IMPLEMENTING 4-BIT BURST ERROR CORRECTION CODES FOR ADJACENT MULTIPLE BIT UPSETS TOLERANCE IN SRAMS

N.PRAMODA [1], V.ANNAPURNA [2]
[1]PG SCHOLAR.DEPT OF ECE, JNTUA, ANANTAPUR, AP, INDIA
[2]ASSISTANT PROFESSOR, DEPT OF ECE, JNTUA,ANANTAPUR, 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 implement 4-bit burst error-correction (BEC) codes i.e., 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 4-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 4-bit BEC is also remarkably improved. The encoding and decoding procedure of the proposed codes is illustrated with an example. Then, the encoders and decoders are implemented and the results show that our codes have moderate total area and delay overhead to achieve the correction ability extension.

Index Terms— Burst error-correction codes (ECCs), ECC, multiple bit upset (MBU), memory, quadruple adjacent error correction (QAEC).

1. INTRODUCTION

Reliabity is a significant necessity for space applications. Recollections as the information putting away segments assume a huge function in the electronic frameworks. They are broadly utilized in the framework on a chip and application-explicit coordinated circuits. In these applications, recollections represent a huge part of the circuit territory. This gains experiences endure more space radiation than different segments. Along these lines, the affectability to radiation of recollections has become a basic issue to guarantee the dependability of electronic frameworks. In present day static arbitrary access recollections (SRAMs), radiation-prompted delicate mistakes as the single function upset (SEU) and different piece upset (MBU) are two noticeable single function impacts. As semiconductor innovation creates from the sub micrometer innovation to the ultra profound sub micrometer (UDSM) innovation, the size of memory cells is more modest and more cells

Error correction codes are normally used to shield recollections from alleged Soft Errors, which change the intelligent estimation of memory cells without harming the circuit. As innovation scales, memory gadgets become bigger and all the more remarkable blunder amendment codes are required. To this end the utilization

of further developed codes has been as of late proposed. These codes can address a bigger number of blunders, yet by and large require complex decoders. To maintain a strategic distance from a high interpreting unpredictability, the utilization of one-venture lion's share rationale decodable codes was first proposed in for memory applications. One stage greater part rationale interpreting can be executed sequentially with extremely basic hardware yet requires long unraveling occasions. In a memory this would expand the entrance time. Just scarcely any classes of codes can be decoded utilizing OS-MLD. Among those are some DS-LDPC codes, EG-LDPC codes and OLS codes.

The utilization of OLS codes has increased recharged interest for interconnections, recollections, and stores. This is because of their particularity with the end goal that the mistake adjustment abilities can be effectively adjusted to the blunder rate or to the method of activity. OLS codes normally require more equality bits than different codes to address similar number of blunders. Notwithstanding, their seclusion and the basic and low postpone deciphering execution (as OLS codes are OS-MLD), balance this drawback in numerous applications. A significant issue is that the encoder and decoder circuits expected to utilize (ECCs) can likewise endure mistakes. At the point when a blunder influences the encoder, an off base word might be composed into the memory. A mistake in the decoder can make a right word be deciphered as mistaken or the reverse way around, an off base word to be deciphered as a right word.

A strategy was as of late proposed in to quicken a sequential execution of dominant part rationale translating of DS-LDPC codes. The thought behind the strategy is to utilize the main cycles of greater part rationale deciphering to recognize if the word being decoded contains mistakes. In the event that there are no mistakes, at that point interpreting can be halted without finishing the leftover emphasess, subsequently incredibly diminishing the unraveling time. What's more, greater part rationale unraveling can be executed sequentially with basic equipment yet requires a huge deciphering time. For memory applications this builds the memory access time. The technique identifies whether a word has blunders in the main emphasis of lion's share rationale deciphering, and when there are no mistakes the unraveling closes without finishing the remainder of the cycles. Since most words in a memory will be without blunder, the normal translating time is incredibly diminished.

2. Existing Method

As innovation scales, memory gadgets become bigger and all the more impressive blunder amendment codes are required. To this end, the utilization of further developed codes has been as of late proposed. These codes can address a bigger number of mistakes, however by and large require complex decoders. To stay away from a high interpreting unpredictability, the utilization of one stage larger part rationale decodable codes was first proposed in for memory applications. Further work on this point was then introduced in. One stage lion's share rationale disentangling can be executed sequentially with exceptionally basic hardware, yet requires long interpreting times. In a memory, this would expand the entrance time which is a significant framework boundary. A couple of classes of codes can be decoded utilizing one stage larger part rationale unraveling. Among those are some Euclidean math low thickness equality check (EG-LDPC) codes

which were utilized in, and contrast set low thickness equality check (DS-LDPC) codes.

