Error Detection in Majority Logic Decoding of Euclidean Geometry Low Density Parity Check (EG-LDPC) Codes

S Sufia Anjum, S Nazneen Shagufa, K Vali Babu

Abstract— **In a recent paper, a 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 one step majority logic 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.**

Index Terms— Error correction codes, Euclidean geometry low-density parity check (EG-LDPC) codes, majority logic decoding, memory.

I. 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 [1]. As technology scales, memory devices become larger and more powerful error correction codes are needed [2], [3]. To this end, the use of more advanced codes has been recently proposed [4]–[8].

Manuscript received May 14, 2014.

S Sufia Anjum, E.C.E Dept., Dr.K.V.Subba Reddy College Of Engineering For Women, India.

S Nazneen Shagufa, E.C.E Dept., Dr.K.V.Subba Reddy College Of Engineering For Women, India

K Vali Babu, E.C.E Dept., Dr.K.V.Subba Reddy College Of Engineering For Women, India

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 [4] for memory applications. Further work on this topic was then presented in [5], [6], [8]. One step majority logic decoding can be implemented serially with very simple circuitry [9], 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 [9]. Among those are some Euclidean geometry low density parity check (EG-LDPC) codes which were used in [4], and difference set low density parity check (DS-LDPC) codes [9].

A method was recently proposed in [10] 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 er- rors, 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 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 in- crease when is large. In [10] it was shown that for DS-LDPC codes, all error combinations of up to five errors can be detected in the first

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

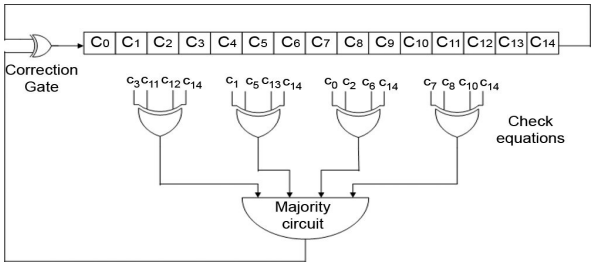

Fig. 1. Serial one-step majority logic decoder for the (15,7) EG-LPDC code

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 un- detected. 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 [10] that the additional area required to implement the scheme was only around 1% for large word sizes.

The method proposed in [10] relies on the properties of DS-LDPC codes and therefore it is not directly applicable to other code classes In the following, a similar approach for EG-LDPC codes is presented

The rest of this brief is divided into the following sections. Section II provides preliminaries on EG-LDPC codes, majority logic decoding and the method proposed in [10]. Section III presents the results of applying the method to EG-LDPC codes, comprising simulation results and a hypothesis based on those results. This is complemented by a theoretical proof for the cases of one and two errors that is provided in an Appendix.

II. PRELIMINARIES

Finite geometries have been used to derive many error-correcting codes [9], [11]. One example are EG-LDPC codes which are based on the structure of Euclidean geometries over a Galois field. Among EG-LDPC codes there is a subclass of codes that is one step majority logic decodable (MLD) [9]. Codes in this subclass are also cyclic. The parameters for some of these codes are given in Table I, where N is the block size, K the number of information bits, J the number of MLD check equations and t_{ML} the number of errors that the code can correct using one step MLD.

One step MLD can be implemented serially using the scheme in Fig. 1 which corresponds to the decoder for the EG-LDPC code with N=1.5. 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 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 N is large.

The check equations must have the following properties (see [9] for more details).

TABLE II
UNDETECTED ERRORS IN EXHAUSTIVE CHECKING

N	1 error	2 errors	3 errors	4 errors
15	0	0	0	0
63	0	0	0	0
255	0	0	0	–
1023	0	0	--	–

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 [10] 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.

III. RESULTS

The method proposed in [10] has been applied to the class of one step MLD EG-LDPC codes. To present the results, the conclusions are presented first in terms of a hypothesis that is then validated by simulation and also partially by a theoretical analysis presented in the

Appendix. The results obtained can be summarized in the following hypothesis

"Given a word read from a memory protected with one step MLD EG-LDPC codes, and affected by up to four bit-flips, all errors can be detected in only three decoding cycles".

Note that this hypothesis is different from the one made for DS-LDPCs codes in [10] as in that case errors affecting up to five bits were always detected. This is due to structural differences between DS-LDPC and EG-LDPC codes, which will be detailed in the Appendix. To validate the above hypothesis, the EG-LDPC codes considered have been implemented and tested. For codes with small words and affected by a small number of bit flips, it is practical to generate and check all possible error combina- tions. As the code size grows and the number of bit flips increases, it is no longer feasible to exhaustively test all possible combinations .Therefore the simulations are done in two ways, by exhaustively checking all error combinations when it is feasible and by checking randomly generated combinations in the rest of the cases.

The results for the exhaustive checks are shown in Table II. These results prove the hypothesis for the codes with smaller word size (15 and 63). For N=255 up to three errors have been exhaustively tested while for N=1023 only single and double error combinations have been exhaustively tested.

TABLE III
UNDETECTED ERRORS WITH ONE BILLION RANDOM ERROR COMBINATIONS

N	5 errors	6 errors	7 errors	8 errors	9 errors	10 errors	11 errors	12 errors
63	5672	5422	1079	1174	823	817	537	549
255	23	10	0	0	0	1	0	0
1023	0	1	0	0	0	0	0	0

To complement the results of the exhaustive checks for larger codes and number of errors, simulations using random error patterns have been used. In all the experiments, one billion error combinations are tested. The results for errors affecting more than four bits are shown in Table III, since for errors affecting up to four bits there were no undetected errors. It can be observed that for errors affecting more than four bits there is a small number of error combinations that will not be detected in the first three iterations. This number decreases with word size and also with the number of errors. The decrease with the word size can be explained as follows: the larger the word size, the larger the number of MLD check equations (see Table I) and therefore it is more unlikely that errors occur in the same equation. As for the number of errors, a similar reasoning applies: the more errors occur, the larger the probability that an odd number of errors occurs in at least one equation. Finally it must be noted that the probabilities of undetected errors are different for an even and an odd number of errors as in the latter case, one of the errors must occur in a bit which is not checked by any equation.

