International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

COPY RIGHT

ELSEVIER SSRN

IJIEMR Transactions, online available on 15th Jun 2019. Link :http://www.ijiemr.org/downloads.php?vol=Volume-08&issue=ISSUE-07

Title: **LOW DELAY 4-BIT BURST ERROR CORRECTION CODES WITH QUADRUPLE ADJACENT ERROR CORRECTION**

Volume 08, Issue 07, Pages: 171–177.

Paper Authors

EDDATA MOUNIKA, M.AMARANATH REDDY, K.MAHESWARI

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per UGC Guidelines We Are Providing A Electronic Bar Code

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

LOW DELAY 4-BIT BURST ERROR CORRECTION CODES WITH QUADRUPLE ADJACENT ERROR CORRECTION

EDDATA MOUNIKA[1], M.AMARANATH REDDY[2], K.MAHESWARI[3]

[1]PG Scholar, Dept of ECE, SIR C.V. RAMAN Institute of Technology & Science, AP, India

[2] Assistant Professor, Dept of ECE, SIR C.V. RAMAN Institute of Technology & Science, AP, India

[3] Research Scholar, Dept of ECE, JNTUA, AP, India

ABSTRACT: The use of error-correction codes (ECCs) with advanced correction capability is a common system-level strategy to harden the memory against multiple bit upsets (MBUs). Therefore, the construction of ECCs with advanced error correction and low redundancy has become an important problem, especially for adjacent ECCs. Existing codes for mitigating MBUs mainly focus on the correction of up to 3-bit burst errors. As the technology scales and cell interval distance decrease, the number of affected bits can easily extend to more than 3 bit. The previous methods are therefore not enough to satisfy the reliability requirement of the applications in harsh environments. In this paper, a technique to extend 3-bit burst error-correction (BEC) codes with quadruple adjacent error correction (QAEC) is presented. First, the design rules are specified and then a searching algorithm is developed to find the codes that comply with those rules. The *H* matrices of the 3-bit BEC with QAEC obtained are presented. They do not require additional parity check bits compared with a 3-bit BEC code. By applying the new algorithm to previous 3-bit BEC codes, the performance of 3-bit BEC is also remarkably improved.

1. INTRODUCTION

Error correction codes are commonly used to protect memories from so called Soft Errors, which change the logical value of memory cells without damaging the circuit. As technology scales, memory devices become larger and more powerful error correction codes are needed. To this end the use of more advanced codes has been recently proposed. These codes can correct a larger number of errors, but generally require complex decoders. To avoid a high decoding complexity, the use of one-step majority logic decodable codes was first proposed in for memory applications. One step majority logic decoding can be implemented serially with very simple circuitry but requires long decoding times. In a memory this would increase the access time. Only few classes of codes can be decoded using OS-MLD. Among those are some DS-LDPC codes, EG-LDPC codes and OLS codes.The use of OLS codes has gained renewed interest for interconnections, memories, and caches. This is due to their modularity such that the error correction capabilities can be easily adapted to the error rate or to the mode of operation. 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.A method was recently proposed in to accelerate a serial implementation of majority logic decoding of DS-LDPC codes. The idea behind the method is to use the first iterations of majority logic decoding to detect if the word being decoded contains errors. If there are no errors, then decoding can be stopped without completing the remaining iterations, therefore greatly reducing the decoding time. And majority logic decoding can be implemented serially with simple hardware but requires a large decoding time. For memory applications this increases the memory access time. The method detects whether a word has errors in the first iterations of majority logic decoding, and when there are no errors the decoding ends without completing the rest of the iterations. Since most words in a memory will be error-free, the average decoding time is greatly reduced.

