International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 9 Issue 3 –SEPTEMBER 2014.

Error Identification to translate majority Logic design by using EG-LDPC codes

[*1]Janaki Jaya Santhosh, and [#2]J. Ravi

[*]*PG Student, KDEG- Gooty, Affiliated to JNTUA University*

[#]*Associate Professor (ECE), KDEG- Gooty, Affiliated to JNTUA University*

[1]jayasanthosh.janaki@gmail.com

[2]jraviskd@gmail.com

***Abstract*— Euclidean Geometry LDPC (EG-LDPC) codes — enable dynamic changes in level of fault tolerance. An EG-LDPC code enables us to dynamically adjust the error correcting capacity for improved system performance apart from the high error correcting capability as well as sparsity. Memories are typically protected with error correction codes to prevent soft errors from causing data corruption. The MLD codes are used for memory application as because of correcting large number of soft errors, less decoding time, area consumption, etc. But Majority logic decoding can be implemented serially with simple hardware but requires a large decoding time compared to difference set low density parity check codes. The combined method of MLD with EG-LDPC 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.**

Keywords: Error correction codes, Euclidean geometry low-density parity check (EG-LDPC) codes, majority logic decoding (MLD).

I. INTRODUCTION

Low Density Parity Check codes[3] are block codes with parity-check matrices contains a very small number of non-zero entries; guarantees both decoding complexity and the minimum distance which increases linearly with the code length. The existing block codes can be successfully used with the LDPC iterative decoding algorithms if represented as a sparse parity-check matrix. LDPC codes are generally decoded iteratively using a graphical representation of their parity-check matrix.

EG-LDPC codes are derived from Euclidean Geometries. Error correction codes are used to protect memories from soft errors which change the logical values of memory cells without damaging the circuitry. Euclidean geometry low density parity check (EG- LDPC) codes are part of Majority Logic Decodable code. This type of code uses the check sum algorithm. The effectiveness of an ECC code in a memory system is evaluated by calculating the probability of getting an uncorrectable error within a word. An uncorrectable error occurs when the number of upset bits (or words) exceeds what can be corrected by the ECC between consecutive memory scrubbing or other access cycles.

II. EXISTING SYSTEM

The soft error is also often referred to as a Single Event Upset (SEU) caused because of a radiation event to reverse or flip the data state of a memory cell. . Majority logic decoding codes are cyclic codes constructed based on number of parity check equations [6]. Though many of the error detection techniques are available, the MLD is a simple detection technique. But the limitation is it increases the average latency of the decoding process because it depends on the size of the code. Also, MLD is based on a number of parity check equations which are orthogonal to each other; at each iteration, each codeword bit only participates only in one parity check equation, except the very first bit which contributes to all equations. ML decoding is that, for a coded word of N - bits, it takes N cycles in the decoding process affecting the system performance. The existing ML-decoders Plain ML Decoder and Plain MLD with Syndrome Fault Detector (SFD) though are powerful decoders depends up on the number of parity check equation and also complex to design.

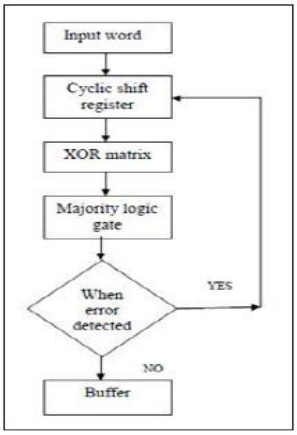

Fig 1: Flowchart of a Majority Logic Decoding Algorithm

EG-LDPCs are decodable using a one-step majority decoder having a modular structure also enables low fan-in circuit implementation. Such modularity makes modularity makes EG-LDPC code dynamic and hence easy to implement even using nanoscale hardware [5].

