data lengths is provided which is useful for designers for system planning. The implementation aspects of error correction and error detection are also discussed, and certain algorithms useful in extending the error-correcting capability for the correction of soft errors such as a-particle-induced errors are examined.

R. Naseer Et'Al in [4] The ECC methodology described in this paper adds the ability to correct adjacent errors at very little cost over conventional SEC-DED codes. The only drawback is the possibility of miscorrection for a small subset of multiple errors, however MBUs caused by a single SEU have a much higher probability of occurring than having multiple independent SEUs accumulating in the same word.Uses error correcting code through heuristic search that can detect and correct the most likely double bit errors in memory

G. C. Cardarilli Et'Al in[5] they have presented a special variant of low-density parity codes (LDPCs) such as Euclidean Geometry LDPC (EG-LDPC) codes to enable dynamic changes in level of fault tolerance. Apart from high error correcting ability and sparsity, a special property of EG-LDPC codes enables to dynamically adjust the error correcting capacity for improved system performance during periods of expected low fault arrival rate. Also a system architecture for Nano memory based on nano PLA building blocks using EG-LDPCs is presented, and an analysis of its fault detection and correction capabilities are explained.

S. Lin Et'Al in [6] They presents DEC code design that is aligned to typical memory word widths and a parallel decoding implementation approach that operates on complete memory words in a single cycle. The practicality of this approach is demonstrated through ASIC implementations, in which it incurs only 1.4ns and 2.2ns decoding latencies for 16 bits and 64 bit words, respectively, using 90nm ASIC technology. It deals with SEC BCH code design that is aligned to typical memory word sizes. This approach enables a simple Single-cycle implementation of DEC decoders rather than iterative multi-cycle decoding used in communication systems. ECC encoder and decoder circuits have been synthesized using standard cell IBM 90nm technology for typical memory word sizes.

H. Naeimi Et'Al in [8] they introduces a fault-tolerant nano scale memory architecture which tolerates transient faults both in the storage unit and in the supporting logic. Euclidean Geometry codes are used. This unified approach reduces the area overhead. This class satisfies a new, restricted definition for ECCs which guarantees that the ECC code word has an appropriate redundancy structure such that it can detect multiple errors occurring in both the stored codeword in memory and the surrounding circuitries. This type of error-correcting codes are called fault-secure detector capable ECCs (FSD-ECC). The parity-check Matrix of an FSD-ECC has a particular structure that the decoder circuit, generated from the parity-check Matrix, is Fault-Secure

III. OLS CODE

The OLS codes are framed up on the perception of Latin squares. OLS codes have been introduced some decades ago and now it's being used to correct caches and interconnects. It is a matrix of the order m and have up to the permutations m-1. Two Latin squares are orthogonal if they are superimposing every ordered pair of elements appear only once. OLS codes that we have presented have data bits of number m2. They have check bits of number 2tm in which, t stands for numeral of errors such that code corrects. For a double error correction code, t = 2, and, 4m check bits, are generated. By this property we can select the capability to correct the corrupted data for a desired word sizes. We can decode it by using MLD because 2t check bits are involved. OLS codes can be decoded using OS-MLD because each data bit participates in accurately 2t check bits and each other bit participates in at the majority in one of those check bits. The 2t check bits are recomputed and a majority vote is taken. If a value of one is obtained, the bit be in error and must be corrected. Otherwise the bit is correct. As long as the digit of errors is t or less, the remaining t −1 errors can, in the worst case, affect t −1 check bits.

The parity check matrix H for OLS codes is constructed from the OLS. The matrix intended for a code with k = 16 for which 8 check bits are generated, that be able to correct single errors is shown in Fig. 1.

$$H = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} \begin{bmatrix} 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1 \\ 1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0 \\ 0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0 \\ 0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0 \\ 0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1 \end{bmatrix} \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix} I_{2m}$$

To obtain a code to 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:

$$H = \begin{bmatrix} M_1 & I_{2m} \\ M_2 & \end{bmatrix}$$

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 . . . Im]

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

$$G = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$$

IV. OS-MLD

OLS codes can be decoded using OS-MLD. The use of OS-MLD enables a simple and fast decoding that is attractive to protect memories when decoding latency is critical In OS-MLD, each bit is decoded by simply taking the majority value on the set of the recomputed parity check equations in which it participates. The idea behind OS-MLD is that when an error occurs in bit d, the recomputed parity checks in which it participates will take a value of one unless there are errors in other bits. Therefore, a majority of ones in those recomputed checks is an indication that the bit is in error and therefore needs to be corrected. If the code is such that two bits share at most one parity check, then t−1 errors on other bits will not affect the majority of the 2t vote and therefore, the error will be corrected. Only a few codes have this property and can be decoded using OS-MLD. This is the case for difference set codes and for OLS codes.

