ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group lf Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

Reduction of Decoding Time in Majority Logic Decoder for Memory Applications

K.Jayalakshmi[1], B.Sivasankari[2]

PG-Scholar, Electronics and communication, SNS College of technology, Coimbatore, India[1]

Assistant Professor, Electronics and communication, SNS College of technology, Coimbatore, India[2]

ABSTRACT: An error detection method for EG-LDPC codes with majority logic decoding is presented in this paper. Majority logic decodable codes are suitable for memory applications due to their capability to correct a large number of errors. They are simple to implement and have modular encoder and decoder. However, they require a large decoding time that impacts memory applications. Mostly memory applications require low latency encoders and decoders. So the proposed method has reduced a decoding time by detecting whether a word has errors in the first iteration of majority logic decoding and when there are no errors in the decoding ends without completing the rest of the iterations. Since most words in a memory will be error free, so the average decoding time is greatly reduced compared to existing methods

I. INTRODUCTION

The continuous impact of technology scaling has been affecting the reliability of memory applications not only in extreme radiation environments like spacecraft and avionics electronics, but also at normal terrestrial environments. Especially, SRAM memory failure rates are increasing significantly, therefore posing a major reliability concern for many applications[1]. Memory cells are mainly susceptible from soft or transient errors.

For reliable performance those faults can be detected using Error Correction Codes (ECC). When digital data is stored in a memory, it is crucial to have a mechanism that can detect and correct a certain number of errors. ECC encodes data in such a way that a decoder can identify and correct certain errors in the data. Usually data strings are encoded by adding a number of redundant bits to them. When the original data is reconstructed a decoder examines the encoded message, to check for any errors[8].

1.1 Error Correction Codes (ECC)

There are 2 basic types of Error Correction Codes:

1.Block codes,: This codes are referred to as (n,k) codes.

2.Convolution codes: The code words produced depend on both the data message and a given number of previously encoded messages.

Among the ECC codes that meet the requirements of higher error correction capability and low decoding complexity, cyclic block codes have been identified as good one, due to their property of being Majority Logic (ML) de codable. LDPC (Low Density Parity Check) codes belong to the family of ML de codable codes. Euclidean Geometry Low Density Parity check Codes (EG-LDPC) are one step ML de codable codes with high error correction capability and are linear cyclic block codes[2].

1.2 EG-LDPC:

Euclidean Geometry codes are also called EG-LDPC codes based on the fact that they are low-density parity-check (LDPC) codes.LDPC codes have a limited number of 1's in each row and column of the matrix; this limit guarantees limited complexity in their associated detectors and correctors making them fast and light weight[3].

EG-LDPC has the following parameters for any positive integer $t \geq 2$

- Information bits, $k=2^{2t} - 3t$

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group lf Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

- Length, $n = 2^{2t} - 1$
- Minimum distance, $d_{min} = 2t + 1$
- Dimensions of the parity-check matrix, n x n
- Row weight of the parity-check matrix, $p = 2t$
- Column weight of the parity-check matrix, $y = 2t$

II. CODING SCHEME

The ECC codes construction starts with its parity check matrix and generator matrix which are the cyclic shifts of their first rows. The checking or detecting operation is basically summarized as the following vector-matrix multiplication:

$$s = c.\, H^T$$

Where, H is an $(n-k) \times n$ Parity-Check matrix. $(n - k)$-bit vector s is called syndrome vector. A syndrome vector is zero if c is a valid codeword and non-zero if c is an erroneous codeword.

2.1 Creating a parity check matrix:

The parity check matrix for a given code can be derived from its generator matrix . If the generator matrix for an [n, k]-code is in standard form:

$$G = [I_k \mid P]$$

Then the parity check matrix is given by

$$H = [-P^T \mid I_{n-k}]$$

Where I is an identity matrix,P-Parity matrix that generates parity bits.Because, $GH^T = P - P = 0$.

2.2 Design structure of an EG- LDPC Encoder:

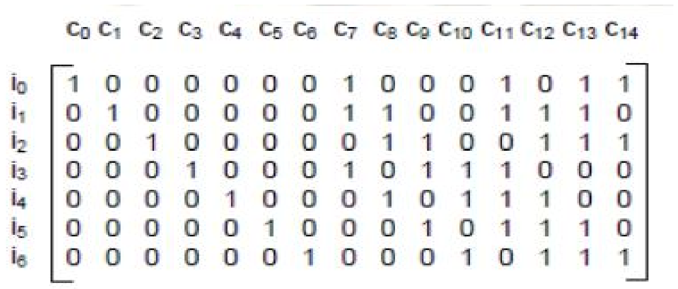
Let $i = (i_0, i_1, i_2, \ldots, i_{k-1})$be the k-bit information vector that will be encoded into an n-bit codeword $c = (c_0, c_1, c_2, \ldots, c_{n-1})$