III. PROPOSED SYSTEM

In the proposed work, the MLD techniques have been combined with EG-LDPC technique based on the structure of Euclidean geometries over a Galois field. Majority logic decoding is a method to decode repetition codes, based on the the largest number of occurrences of a transmitted symbol. At each iteration, each code word bit only participates in one parity check equation which is orthogonal to each other.

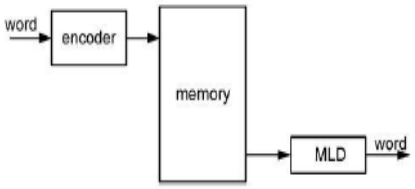

Fig 2: Block Diagram of a MLD

From the above Fig 2: it is clear that initially, the data words are encoded and then stored in the memory. When the memory is read, the code word is then fed through the ML decoder before sent to the output for further processing. In this decoding process, the data word is corrected from all bit-flips that it might have. We have two types of MLD decoders:

Type-I ML decoder - determines which bits need to be corrected based upon XOR combinations

Type-II ML decoder - calculates directly out of the code word bits, the information of correctness of the current bit under decoding.

EG-LDPC codes have block lengths close to a power of two, fitting well to the requirements of modern memory systems, thereby reducing the cost of the modern memory systems.

Euclidean Geometry Low Density Parity Check Codes: An m-dimensional Euclidean Geometry over the Galois Field $\mathrm{GF}(2^s)$ denoted by $\mathrm{EG}(m, 2^s)$ is formed by the 2^{ms} m-tuples over $\mathrm{GF}(2^s)$ by defining vector addition and scalar multiplication as follows [9]:

$$\begin{aligned}&(a_0, a_1, \ldots, a_{m-1}) + (b_0, b_1, \ldots, b_{m-1})\\ &\quad = (a_0 + b_0, a_1 + b_1, \ldots, a_{m-1} + b_{m-1})\\ &\beta \cdot (a_0, a_1, \ldots, a_{m-1})\\ &\quad = (\beta \cdot a_0, \beta \cdot a_1, \ldots, \beta \cdot a_{m-1}).\end{aligned} \quad (1)$$

Each m-tuple is called a *point* in $\mathrm{EG}(m, 2^s)$, with the all zero m-tuple being called the *origin*. A line in $\mathrm{EG}(m, 2^s)$ is formed by 2^s points and can be expressed as $\{a + \beta c\}$ where a and c are points in $\mathrm{EG}(m, 2^s)$.

One step MLD can be implemented serially using the scheme in Fig. 1 which communicates to the decoder for the EG-LDPC code with N=15. If errors can be detected in the first few iterations of MLD, after that on every occasion no errors are detected in those iterations, the decoding can be congested without finishing the rest of the iterations. In the first iteration, errors will be identified when at least one of the check equations is affected by an odd number of bits in error. In the following iteration, as bits are intermittently shifted by one position, errors will impinge on other equations such that some errors unidentified in the first iteration will be identified. The advantage of the proposed method is that it requires very little additional circuitry as the decoding circuitry is also used for error detection.

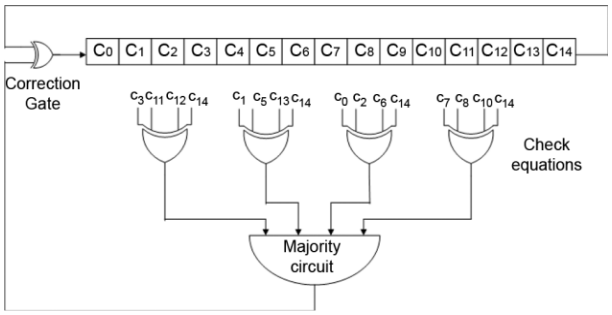

Fig 3: Serial one-step majority logic decoder for the (15,7) EG-LPDC code.

In general, For a code with block length *N* majority logic decoding requires *N* iterations so that as the code size grows, the decoding time also grows linearly. In the proposed approach, only the first three iterations are used to detect errors, thereby achieving a large speed increase when *N* is large.