2. Existing Method

As technology scales, memory devices become larger and more powerful error correction codes are needed. To this end, the use of more advanced codes has been recently proposed. These codes can correct a larger number of errors, but generally require complex decoders. To avoid a high decoding complexity, the use of one step majority logic decodable codes was first proposed in for memory applications. Further work on this topic was then presented in. One step majority logic decoding can be implemented serially with very simple circuitry, but requires long decoding times. In a memory, this would increase the access time which is an important system parameter. Only a few classes of codes can be decoded using one step majority logic decoding. Among those are some Euclidean geometry low density parity check (EG-LDPC) codes which were used in, and difference set low density parity check (DS-LDPC) codes. This method was proposed to accelerate the majority logic decoding of difference set low density parity check codes. this is useful as majority logic decoding can be implemented serially with simple hardware but requires a large decoding time. for memory applications, this increases the memory access time. the method detects whether a word has errors in the first iterations of majority logic decoding, and when there are no errors the decoding ends without completing the rest of the iterations. since most words in a memory will be error-free, the average decoding time is greatly reduced. in this brief, we study the application of a similar technique to a class of euclidean geometry low density parity check (EG-LDPC) codes that are onestep majority logic

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

decodable. the results obtained show that the method is also effective for EG-LDPC codes. extensive simulation results are given to accurately estimate the probability of error detection for different code sizes and numbers of errors. A method was recently proposed in to accelerate a serial implementation of majority logic decoding of DS-LDPC codes. The idea behind the method is to use the first iterations of majority logic decoding to detect if the word being decoded contains errors. If there are no errors, then decoding can be stopped without completing the remaining iterations, therefore greatly reducing the decoding time. For a code with block length N, majority logic decoding (when implemented serially) requires N iterations, so that as the code size grows, so does the decoding time. In the proposed approach, only the first three iterations are used to detect errors, thereby achieving a large speed increase when N is large. In it was shown that for DS-LDPC codes, all error combinations of up to five errors can be detected in the first three iterations. Also, errors affecting more than five bits were detected with a probability very close to one. The probability of undetected errors was also found to decrease as the code block length increased. For a billion error patterns only a few errors (or sometimes none) were undetected. This may be sufficient for some applications. Another advantage of the proposed method is that it requires very little additional circuitry as the decoding circuitry is also used for error detection. For example, it was shown in that the additional area required to implement the scheme was only around 1% for large word sizes.One step MLD can be implemented serially using the scheme in Fig 3.1 which corresponds to the decoder for the EG LDPC code with N=15. First the data block is loaded into the registers. Then the check equations are computed and if a majority of them has a value of one, the last bit is inverted. Then all bits are cyclically shifted. This set of operations constitutes a single iteration: after N iterations, the bits are in the same position in which they were loaded. In the process, each bit may be corrected only once. As can be seen, the decoding circuitry is simple, but it requires a long decoding time if Nis large.

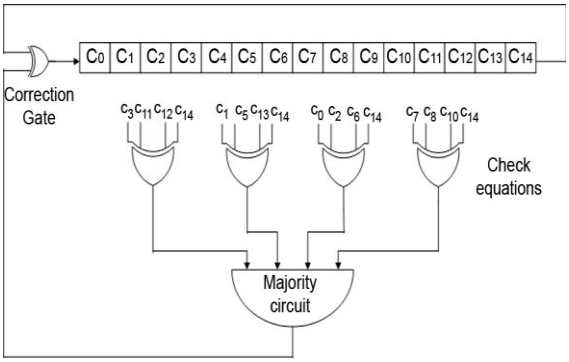

Fig. 1 Serial one-step majority logic decoder for the EG-LPDC code

The check equations must have the following properties

1. All equations include the variable whose value is stored in the last register (the one marked as C_{14}).
2. The rest of the registers are included in at most one of the check equations.

If errors can be detected in the first few iterations of MLD, then whenever no errors are detected in those iterations, the decoding can be

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

stopped without completing the rest of the iterations. In the first iteration, errors will be detected when at least one of the check equations is affected by an odd number of bits in error. In the second iteration, as bits are cyclically shifted by one position, errors will affect other equations such that some errors undetected in the first iteration will be detected. As iterations advance, all detectable errors will eventually be detected. In it was shown that for DS-LDPC codes most errors can be detected in the first three iterations of MLD. Based on simulation results and on a theoretical proof for the case of two errors, the following hypothesis was made. "Given a word read from a memory protected with DS-LDPC codes, and affected by up to five bit-flips, all errors can be detected in only three decoding cycles". Then the proposed technique was implemented in VHDL and synthesized, showing that for codes with large block sizes the overhead is low. This is because the existing majority logic decoding circuitry is reused to perform error detection and only some extra control logic is needed.

