RESEARCH ARTICLE OPEN ACCESS

Low Power Error Correcting Codes Using Majority Logic Decoding

A. Adline Priya., II Yr M. E (Communicasystems), Arunachala College Of Engg For Women, Manavilai, adline.priya@yahoo.com

Guided By: K.L. Nisha, Assist Prof (Ece) Arunachala College Of Engg For Women, Manavilai, nisha2787@gmail.com

Abstract
A low-density parity-check (LDPC) code is a linear error correcting code, and is used for transmitting message over a noisy transmission channel. A new class of error correcting codes called EG-LDPC (Euclidean Geometry-LDPC) codes and its encoder and decoder architectures for nanomemory applications is designed. EG-LDPC codes also have fault secure detection capability. One step majority logic decoding technique is used to detect errors in the memory. The method detects whether a word has errors in the first iterations, and when there are no errors the decoding ends without completing the rest of the iterations. This greatly reduces the average decoding time.

Index terms- Error correction codes (ECC), EG-LDPC codes, Majority logic decoding, Memory.

I. INTRODUCTION

LDPC codes and Turbo codes are among the known near Shannon limit codes that can achieve very low bit error rates for low SNR applications. When compared to the decoding algorithm of Turbo codes, LDPC decoding algorithm has more parallelization, low implementation complexity, low decoding latency, as well as no error-floors at high SNRs. For decoding of turbo codes LUT-Log BCJR algorithm is used and the decoding architecture contains 2^m ACS units which are placed in parallel, where m refers to the number of memory elements. And also the turbo decoder architecture contains memory units and also a number of registers [3]. The turbo decoder architecture has increased hardware complexity and power consumption. Also, in the last few years, the advances of low-density parity-check codes have seen them surpass turbo codes in terms of error floor and performance in the higher code rate range, leaving leaving turbo codes better suited for the lower code rates only. LDPC codes are considered for all the next generation communication standards.

Error correction codes are used to protect memories from soft errors, which change the logical value of memory cells without damaging the circuit. Due to the increase in soft error rate in logic circuits, the encoder and decoder circuitry around the memory blocks have become susceptible to soft errors and it should also be protected. Hence we design a new encoder and decoder circuit for memory designs. As technology improves, memory devices become larger and more powerful error correction codes are needed. The codes such as turbo codes and other error correcting codes can correct a large number of errors, but it generally requires larger complex decoders. This results in increased power consumption. To avoid a high decoding complexity, one step majority logic decodable codes are used for memory applications. Euclidean Geometry Low Density Parity Check (EG-LDPC) codes are one step majority logic decodable. The idea behind the method the method is that, the first iterations of the majority logic decoding is to detect if the word being decoded contains errors. If there are no errors, the decoding can be stopped without completing the rest of the iterations. Thus the decoding time is reduced. Majority logic decoding is used to detect and correct in memory applications.

Section I gives the introduction and Euclidean geometry codes are explained in section II. The design structure of the EG-LDPC codes are explained in section III. And then the results and conclusion are given in IV and V.

II. EUCLIDEAN GEOMETRY CODES

Euclidean geometry codes are low density parity check codes and it is hence called EG-LDPC codes. These codes are built using special structures of finite Euclidean Geometry. LDPC codes have limited number of 1's in the rows and columns of the matrix, and hence the complexity of the detectors and correctors is reduced. Euclidean Geometry is a finite geometry with **n** points and **J** lines. The properties of Euclidean Geometry [2] are given as follows:
1)every line consists of ρ points
2)any two points connected by exactly one line
3)every point is intersected by γ lines

4)two lines intersect exactly in one point or they are parallel

H is a Jxn parity check matrix, whose rows and columns corresponds to lines and points in an Euclidean Geometry. Every column of H matrix represents a point in the space, every row represents a line and every entry of 1 in the matrix represents that the corresponding row line is incident on the column point. The row weight ρ should be equal to the column weight γ. $h_{ij} = 1$ iff i^{th} line of EG contains j^{th} point of EG, and zero otherwise. **H** is an LDPC matrix and therefore the code is an LDPC code. Also, EG-LDPC codes are cyclic codes.

III. DESIGN STRUCTURE

All the existing designs of encoders and decoders use the conventional fault tolerant scheme to protect the encoder end decoder circuitry. The fault tolerant scheme includes logic replication or concurrent parity prediction. These schemes add additional logic to check the correctness of the circuit calculation. The overall architecture of ECC nanomemory is given in [4].