TABLE I
ONE STEP MLD EG-LDPC CODES

N	K	J	t_{ML}
15	7	4	2
63	37	8	4
255	175	16	8
1023	781	32	16

The advantage of the proposed method is that it requires very little additional circuitry as the decoding circuitry is also used for error detection. The experimental results can be proved as follows:

Lemma 1: The MLD check equations in a one step MLD EG-LDPC code are all cyclic shifts of one another.

Proof: Since there are $2^{2s}-1$ different incidence vectors and there are $2^{2s}-1$ cyclic shifts of one vector also (since $N=2^{2s}-1$), we can conclude that all the vectors are made from the cyclic shift of a single vector which defines the code. Therefore the equations for MLD may all be obtained from cyclically shifting this single vector.

Lemma 2: There is no MLD check equation which has two ones at a distance $k\cdot(2^s+1)$ with $k=1,2,\ldots 2^s-2$.

Proof: Suppose there are two such ones in an equation, then as equations are shifted to obtain the rest of the equations, at some point there will be an equation with a one on position $2^{2s}-1-k\cdot(2^s+1)=(2^s-1-k)\cdot(2^s+1)$ which would contradict one of the properties of the EG-LDPC codes.

Lemma 3: Every pair of ones in a check equation is at a different distance.

Proof: Suppose that there is another pair of ones at the same distance in the check equation. Since every check equations corresponds to a line in the Euclidean geometry, and any cyclic shift of such a line necessarily yields another line, then it may be seen that shifting the check equation yields a line which shares two points with the first one. This is not possible as in a Euclidean geometry, two distinct lines cannot share the same two points.

Using the above Lemma 1,2,3, we have the following theorem:

Theorem: Given a block protected with a one step MLD EG-LDPC code, and affected by two bit-flips, these can be detected in only three decoding cycles.

Proof: Let us consider the different situations that can occur for errors affecting two bits. An error will be detected in the first iteration unless a) they occur in bits which are not checked by any equation, or b) both errors occur in bits which are checked by the same MLD equation in the first iteration.

For case a), the properties of the code force the bits in error to be at a distance $k\cdot(2^s+1)$. Therefore, the error will be detected in the second iteration unless there are two ones in the MLD vector at a distance $k\cdot(2^s+1)$. This cannot be the case due to Lemma 2. Therefore the error must be detected in the second iteration. For case b), the separation of the bits which are not checked by any equation means that it is not possible in the second and third iteration for the two errors not to be checked by any equation.

Also, using Lemma 3 for case b), in the second iteration the bits will be checked by a single equation again only if this second equation is simply the previous one shifted by one position. The same applies to the rest of the iterations: if the bits are checked by one equation then it must be the one in the previous iteration shifted by one position. Finally there can not be three MLD equations that are consecutive shifts as that would mean that there are three consecutive ones in the equations. This would mean that at least one register apart from the last one is checked by more than one equation and therefore the code would not be one step MLD. Therefore the errors will always be detected in the first three iterations.

IV. CONCLUSION AND FUTURE WORK

In the implemented work, the detection of errors during the first iterations of serial one step Majority Logic Decoding of EG-LDPC codes has been studied. The objective of the implemented system is to reduce the decoding time by stopping the decoding process when no errors are detected.

Future work includes extending the theoretical analysis to the cases of three and four errors. Also, the scope can be extended to determine the number of iterations to detect errors affecting a number of bits gives a tradeoff between decoding time and error detection capability.

REFERENCES

[1] Pedro Reviriego, Juan A. Maestro, and Mark F. Flanagan, "Error Detection in Majority Logic Decoding of Euclidean Geometry Low Density Parity Check (EG-LDPC) Codes".

[2] M. A. Bajura, Y. Boulghassoul, R. Naseer, S. DasGupta, A. F.Witulski, J. Sondeen, S. D. Stansberry, J. Draper, L. W.ssengill, 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] 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. 200S.

[6] 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.