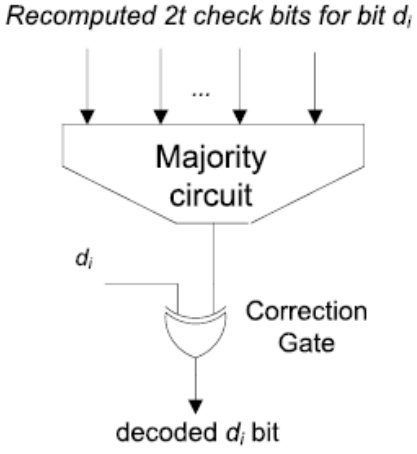

Fig 2: OS-MLD decoding for OLS codes

As shown in Fig. 2, the use of OS-MLD enables a simple and fast decoding that is attractive to protect memories when decoding latency is critical. Another characteristic of OLS codes is that they correct only errors on the data bits. No correction is done for the parity bits. The presented codes are derived from DEC OLS codes. These are block linear codes that are defined by their parity generating G and parity check H matrixes. The parity check matrix is used to detect errors by computing the syndrome s that is obtained by multiplying the stored word by the H matrix.

V. PROPOSED TECHNIQUE

The proposed scheme is derived from OLS codes. The first step is to eliminate the first row M1. The data bits that were in the row that was eliminated will not share any parity bits in the reduced matrix. Therefore elimination of such a row has no impact on the reduced matrix in anyway. The row that has been eliminated is given the label g1, g2, g3, g4. The 16 bit data is divided as 4 four packs of data.so that it makes the correction process easier for the programmer. This separation is done both in row vise and column vise.

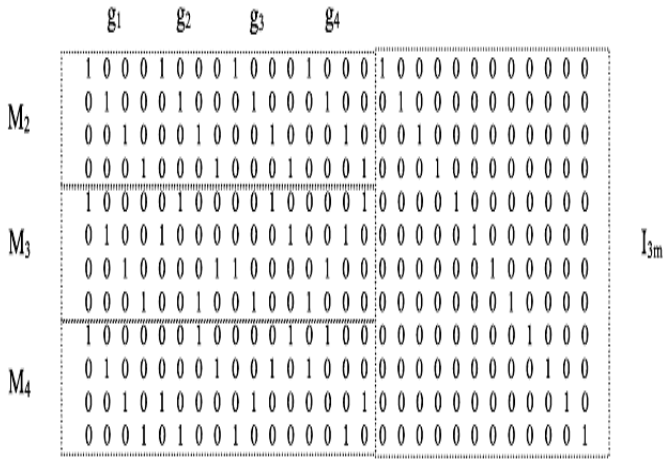

Fig 3: Reduced parity check matrix H

The first step is to generate the parity check bits for the given data. The first check bit is generated using s1, s5, s9, s13 and similarly the remaining bits are generated in accordance with the OLS pattern. Each check bit participate in three other bits in the matrix.

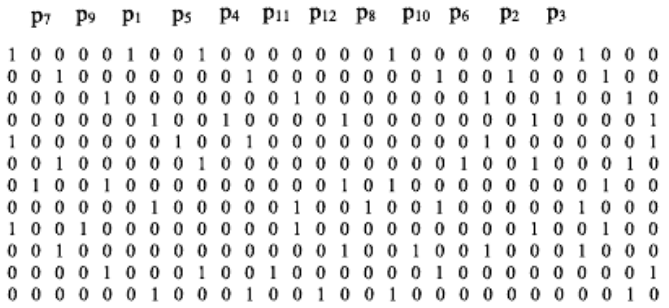

Fig 4: Replacement pattern for the H matrix

After the generation of the parity bit the data has to be shuffled in order to encode the data. The above mentioned pattern is used to shuffle the data so to send to the decoder. The first bit of the data is kept as it is in the first position and the second is replaced by the seven check bit. Similarly the 12 check bits are placed in between the 16 data bit. The resulting H matrix is the fully encoded data. The decoder side has the correction scheme. The data to be decoded is first separated as data and check bits and then another set of check bits are generated for the incoming data. The incoming check bits and generated check bits are compared to check for the occurrence of error.

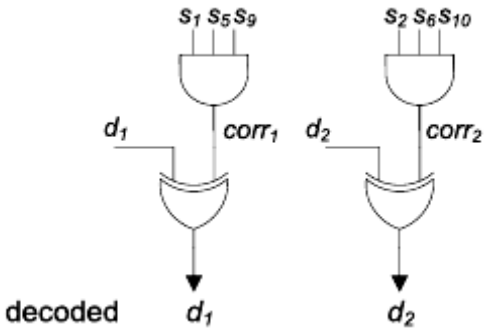

Fig 5: Single error detection block

In order to check for single and double adjacent errors the block shown in fig 4 is used. If the error is at d1 is to be checked then the check bits s1, s5, s9 are to be put together in a AND gate. It is then put together with the data bit d1 in a XOR gate. Similarly the check bits that participate in the corresponding data bits are to be considered for the errors to be detected in the data. The scheme is useful in detecting and correcting single and double adjacent errors. In order to correct the errors that occur non adjacently the OLS codes are to be slightly modified.

The technique that we propose is based on the use of parity prediction. Since the previous work had a drawback of having more check bits generated. This work has reduced the check bits generation by 4 and has only 8 check bits being generated for the same 16 bit data. For the encoder, it is proposed that the parity of the computed check bits 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.