A)Encoder:

Let **i** be the k-bit information vector and **G** be the kxn generator matrix, then the received n-bit codeword **c** is given by,

$$c = i \times G$$

the generator matrix is given by, G = [I:X] where **i** is the k x k identity matrix and **X** be the k x (n-k) matrix that generates the parity bits. (n,k,d) represent an error correction code with code length **n**, information bit length **k** and minimum distance **d**. Minimum distance d refers to the minimum number of codebits that are different between any two codewords. The structure of the encoder circuit for (15,7,5) EG-LDPC code is given below as obtained from [2] in fig.1. i_0 to i_6 represents the 7-bit information vector. Each XOR gates generate one parity bit of the encoded vector. The encoder structure contains (n-k) XOR gates.

B)Detector:

Validity of received encoded vector is checked with the parity check matrix. The operation of the detector is to generate the syndrome vector. The checking or detecting operation is given by,

$$s = c \times H^T$$

the syndrome vector **s** is an (n-k) bit vector. Each bit of the syndrome vector is the product of c with one row of H. If the syndrome s is '0' the c is a valid codeword and if not equal to '0' the c is erroneous. The binary sum of this product is implemented with an XOR gate. Since row weight of H is ρ, to generate 1 digit of the syndrome vector, we need a ρ input XOR gate or (ρ-1) two input XOR gate. The whole detector takes n(ρ-1) two input XOR gates. An error is detected if any syndrome bits has a non-zero value. OR function of all syndrome bits are used to detect errors. (n-input XOR gate used). The detector for an (15,7,5) EG-LDPC code is given in fig. 2 as obtained from [2].

C)Memory:

In memory units with ECC scrubbing logic is used to maintain integrity. To avoid accumulation of too many errors in memory, scrubbing is performed. Scrubbing refers to the process of periodically reading memory words from the memory, correct the potential errors and write them back into the memory. While performing the operation of memory scrubbing,

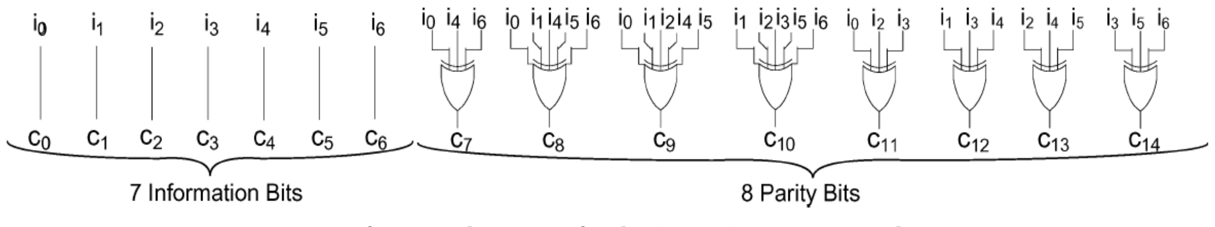

i_0 i_1 i_2 i_3 i_4 i_5 i_6
i_0 i_4 i_6
i_0 i_1 i_4 i_5 i_6
i_0 i_1 i_2 i_4 i_5
i_1 i_2 i_3 i_5 i_6
i_0 i_2 i_3
i_1 i_3 i_4
i_2 i_4 i_5
i_3 i_5 i_6
c_0 c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} c_{11} c_{12} c_{13} c_{14}
7 Information Bits
8 Parity Bits

Fig.1. structure of an encoder circuit for the (15,7,5) EG-LDPC code

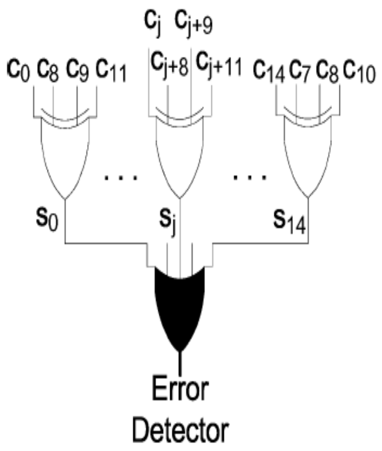


Fig.2. Structure of an detector circuit for the (15,7,5) EG-LDPC code.

normal memory access operation is stopped.

