International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 7, July - 2014

Efficient Majority Logic Fault Detection With Eg-Ldpc Codes for Memory Applications

Mrunali Namdeorao Ingole
Department of Electronics and Telecommunication Engineering
Patel Institute of Engineering and Science
Bhopal, India

Mrs. Sameena Zafar
H.O.D Department of Electronics and Telecommunication Engineering
Patel Institute of Engineering and Science
Bhopal, India

***Abstract*— now a day, a method was proposed to accelerate the majority logic decoding of difference set low density parity check codes. This can be implemented serially with simple hardware but require a large decoding time and also useful as majority logic decoding. This increases the memory access time for memory applications. The method detects a word has errors in 1st iterations of majority logic decoding, and if are no errors the decoding ends without completing the rest of the iterations. Since most of the words in a memory will be error-free, and 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 code (EG-LDPC) which are one step majority logic decidable. The result obtained show that this method is also effective for EG-LDPC codes. Extensive simulation results are given to accurately estimate the probability of error detection for different code size and numbers of errors.**

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

I INTRODUCTION

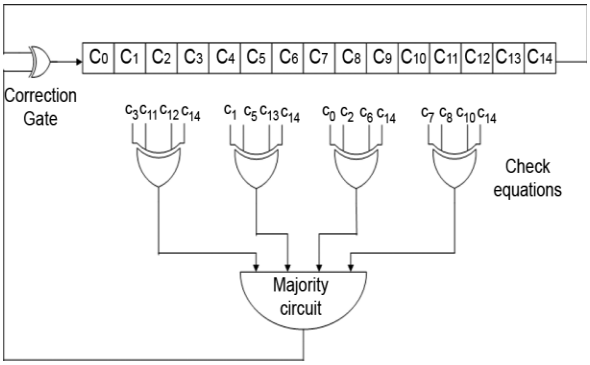
Error correction codes are usually used for protecting memories from *soft errors*, which change the logical value of memory cells without damaging the circuit[1]. A soft error occurs when a radiation event causes enough of a charge disturbance for reversing or flips the data state of a memory cell, register, latch, or flip-flop. The error is said to be "soft" because the circuit/device is not permanently damaged by the radiation, if new data are written in the bit, the device will store the data correctly. The soft error is also referred as a single event upset (SEU)[2] . Recently proposed a most useful advanced codes . With the use of these codes a large number of errors can be corrected, but generally it require a complex decoders. Decoding methods for the class of majority-logic decodable codes, and a class of codes that perform well with iterative decoding in spite of having many cycles of length 4 in their Tanner graphs, are presented[4]. One step majority logic decoding can be implemented with very simple circuitry, but require a large decoding times. In a memory, this increases the memory access time and this is an important system parameter.

By using one step majority logic decoding only a few classes of codes can be decoded. Out of these, some differential set low density parity check (DS-LDPC) codes, and Euclidean geometry low density parity check (EG-LDPC) code[1] LDPC code 1s a block code with parity-check matrices. It requires a very small number of non-zero entries. In which decoding complexity increases linearly with the code length and also increases a minimum distance linearly with code length. LDPC code is not differ from other block codes. The existing block codes can be successfully used with the LDPC iterative decoding algorithms when they are represented by a sparse parity-check matrix. Usually it is not practically find a sparse parity-check matrix for an existing code. LDPC codes are designed by constructing first with a sparse parity-check matrix and then determining a generator matrix for the code. The largest difference between LDPC codes and classical block codes is the decoding process. Classical block codes are usually decoded with ML like decoding algorithms and it is usually short and designed algebraically for making this task less complex. These codes are decoded iteratively with a graphical representation of their parity-check matrix[3]

II PROPOSED SYSTEM

Introduction

The proposed technique was implemented in VHDL and synthesized, showing that for codes with large block sizes the cost is low because the existing majority logic decoding



circuitry is reused for performing error detection and only some extra control logic is required. Codes with small words and affected by a small number of bit flips, it's practical to generate and test all possible error combinations. If the code size increases, the number of bit flips also increases, it is not necessary to test all possible combinations. This means that in some cases it may be more convenient to use an EG-LDPC code and keep a word size compatible with existing designs than the use of DS-LDPC code which requires a different word size or a shortened version of that code. When word size is used which is a power of two, there would be a bit is not used by the EG-LDPC code.