D1 C5 D2 C8 D3 C4 D4 C7 D5 C6 D6 C3 D7 C1 D8 C2 D9

Fig 6: replacement pattern for the proposed scheme

The OLS code that has been used is shown in fig1. The pattern has the eliminated combination of the previous code. The modification is that the columns three and four are eliminated. And also due to the elimination the number of check bits are also reduced from 12 to 8. The replacement and encoding is similar to that of the previous work. The data bits are placed in place after all 8 check bits are placed in between the data bits.

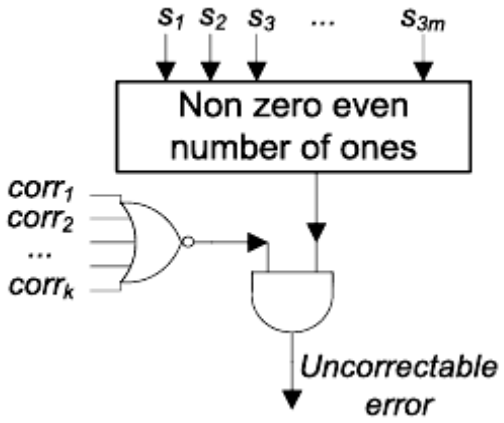

Fig 7: Double non adjacent error detection block

The errors that occur non adjacently can be detected using fig7, ie the check bits that are generated from the incoming data bits and the check bits that are already existing are compared to form S_n. Total number of error correction bits are are put together to check for non-zero data. The data should also be checked for even parity.

VI. SIMULATION ANALYSIS

Simulations are done using Xilinx and ModelSim. Several combinations of datas have been used to check the effectiveness of the correction code. Combinations of data for correcting double non adjacent errors have also been used to check the proposed scheme of error correction codes. The encoders and decoders have been implemented in Hardware Description Language (HDL) and synthesized for single error, double adjacent and double non adjacent errors and verified. The error detected also indicates the error occurrence state. If the there is an error detected then the state is one else the other way. For the former scheme there will a state where the error cannot be corrected. And so the non correctable error state is also mentioned. And the latter scheme has no such state still it is included for reference.

16 bit in Data = 1100000110000111
Check bits= 12

Number of errors	Encoded data	Decoded data	Error state	Non correctable error
No error	1110000 0000100 1010000 0010111	11000001100 00111	0	0
Single error	1111000 0000100 1010000 0010111	11000001100 00111	1	0
Double adjacent error	111110 0000100 1010000 0010111	11000001100 00111	1	0
Double non adjuacent	1111000 0000110 1010000 0010111	11000011100 00111	1	1

16 bit in Data = 1100000110000011
Check bits= 8

Number of errors	Encoded data	Decoded data	Error state	Non correctable errors
No error	101000010 101001110 000011	11000001100 00011	0	0
Single error	001000010 101001110 000011	11000001100 00011	1	0
Double adjacent error	101110010 101001110 000011	11000001100 00011	1	0
Double non adjacent error	001000010 101001110 000010	11000001100 00011	1	0

The simulations are also done for area, delay and power using Xilinx again. The area is in terms of gate count and delay in nanoseconds

ANALYSIS

	Area (gate count)	Delay (NS)	Power (mW)
Encoder SEC DAEC	72	7.85	24
Decoder	240	10.79	24
Encoder SEC DAEC DNAEC	48	7.85	24
Decoder	282	19.10	24

VII. CONCLUSION

A new class of SEC-DED-DAEC-DNAC codes has been presented. The codes are derived from DEC OLS codes and can be decoded with low latency. The noteworthy advantageous feature is that the codes do not experience miscorrections when double nonadjacent error occurs. This is found to be interesting to minimize silent data corruption. The codes can also correct nonadjacent double errors. Compared with existing SEC-DED-DAEC codes, they require a lesser number of parity check bits. The codes have been implemented in HDL and the resulting implementations compared with existing SEC-DED-DAEC codes.

REFERENCES

[1] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semi conductor memory applications: A state-of-the-art review," IBM J. Res. Develop., vol. 28, no. 2, pp. 124–134, Mar. 1984.
[2] M. Y. Hsiao, "A class of optimal minimum odd-weight column SEC-DED codes," IBM J. Res. Develop., vol. 14, no. 4, pp. 395–301, Jul. 1970.
[3] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in Proc. 25th IEEE VLSI Test Symp., May 2007, pp. 349–354.
[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] 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, USA: Prentice-Hall, 2004.

[7] S. Ghosh and P. D. Lincoln, "Dynamic low-density parity check codes for fault-tolerant nano-scale memory," in Proc.
[8] Found. Nanosci. FNAN, 2007.
[9] 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.
[10] 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.
[11] P. Reviriego, M. Flanagan, S. Liu, and J. A. Maestro, "Multiple cell upset correction in memories using difference set codes," IEEE Trans. Circuits Syst. I, Regular Papers, vol. 59, no. 11, pp. 2592–2599, Nov. 2012.