For linear codes, the encoding operation essentially performs the following vector-matrix multiplication:

$$c_{k \times n} = [\, i \,]_{k \times k} \,.\, [\, G \,]_{k \times n}$$

where G is a generator matrix

Consider the encoder circuit to compute the parity bits of the (15, 7, 5) EG-LDPC code is shown in figure.1 and generator matrix also shown in figure.2



$$\begin{array}{c|ccccccccccccccc}
 & c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} \\
\hline
i_0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
i_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
i_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
i_3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
i_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
i_5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
i_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1
\end{array}$$

Fig.1 Generator matrix for(15,7)EG-LDPC code in systematic format

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group lf Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

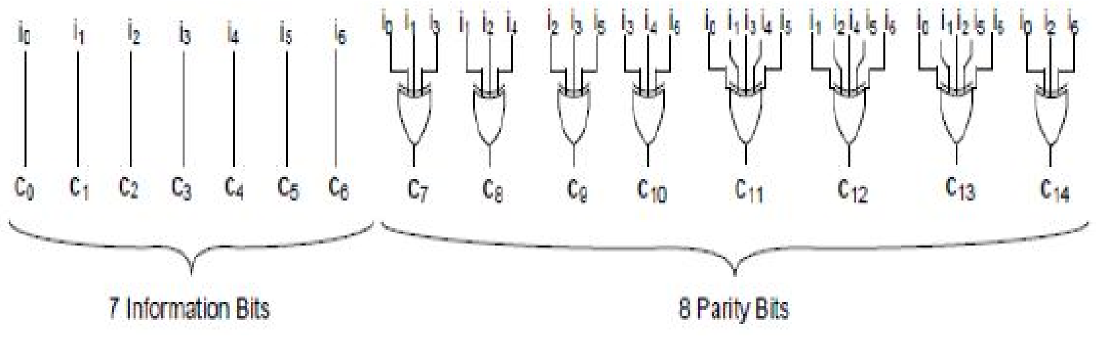


Fig.2 Structure of an encoder circuit for the (15, 7, 5) EG-LDPC code

The information vectors are (i_0 to i_6) copied into (c_0 to c_6) and (c_7 to c_{14}) are the linear sums (XOR) of the information bits.

III. EXISTING MAJORITY LOGIC DECODER (MLD)

MLD is based on a number of parity check equations which are orthogonal to each other.so that, at each iteration, each code word bit only participates in one parity check equation, except the very first bit which contributes to all equations. For this reason, the majority result of these parity check equations decide the correctness of the current bit under decoding. ML decoder is a simple and powerful decoder, capable of correcting multiple random bit-flips depending on the number of parity check equations.

3.1 Plain ML Decoder

As described before, the ML decoder is a simple and powerful decoder, capable of correcting multiple random bit-flips depending on the number of parity check equations. It consists of four parts: 1) a cyclic shift register; 2) an XOR matrix; 3) a majority gate; and 4) an XOR for correcting the codeword bit under decoding. The input signal is initially stored into the cyclic shift register and shifted through all the taps. The intermediate values in each tap are then used to calculate the results {Bj} of the check sum equations from the XOR matrix. In the Nth cycle, the result has reached the final tap, producing the output signal (which is the decoded version of input) [1].

3.2 Plain MLD With Syndrome Fault Detector (SFD)

In order to improve the decoder performance, alternative designs may be used. One possibility is to add a fault detector by calculating the syndrome, so that only faulty code words are decoded [4]. Since most of the code words will be error-free, no further correction will be needed, and therefore performance will not be affected. Although the implementation of an SFD reduces the average latency of the decoding process, it also adds complexity to the design.

IV. PROPOSED ML DETECTOR / DECODER

In the proposed Majority Logic Detector/Decoder (MLDD), the data words are encoded using EG-LDPC and then stored in the memory. When the memory is read, the code word is then fed through the ML detector/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 suffered while being stored in the memory.

4.1. Hypothesis:

The proposed technique is based on the following hypothesis: “Given a word read from a memory protected with EG-LDPC codes, and affected by up to four bit-flips, all errors can be detected in only three decoding cycles.”[2]

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group If Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

4.2.MLDD design:

The figure 3 shows the proposed ML detector/decoder with 15-tap shift register and an XOR array to calculate the orthogonal parity check sums.A majority gate for deciding if the current bit under decoding needs to be inverted. The control unit triggers a finish flag when no errors are detected after the third cycle. The output tristate buffers are always in high impedance unless the control unit sends the finish signal so that the current values of the shift register are forwarded to the output y.