B] Efficiency of EG-LDPC

The comparison of the rate of the EG-LDPC code with other codes is more important to understand if the interesting properties of low-density and FSD-ECC come at the expense of lower code rates. the code rates of the EG-LDPC codes are compared with an achievable code rate upper bound (*Gilbert-Varshamov bound*) and a lower bound (*Hamming bound*). The EG-LDPC codes are not larger than the achievable Gilbert bound for the same and value and they are not much larger than the Hamming bound [5].

C] INTRODUCTION TO LDPC CODES

Let c is a LDPC code with length ɳ and dimension of the number of information bits ƙ . Let H an (m×n) matrix which denotes a parity check matrix of c. The Tanner graph of an LDPC code, Ç is a bipartite graph having two sets of nodes: variable (bit) nodes and check (constraint) nodes. The check nodes (variable nodes) connected with a variable node (check node) are said to be its neighbors. The degree of a node is the number of its neighbors. In a (γ,ρ) regular LDPC code, here each variable node has degree γ and each check node has degree ρ. The girth ᵍ is the length of the shortest cycle in Ç . A check node is referred as satisfied if the sum of all incoming messages has even parity and unsatisfied otherwise. The corresponding H matrix is given by

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

The Gallagher B algorithm on the BSC operates passing binary messages along with the edges of the Tanner graph of LDPC code. Every iteration of message starts with sending messages from variable nodes and ends by sending messages from check nodes to variable nodes. For a variable node ᵛ (check node c), let Ɛ(ᵛ) (Ɛ(c)) denote the edges incident on ᵛ(c) .

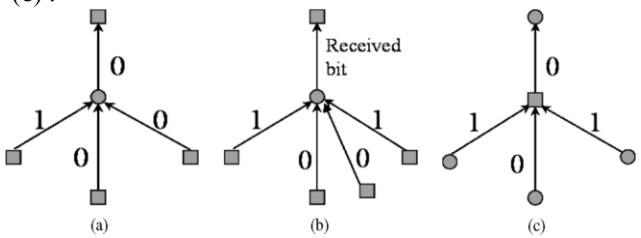


Fig. 2. Illustration of message passing. (a) Variable to check message. (b) Variable to check message in case of tie. (c) Check to variable message.

Also, let $_r(^v)$ be the received value of node ᵛ. Let the degree of a variable node denoted by j . A threshold is denoted by $b_{i,j}$ is determined for every iteration i and variable degree j. Let $m_i(e)$ be the messages passed on an edge e from variable node to check node and vice versa in iteration i respectively. Then for each and every node ᵛ , the Gallagher B algorithm passes the following messages in the iteration.[6]

$$\overrightarrow{m_i}(e) = r(v), \qquad i = 1.$$

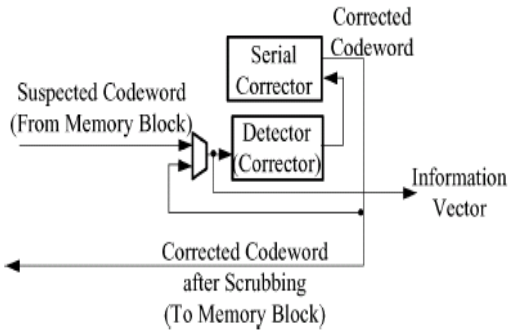
For $i > 1$

$$\overrightarrow{m_i}(e) = \begin{cases} 1, & \sum_{e' \in \widetilde{E_e(v)}} \overleftarrow{m_{i-1}}(e') \geq b_{i,j} \\ 0, & \sum_{e' \in \widetilde{E_e(v)}} \overleftarrow{m_{i-1}}(e') \leq j - 1 - b \\ r(v), & \text{otherwise} \end{cases}$$

where $\widetilde{E_e(v)} = E(v) \backslash \{e\}$.

$$\overleftarrow{m_i}(e) = \left(\sum_{e' \in E(c) \setminus \{e\}} \overrightarrow{m_i}(e') \right) \bmod 2.$$

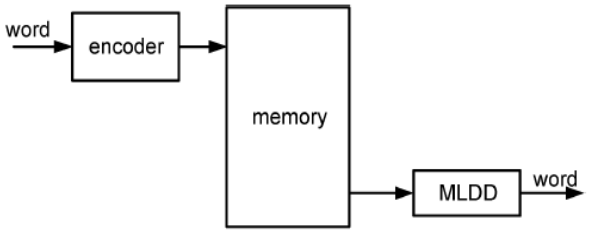
Fig. 2 illustrates the message passing rules for different cases.