3. IMPLEMENTATION OF PROPOSED ARCHITECTURE

OLS codes are based on the concept of Latin squares. A Latin square of size m is an $m \times m$ matrix that has permutations of the digits 0, 1,…, $m-1$ in both its rows and columns. Two Latin squares are orthogonal if when they are superimposed every ordered pair of elements appears only once. OLS codes are derived from OLS. 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. 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.

$$H=\begin{bmatrix}
1&0&0&0&0&0&0&1&0&0&1&0&1&0&1&1&1&0&0&0&1&0&0\\
0&1&0&0&0&0&0&0&1&0&1&1&0&1&1&0&0&1&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&1&1&0&1&0&0&1\\
0&0&0&1&0&0&0&1&0&0&1&0&0&0&1&0&1&0&1&1&0&0&0\\
0&0&0&0&1&0&0&0&1&0&0&1&1&1&1&0&0&0&0&1&1&0&0\\
0&0&0&0&0&1&0&0&1&0&0&0&1&1&0&0&0&1&0&1&0&0&1\\
0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&1&0&1&1&1&0&1&1
\end{bmatrix}$$

Fig.2. Parity check matrix for OLS code with $k = 16$ and $t = 1$.

(1) 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

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

checking against the stored parity check bits. 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. 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:

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

where $I2m$ is the identity matrix of size $2m$ and $M1$, $M2$ are matrices of size $m \times m2$. The matrix $M1$ has m ones in each row. For the r th row, the ones are at positions $(r-1) \times m + 1, (r-1) \times m + 2, \ldots (r-1) \times m + m - 1, (r-1) \times m + m$. The matrix $M2$ is constructed as follows:

$M2 = [Im\ Im \ldots Im]$.

(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

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

. (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.

1) Each data bit participates exactly in $2t$ parity checks.

2) 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.

4.Proposed Concurrent Error Detection Technique

Based on the structure of the parity check matrix, the check bits are calculated by the corresponding data bits. The new encoded codeword, the combination of check bits and data bits is stored in the memory. When the particles hit the memory resulting in MBUs, the contents of affected memory cells are flipped. Here, to elaborate on the correction ability of QAEC codes, quadruple adjacent bits are flipped on $D2$, $D3$, $D4$, and $D5$. In the decoding process, the syndrome is calculated using the stored check bits and data bits and the structure of the parity check matrix. Through the corresponding relationship between the syndrome and the XOR result of the columns mentioned in Section II, the flipped bits can be located. With the flipped bits inverted, the errors from the storage stage in the memory are effectively corrected. This is the whole procedure of encoding and decoding for the proposed QAEC codes