This strategy was proposed to quicken the larger part rationale disentangling of distinction set low thickness equality check codes. this is valuable as larger part rationale translating can be executed sequentially with straightforward equipment however requires an enormous unraveling time. for memory applications, this expands the memory access time. the technique distinguishes whether a word has blunders in the principal cycles of greater part rationale translating, and when there are no mistakes the unraveling closes without finishing the remainder of the emphasess. since most words in a memory will be without blunder, the normal interpreting time is incredibly diminished. in this short, we study the use of a comparative procedure to a class of euclidean calculation low thickness equality check (EG-LDPC) codes that are onestep greater part rationale decodable. the outcomes acquired show that the strategy is likewise compelling for EG-LDPC codes. broad reenactment results are given to precisely gauge the likelihood of mistake discovery for various code sizes and quantities of blunders.

A strategy was as of late proposed in to quicken a sequential usage of larger part rationale translating of DS-LDPC codes. The thought behind the technique is to utilize the main emphasess of larger part rationale disentangling to distinguish if the word being decoded contains mistakes. On the off chance that there are no mistakes, at that point unraveling can be halted without finishing the excess cycles, along these lines extraordinarily decreasing the disentangling time. For a code with block length N, larger part rationale disentangling (when executed sequentially) requires N emphasess, so that as the code size develops, so does the translating time. In the proposed approach, just the initial three cycles are utilized to identify mistakes, in this manner accomplishing an enormous speed increment when N is huge. In it was indicated that for DS-LDPC codes, all blunder mixes of up to five mistakes can be distinguished in the initial three emphasess. Likewise, mistakes influencing in excess of five pieces were recognized with a likelihood near one. The likelihood of undetected blunders was additionally found to diminish as the code block length expanded. For a billion mistake designs a couple of blunders (or some of the time none) were undetected. This might be adequate for certain applications. Another bit of leeway of the proposed technique is that it requires almost no extra hardware as the disentangling hardware is likewise utilized for blunder discovery. For instance, it was appeared in that the extra zone needed to actualize the plan was distinctly around 1% for enormous word sizes.

One stage MLD can be actualized sequentially utilizing the plan in Fig 3.1 which compares to the decoder for the EG LDPC code with N=15. First the information block is stacked into the registers. At that point the check conditions are registered and if a larger part of them has an estimation of one, the last piece is upset. At that point all pieces are consistently moved. This arrangement of tasks establishes a solitary emphasess: after N emphasess, the pieces are similarly situated in which they were stacked. Simultaneously, each piece might be revised just a single time. As can be seen, the deciphering hardware is straightforward, yet it requires a long unraveling time if Nis enormous.

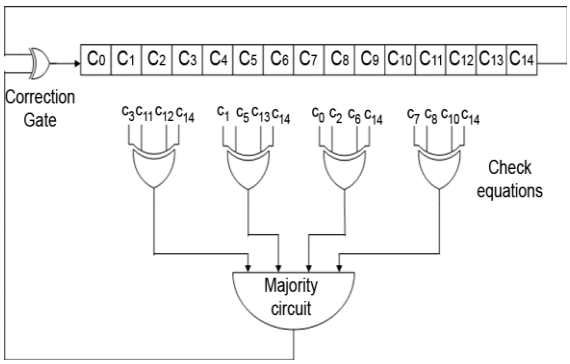

Fig. 1 Serial one-step majority logic decoder for the EG-LPDC code

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 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 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 $D1$, $D2$, $D3$, and $D4$. 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