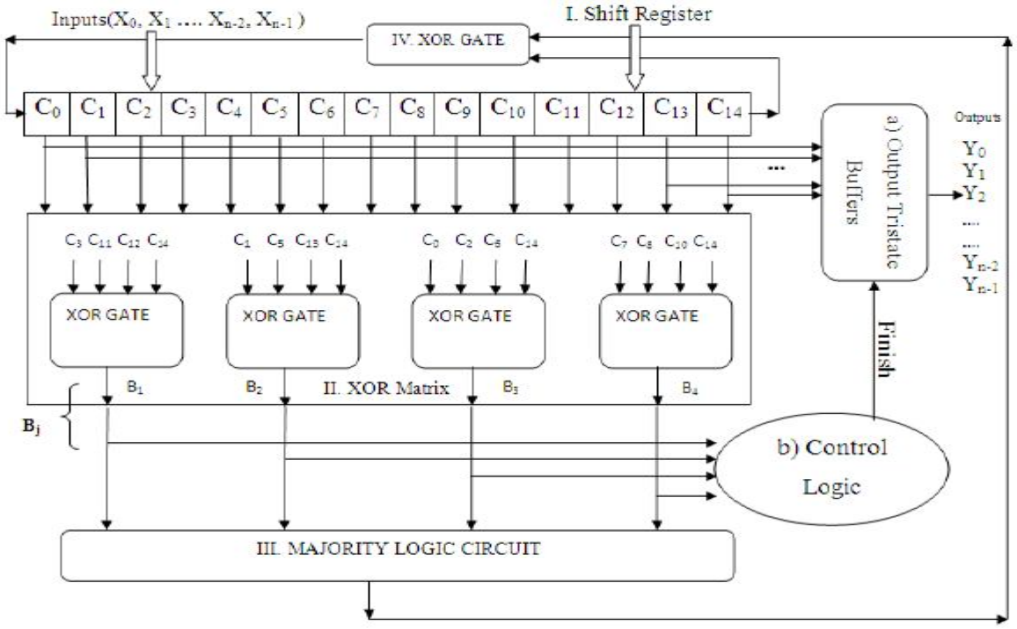

Inputs(X_0, X_1 .... X_{n-2}, X_{n-1})
IV. XOR GATE
I. Shift Register
C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8 C_9 C_{10} C_{11} C_{12} C_{13} C_{14}
a) Output Tristate Buffers
Outputs
Y_0 Y_1 Y_2 .... Y_{n-2} Y_{n-1}
C_3 C_{11} C_{12} C_{14}
C_1 C_5 C_{13} C_{14}
C_0 C_2 C_6 C_{14}
C_7 C_8 C_{10} C_{14}
XOR GATE
B_1 B_2 B_3 B_4
II. XOR Matrix
B_j
Finish
b) Control Logic
III. MAJORITY LOGIC CIRCUIT

Fig:3.Schematic of the proposed MLDD for 15 bit code word. a) Control logic b) Output tristate buffers

The Proposed MLDD algorithm is illustrated in Fig.4.Proposed MLDD algorithm requires additional logic compared to the MLD algorithm. The corrections are performed during the first N iterations If the majority gate detect any error in codeword the iterations take place depends upon the length of the codeword's.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group lf Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