D] Majority Logic Decoder/Detector

Results of these prove the hypothesis for the codes with small word size (N=15 and N=63). For N=255 up to three errors have been completely checked while for N=1023 only single and double error combinations have been completely checked. For complementing the results of the complete tests for large codes and number of errors, random error patterns have been used by using simulations. One billion error combinations are tested. For the number of errors, a similar reasoning applies, the large number of errors occur, the greater the probability, odd number of errors occurs in at least one equation. Finally it shows that the possibility of undetected errors are differ for an even and an odd number of errors as in the latter case, one of the errors occur in a bit that is not tested by any equation. The simulation results suggest that all errors corrupting three and four bits would be number of errors. The explanations of decreased word size as follows, the larger the word size, the large number of the number of MLD check equations in Table I and therefore it is more unlikely the errors occur in the same equation. For the number of errors, a similar reasoning applies the more number of errors, the larger the possibility that an odd number of errors occurs in at least one equation. Finally it must be show that the possibilities of undetected errors are differ for an even and an odd number of errors as in the latter case, any one of the errors must occur in a bit which is not tested by any equation. The simulation result shows that all errors corrupting three and four bits would be observed in the first three iterations. For errors affecting a large bits, there is less possibility of not detected in those iterations. For greater word sizes, the probabilities are sufficiently less which is acceptable in many applications [3].

In summary, the first three iterations will find all errors corrupting four or fewer bits, and almost other detectable error affecting greater bits. This is a slightly bad performance than in the case of DS-LDPC codes where errors affecting five bits were always found. The majority logic circuitry is very simple for EG-LDPC codes, as the number of equations is a power of two and an approach based on sorting networks proposed which can be used for reducing the cost of the majority logic decoding. EG-LDPC codes have block lengths closer to a power of two, thus matched well to the requirements of modern memory systems. This may means that in many cases it may be more convenient the use of EG-LDPC code and maintain a word size adjustable with existing designs i.e. power of two than the use of DS-LDPC code having a different word size or a short version of that code. By using word size with a power of two, there would be a bit which is not utilized by the EG-LDPC code. This bit can be utilized for a parity covering all bits in the word that would find all errors corrupting an odd number bit. In that case, the design using the EG-LDPC would also find all errors corrupting five or fewer bits.[3]



E] Need For Error Detection

Physical defects and environmental interference in the communication medium cause random bit errors during transmission of data. Error coding is one of the methods of detecting and correcting these errors for ensuring information is transferred from its source to its destination. In computer memory, error coding is used for fault tolerant computing, magnetic and optical data storage media, satellite and deep space communications, network communications, cellular telephone networks, and almost any other form of digital data communication. It also uses mathematical formulas to encode data bits at the source into larger bit words for transmission[3].

The “code word" can be decoded at the destination for retrieving the information. In the code word the extra bits give redundancy, according to the coding scheme used the destination uses the decoding process to determine if the communication medium introduced errors and in some cases correct them so that the data need not be transmitted again. Depending on the types of errors, different error coding schemes is chosen and whether or not again data transmission is possible. Good communications technology and faster processors makes more complex coding schemes, with greater error detecting and correcting capabilities, possible for small embedded systems, which allow for more robust communications. The tradeoffs in between coding and bandwidth overhead, coding complexity and allowable coding delay between transmissions, must be considered for each application.[3]

III.,PARITY CHECK CODE

For simplicity, we are discussing the even-parity checking, where the number of 1’s should be an even number. It’s also possible of using an odd-parity checking, where the number of 1’s should be odd.

One Dimensional Parity Check

The most common and less costly mechanism is the simple parity check for error- detection. A redundant bit is said to be parity bit in this technique is attached to every data unit so the number of 1's in the unit with the even parity. Blocks of data are subjected to a check bit or Parity bit in the form of generation from the source, where a parity of 1 is added to the block when it contains an odd number of 1's (ON bits) and 0 is added when it contains an even number of 1's. At the receiving end the parity bit is computed from the received data bits and these bits is compared with the received parity bit. This scheme makes the total number of 1's even, that's why it's called even parity checking.