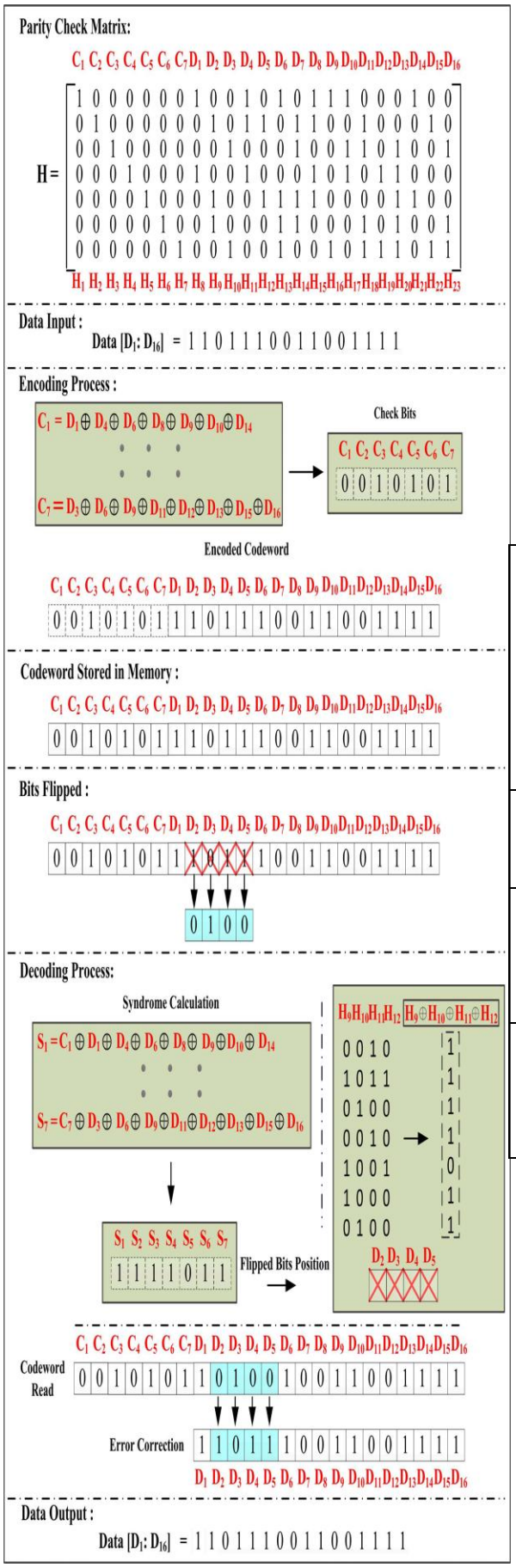

Fig 3 Implementation Flow

5.RESULTS

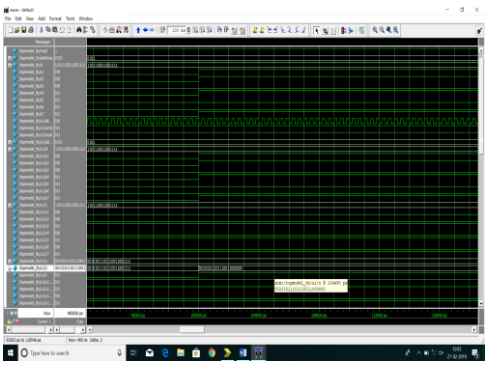

Fig 4 Simulation Result for Proposed Method

6.COMPRESSION TABLE

NO OF BITS	Codes	DELAY(NS)	
		ENCODER	DECODER
16 BITS	Existing Code	7.305	7.747
16 BITS	Implemented Code	7.055	7.016

7. CONCLUSION AND FUTURE SCOPE

In this paper, a technique to implement the 4-bit BEC codes with the QAEC is presented. The proposed codes have the same redundancy as the previous 3-bit BEC codes. To accelerate the searching process of target matrices, a new algorithm with column weight control and recording function is

proposed. Based on the proposed algorithm, a searching tool is developed to execute the searching process automatically. To prove the validity of the proposed algorithm, it is applied to the previous 3-bit BEC codes and the codes are remarkably improved on the two optimization criteria. Then, the proposed algorithm is used to find the solution for the QAEC. The complete solution searching process is finished for 16 data bits and the searching process using optimization algorithm is carried out for 32 and 64 data bits. Therefore, in this paper, the best solutions are presented for 16 data bits and the best solutions found in a reasonable computation time are presented for 32 and 64 data bits. The encoder and decoder of the proposed codes are implemented by using the HDL. The overhead of area and delay is moderate versus previous 3-bit BEC codes. This suggests that the proposed 4-bit BEC with QAEC codes can be effectively used by designers to protect the SRAM memories from radiation effect and mitigate the MBUs that affect up to four adjacent bits.

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.