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

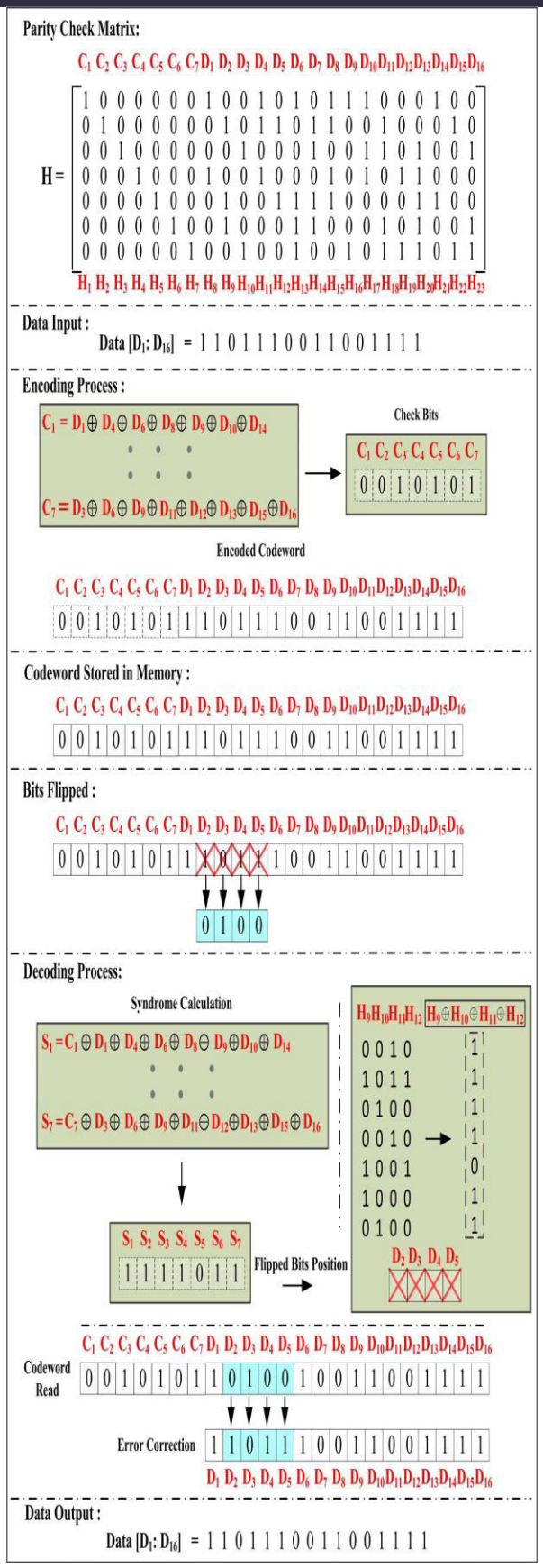

Fig 3 Implementation Flow

5.RESULTS

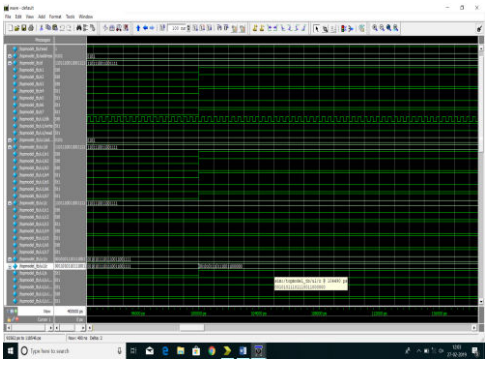

Fig 4 Simulation Result for Proposed Method

6. CONCLUSION AND FUTURE SCOPE

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. 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] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. Device Mater. Reliab.*, vol. 5, no. 3, pp. 301–316, Sep. 2005.

[2] M. A. Bajura, Y. Boulghassoul, R. Naseer, S. DasGupta, A. F.Witulski, J. Sondeen, S. D. Stansberry, J. Draper, L. W. Massengill, and J. N. Damoulakis, "Models and algorithmic limits for an ECC-based approach to hardening sub-100-nm SRAMs," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 4, pp. 935–945, Aug. 2007.

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

[4] S. Ghosh and P. D. Lincoln, "Dynamic low-density parity check codes for fault tolerant nano-scale memory," presented at the Foundations Nanosci. (FNANO), Snowbird, Utah, 2007.

[5] S. Ghosh and P. D. Lincoln, "Low-density parity check codes for error correction in nanoscale memory," SRI Computer Science Lab., Menlo Park, CA, Tech. Rep. CSL-0703, 2007.

[6] H. Naeimi and A. DeHon, "Fault secure encoder and decoder for memory applications," in *Proc. IEEE Int. Symp. Defect Fault Toler. VLSI Syst.*, 2007, pp. 409–417.

[7] B. Vasic and S. K. Chilappagari, "An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 11, pp. 2438–2446, Nov. 2007.

[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. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004.

[10] S. Liu, P. Reviriego, and J. 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] H. Tang, J. Xu, S. Lin, and K. A. S. Abdel-Ghaffar, "Codes on finite geometries," *IEEE Trans. Inf. Theory*, vol. 51, no. 2, pp. 572–596, Feb. 2005.