Two Dimensional Parity Check

By using two-dimensional parity check, performance can be improved, which organizes the block of bits in the form of a table. There are calculated for each row, which is equal to a simple parity check bit. These bits are also calculated for all columns then both are sent along with the data. At the receiving end these are compared with the parity bits

RESULT

This method is applied to the class of one step MLD EG-LDPC codes. The conclusions are introduced first in terms of a hypothesis to present the results, and then it is validated by simulation and also a theoretical analysis is done. The results obtained can be summarized in the following assumption.

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

This assumption is differ from the DS-LDPCs codes in that case errors corrupted five bits always detected. This is cause due to structural differences between EG-LDPC and DS-LDPC codes, which will be detailed. By assuming the above assumption, the EG-LDPC codes is implemented and tested. For codes with small words and corrupted by a small number of bit flips, it is used to generate and test all error combinations. As the code size, the number of bit flips also increases. Therefore the simulations are done in two ways, by checking all error combinations if it is feasible and by checking randomly generated combinations in the rest of the cases. The results for the exhaustive checks. For N=255 up to three errors have been completely checked while for N=1023 only single and double error combinations have been checked. To complement the results of the exhaustive checks for greater codes and number of errors, simulations is done using random error patterns. One billion error combinations are checked. The results for errors corrupting more than four bits are shown in Table, since for errors up to corrupting four bits no undetected errors. It can be noted that for errors affect more than four bits, there are less number of error combinations that will not observed in the first three iterations. This number is reduced with word size and also with number of errors. The reduced word size can be explained as follows, the greater the word size, the greater the number of MLD check equations and therefore it is occur in the same equation. Finally it must be noted that the possibilities of undetected errors are differ for an even and an odd number of errors as in the last case, one of the errors must occur in a bit which is not tested by any other equation.

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	0	0	0
1023	0	1	0	0	0	0	0	0

The simulation results noted that all errors corrupting three and four bits would be obtained in the first three iterations. For errors corrupting a large number of bits, there is a small possibility of not found in those iterations. The possibilities are sufficiently small to be acceptable in many of the applications for large word sizes. The first three iterations will detect all errors corrupting four or fewer bits, and almost every other findable error corrupting more bits. The majority logic circuitry is simple for EG-LDPC codes. In addition, EG-LDPC codes have block lengths nearer to a power of two, thus matching well to the requirements of modern memory systems. This is means that, in some cases it may be more convenient to use an EG-LDPC code than using a DS-LDPC code having a different word size or a short version of that code and keep a word size compatible with existing designs i.e. power of two. By using word size which is a power of two, then a bit is not used by the EG-LDPC code. The design using the EG-LDPC code would also detect all errors corrupting five or fewer bits.

Table: Undetected errors with one billion random errors combinations

CONCLUSION

In this brief, the error detection during the first iterations of serial one step Majority Logic Decoding of EG-LDPC codes has been studied. The main objective was to reduce the decoding time by stopping the decoding process when there are no errors 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 error correction and capabilities word lengths. 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 error detecting capability and decoding time.

REFERENCES

[1] Pedro Reviriego, Juan A. Maestro, and Mark F. Flanagan, (Jan 2013)"Error Detection in Majority Logic Decoding of Euclidean Geometry Low DensityParity Check (EG-LDPC) Codes," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*. vol. 21, no. 1,

[2] Robert C. Baumann, Fellow, IEEE, (Sept 2005)" Radiation Induced Soft Error in Advanced Semiconductor Technologies," IEEE Trans. Device And Materials Reliability, vol. 5, no. 3,

[3] P. Kalai Mani, V. Vishnu Prasath, (March 2014) "Majority Logic Decoding of Euclidean Geometry Low Density Parity Check (EG-LDPC) Codes," International Journal of Innovative Research in Computer and Communication Engineering, vol. 2, special issue 1.

[4] Heng Tang, Member, IEEE, Jun Xu, Member, IEEE, Shu Lin, Life Yellow, IEEE< and Khaled A. S. Abdel-Ghaffar, Member, IEEE, (Feb 2005) "Codes on Finites Geometries," IEEE Trans. On Information Theory, Vol. 51, no. 2,

[5] Naeimi.H and A. DeHon, (Apr. 2009) "Fault secure encoder and decoder for nanomemory applications," *IEEE Trans. Very Large Scale Integr*.
(VLSI) Syst., vol. 17, no. 4,.

[6] Vasic.B and S. K. Chilappagari, (Nov. 2007) "An information theoretical framework for analysis and design of nanoscale fault-tolerant memories

I. BASED ON LOW-DENSITY PARITY-CHECK CODES," *IEEE TRANS*