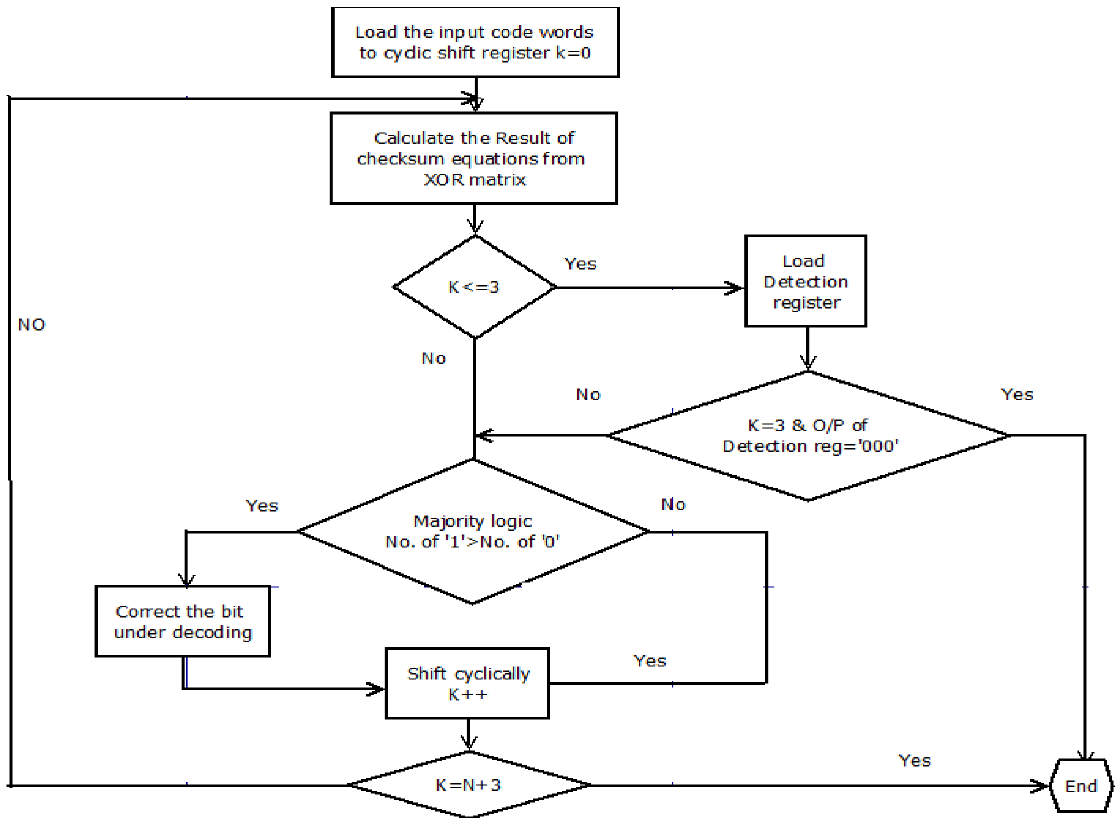

Load the input code words to cyclic shift register k=0
Calculate the Result of checksum equations from XOR matrix
K<=3
Yes
Load Detection register
No
NO
No
K=3 & O/P of Detection reg='000'
Yes
Yes
Majority logic No. of '1'>No. of '0'
No
Correct the bit under decoding
Shift cyclically K++
Yes
K=N+3
Yes
End

Figure.4 Flow diagram of the MLDD algorithm

An additional logic unit of proposed MLDD is control and tristate buffer logic which shown in figure.5The control unit manages the detection process. It uses a counter that counts up to three, which distinguishes the first three iterations of the ML decoding.

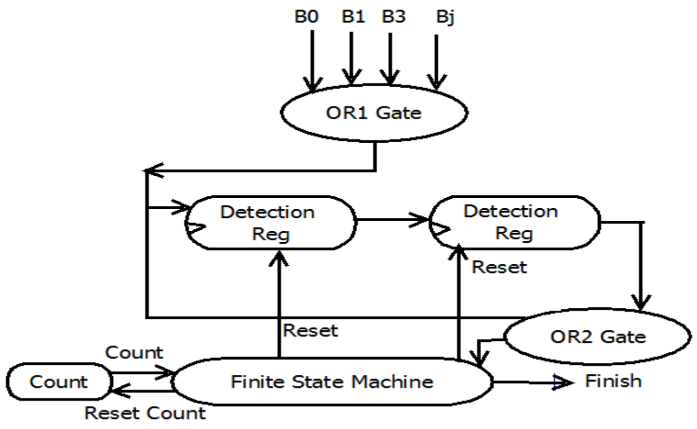


Figure.5 Schematic of the control unit

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

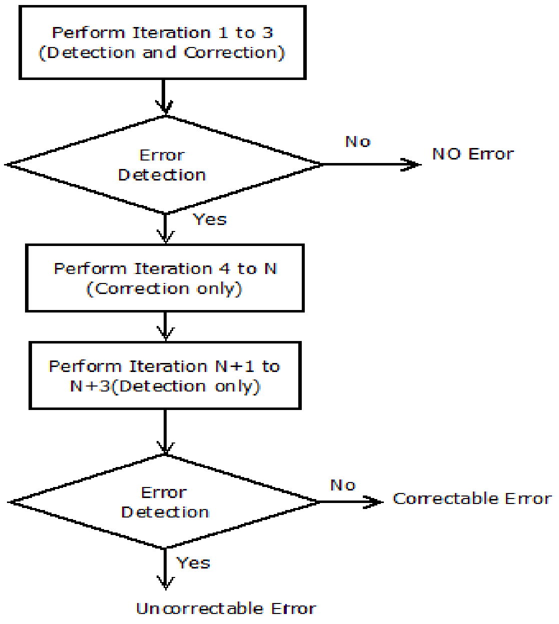
(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group If Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

Data bits stay in memory for a number of cycles and, during this period, each memory bit can be upset by a transient fault with certain probability. Therefore, transient errors accumulate in the memory words over time. In order to avoid accumulation of too many errors in any memory word that surpasses the code correction capability, the system must perform memory scrubbing. Memory scrubbing is the process of periodically reading memory words from the memory, correcting any potential errors, and writing them back into the memory.