D)One step majority logic decoding:

One step majority logic correction is a fast and relatively compact error correcting technique. Type 1 2D EG-LDPC codes are one step majority logic decodable. Type 1 EG-LDPC codes are systematic. One step majority logic corrector is used to identify the correct value of each bit in the codeword directly from the received codeword. Decoding of EG-LDPC codes have low computational overhead and is sparseness. Hence it can be easily implemented using nanoscale hardware [4]. The conventional technique includes message passing error correction strategy. This technique demands multiple iterations or error diagnosis and trial correction.

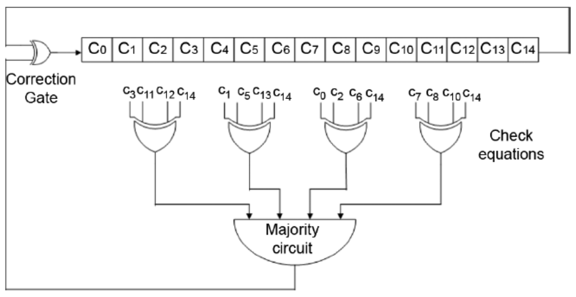
The one step majority logic corrector when implemented serially provides compact implementation and when implemented in parallel minimize correction latency. The advantage of this method is that it requires very little additional circuitry as the decoding circuitry is also used for error detection. This method contains two main parts.

1) generating a specific set of linear sums of the received vector bits.
2) finding majority value of the computed linear sums.

A linear sum of the received encoded vector bits can be formed by computing the inner product of the received vector and a row of a parity-check matrix. This sum is called Parity-Check sum. The core of the one-step majority-logic corrector is generating γ parity-check sums from the appropriate rows of the parity-check matrix. The one-step majority logic error correction is summarized in the following procedure. These steps correct a potential error in one code bit lets say, c_{n-1}

1) Generate γ parity-check sums by computing the inner product of the received vector and the appropriate rows of parity-check matrix.
2) The γ check sums are fed into a majority gate. The output of the majority gate corrects the bit c_{n-1} by inverting the value c_{n-1} of if the output of majority gate is “1”.

The circuit implementing a serial one-step majority logic corrector for a (15,7,5) EG-LDPC code is shown in Fig. 3. This circuit generates γ parity-check sums with γ XOR gates and then computes the majority value of the parity-check sums. Since each parity-check sum is computed using a row of the parity check matrix and the row density of EG-LDPC codes are ρ , each XOR gate that computes the linear sum has ρ inputs.


C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8 C_9 C_{10} C_{11} C_{12} C_{13} C_{14}
Correction Gate
c_3 c_{11} c_{12} c_{14}
c_1 c_5 c_{13} c_{14}
c_0 c_2 c_6 c_{14}
c_7 c_8 c_{10} c_{14}
Check equations
Majority circuit

If errors can be detected in the first few iterations of majority logic decoding , 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.

IV. RESULTS

Fig.3. serial one-step majority logic decoder for the (15,7,5) EG-LDPC code

The parameters of the EG-LDPC codes are given such as: N is the block size, K the number of information bits, J the number of majority logic decoding check equations and t_{ML} the number of errors that the code can correct using one step majority logic decoding. In fig.3. the block size N = 15. First the entire data block is loaded into the registers. Then the check equations are computed and the majority value is calculated. The majority value indicates the correctness of the code-bit under consideration. If majority value is 1, the bit c_{n-1} is inverted, otherwise it is kept unchanged. Once the code bit c_{n-1} is corrected the codeword is cyclic shifted and code bit c_{n-2} is placed at position c_{n-1} and will be corrected. Thus all the bits are cyclically

shifted. This set of operations constitute a single iteration: the bits are in the same position after N iterations, in which they are loaded. Thus, each bit may be corrected only once. The properties of check equations are given as follows:
1)All check equations include the variable whose value is stored in the last register(i.e c_{14})
2)The rest of the registers are included in at most one of the check equations.