The simulation results presented suggest that all errors affecting three and four bits would be detected in the first three iterations. For errors affecting a larger number of bits, there is a small probability of not being detected in those iterations. For large word sizes, the probabilities are sufficiently small to be acceptable in many applications.

In summary, the first three iterations will detect all errors affecting four or fewer bits, and almost every other detectable error affecting more bits. This is a slightly worse performance than in the case of DS-LDPC codes [10] where errors affecting five bits were addition- ally always detected. However, the majority logic circuitry is simpler for EG-LDPC codes, as the number of equations is a power of two and an approach based on sorting networks proposed in [8] can be used to reduce the cost of the majority logic voting. In addition, EG-LDPC codes have block lengths close to a power of two, thus fitting well to the requirements of modern memory systems. This may mean that in some cases it may be more convenient to use an EG-LDPC code and keep a word size compatible with existing designs (power of two) than using a DS-LDPC code requiring a different word size or a shortened version of that code. When using word size which is a power of two, there would be a bit which is not used by the EG-LDPC code (see Table I). This bit can be used for a parity covering all bits in the word that would detect all errors affecting an odd number of bits. In that case, the design using the EG-LDPC would also detect all errors affecting five or fewer bits.

IV. CONCLUSION

In this brief, the detection of errors during the first iterations of serial one step Majority Logic Decoding of EG-LDPC codes has been studied. The objective was to reduce the decoding time by stopping the decoding process when no errors are detected. The simulation results show that all tested combinations of

errors affecting up to four bits are detected in the first three iterations of decoding. These results extend the ones recently presented for DS-LDPC codes, making the modified one step majority logic decoding more attractive for memory applications. The designer now has a larger choice of word lengths and error correction capabilities.

Future work includes extending the theoretical analysis to the cases of three and four errors. More generally, determining the required number of iterations to detect errors affecting a given number of bits seems to be an interesting problem. A general solution to that problem would enable a fine-grained tradeoff between decoding time and error detection capability.

APPENDIX

In this Appendix, a theoretical proof is presented for the case of errors affecting one or two bits. We begin by introducing the necessary background information about EG-LDPC codes.

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

$(a_0,a_1,\ldots\ldots\ldots a_{m-1})+(b_0,b_1,\ldots\ldots\ldots b_{m-1})=$

$(a_0+b_0.a_1+b_1.,\ldots\ldots\ldots a_{m-1+}\ b_{m-1})$

$\beta(a_0,a_1,\ldots\ldots\ldots a_{m-1})=(\beta.a_0.\beta.a_1\ldots\ldots\ldots\beta.a_{m-}$

----------(1)

Each m-tuple is called a point in EG(m,2^s) , with the all zero m-tuplr being called the origin. A line in 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 EG(m,2^s).

EG-LDPC codes are derived from these Euclidean Geometries. The EG-LDPC codes which are one step majority logic decodable are those derived from geometries EG(2,2^s) with block length $N=2^{2s}-1$. For these codes, the MLD check equations are taken from the incidence vectors of the lines which do not pass through the origin in EG(2,2^s) . Each of those vectors has 2^s points that correspond to the ones in the check equations. For each point there are 2^s such lines, and for each line there is an incidence vector [9]. But since each line passes through 2^s points, there are in reality only $2^{2s}-1$ (the number of points different than the origin) different lines (and incidence vectors) as the same line is counted in each of its points. Another useful property of the EG-LDPC codes is that every cyclic shift of an incidence vector is also an incidence vector [9].

Next we consider the number of bits that participate in an MLD equation at each iteration. Each equation has 2^sequations [9]. As every equation checks the final bit, this means that $2^{2s}-2^s+1$ bits are checked by the equations. This leaves 2^s-2 bits which are not checked by any equation at a given iteration. Those positions are precisely the ones which correspond to the line that passes through the origin [9] and are the form of k.(2^s+1) with k=1,2,….. 2^s-2.

This is in contrast to DS-LDPC codes where every bit is checked by all of the MLD check equations at every iteration.

A. Theoretical Proof for Single and Double Errors

Single errors will not be detected in the first iteration when they occur in one of the positions not checked by any equation. As these positions are spaced 2^s+1 apart, the error will be always detected in the second iteration.

For double errors, we begin the proof with the following lemmas.

Lemma 1: The MLD check equations in a one step MLD EG-LDPCcode 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), 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. This property is also found in DS-LDPC codes [9].In addition, to meet the conditions required for one step MLD (see Section II), the $2^{2s}-1$ remaining equations are obtained from the first equation by cyclically shifting the previous equation by the smallest amount such that the last bit is checked.

Lemma 2*:* There is no MLD check equation which has two ones at a distance k.(2^s+1) with k=1,2,….. 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 which would contradict one of the properties of the EG-LDPC codes (see previous subsection).

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 equation 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.

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 $k.(2^s+1)$. Therefore, the error will be detected in the second iteration unless there are two ones in the MLD vector at a distance $k.(2^s+1)$. 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 cannot 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.

REFERENCES

[1] R. C. Baumann, "Radiation-induced soft errors in advanced semicon- ductor 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 ap- proach 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 reli- ability 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

[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 frame- work 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.