The simulation results of encoder, decoder, detector and memory is given below

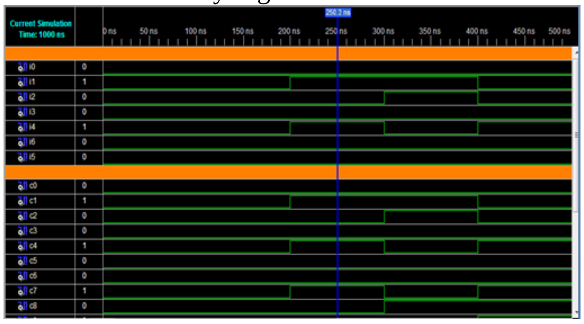


Fig.4. simulation of encoder

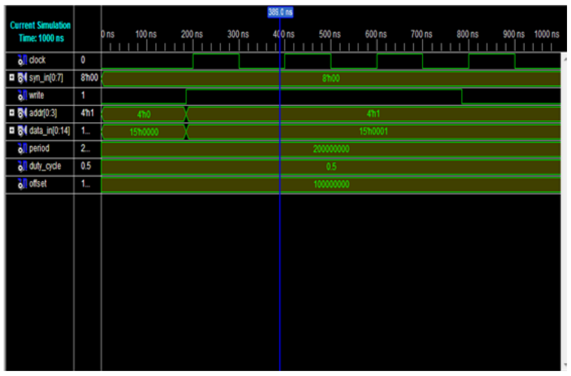


Fig.5. simulation of memory

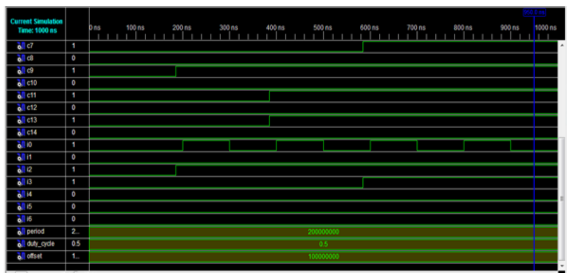


Fig.6. simulation of decoder

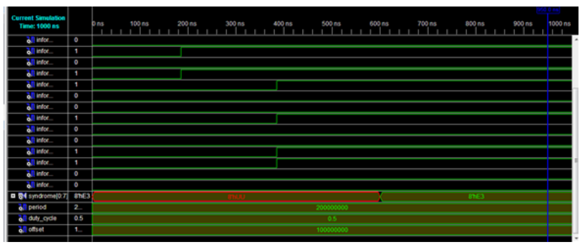


Fig.7. simulation of detector

Parameters	**LDPC code (Proposed)**	**Turbo code [3]**
Decoding Algorithm	Majority logic decoding	LUT-Log-BCJR
Delay(ns)	4.910	30.413
Power(W)	0.034	0.042
Area(mm^2)	0.15	0.35

An hypothesis was given in [1] and is given as "given a word read from memory protected with one step majority logic decoding EG-LDPC codes, and affected by up to four bit-flips, all errors can be detected in only three decoding cycles. Also the majority logic circuitry is simpler for EG-LDPC codes, as the number of equations is a power of two. The majority gate has an application in other error correcting codes, and this compact implementation can improve many other applications. A majority function of γ binary digits is simply the median of the digits (where we define the median of an even number of digits as the $\frac{\gamma}{2}$ + 1st smallest digit. The majority logic decoding technique was implemented in VHDL and synthesized, and the results show that for codes with large block sizes the overhead is low. The comparison between LDPC and Turbo codes is given in terms of area, power and delay in the table. 1. Thus decoding of LDPC codes are less complex than turbo codes. And it also reduces the hardware complexity and power consumption.

Table. 1. Comparison of LDPC and Turbo codes

IV. CONCLUSION

Thus the decoder architecture for LDPC codes are designed. And the simulation results for encoder, decoder, memory and detector are obtained. And also the majority logic decoder is implemented serially. And the future work is to implement the majority logic decoder in parallel.

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" IEEE Transactions On Very Large Scale Integration Systems, Vol. 21, No. 1, January 2013

[2] Helia Naeimi and André DeHon," Fault Secure Encoder and Decoder for NanoMemory Applications" IEEE Transactions On Very Large Scale Integration Systems, Vol. 17, No. 4, April 2009

[3] Liang Li, Robert G. Maunder, Bashir M. Al-Hashimi, and Lajos Hanzo," A Low-Complexity Turbo Decoder Architecture for Energy-Efficient Wireless Sensor Networks" IEEE Transactions On Very Large Scale Integration Systems, Vol. 21, No. 1, January 2013

[4] Shalini Ghosh and Patrick D. Lincoln," Low Density Parity Check Codes for Error Correction in Nanoscale Memory" September 25, 2007

[5] 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

.[6] 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

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

.[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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. (V*LSI) Syst.*, vol. 20, no. 1, pp. 148–156, Jan. 2012.

[15] 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.