Perform Iteration 1 to 3 (Detection and Correction)
Error Detection
No
NO Error
Yes
Perform Iteration 4 to N (Correction only)
Perform Iteration N+1 to N+3(Detection only)
Error Detection
No
Correctable Error
Yes
Uncorrectable Error

Figurre.6 Flow chart of MLDD method

The flow chart for MLDD method as shown in Fig.6. Iteration from 1 to 3 will be performed for detection of the error in the codeword. If the codeword is error free then output will be obtained in three iteration. Otherwise iteration is continued to detect and correct the error in the codeword. Iteration from 4 to N will be preformed to correct the error in the codeword. Performances N+1 to N+3 iteration are to detect the error in codeword. If the error is detected then it will corrected. Otherwise retransmission will be processed.

Since we are using a separate module for fault detection, there will be a slight area overhead. This area overhead can be overcome by using sorting network in the majority gate. The EG LDPC code used here is only for 15 bits, it have only outputs four outputs from xor matrix. It takes only four input bits and the vertical lines shown here is comparator as shown in Figure 7 (b) which has two inputs and it will compare and larger bit is given to the top output and smaller to bottom respectively [3].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group If Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

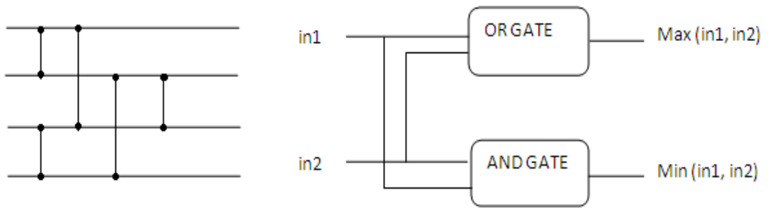


Figure 7. (a)Sorting network-four input (b) Schematic of one comparator

V. CONCLUSION

For the plain MLD, the memory read access delay is directly dependent on the code size(in this case a code with length 15 needs 15 cycles etc.) Then, for I/O two extra cycles needed. On the other hand, for proposed MLDD the memory read access delay is only dependent on the word error rate (WER). The proposed design just requires three cycles to detect any error (plus two of I/O). If an error is detected, all of the techniques need to run the whole decoding process.The proposed MLDD also have same procedure, but instead of N+2 cycles, three extra cycles are needed (for a total of N+2) is summerised in table I.

Table I Performance Of The Existing And Proposed Design

Technique	I/O	Error Detection	Without Error	With Error
Plain MLD	2	N	N+2	N+2
Proposed MLDD	2	3	3+2=5	N+5 (eg. N=15+5)

Table II Analysis of Memory Read Access Delay

Technique	Delay(ns)
Plain MLD	5.609
Proposed MLDD	4.567

Table III Comparision Of Power Utilization

Technique	Power(mW)
Plain MLD	26.33
Proposed MLDD	26.13

The Proposed MLDD have less delay when compared to existing MLD, since the proposed dessign detects the faults in just three cycles. The delay comparison is shown in table II.When the code word does not suffer from errors, it can come out in the next 4th cycle itself without further shifting. Therefore this is a great advantage for MLDD in terms of delay and performance.

VI. FUTURE WORK

The further scope is to eliminate the silent error corruption. If the input has more than four bit error in the codeword, then the MLDD process is not exactly suitable to correct the codeword. In such case, silent fault corruption may occur. To reduce such fault, one more detection logic can be implemented after the completion of 15 iteration. The applications of the proposed techniques to memories that use scrubbing is also an interesting topic and was in fact the original motivation that led to the Parallel MLDD scheme.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group lf Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

REFERENCES

[1]Shih-Fu Liu, Pedro Reviriego, and Juan Antonio Maestro,"Efficient majority logic fault detection with difference-set codes for memory applications" , IEEE Transactions on Very Large Scale Integration (VLSI) systems, vol. 20, no. 1, January 2012.
[2]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 (VLSI) systems, vol. 21, no. 1, January 2013.
[3] 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, Apr. 2009.
[4]R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," IEEE Trans. Device Mater. Reliabil., vol. 5, no.3, Sep. 2005.
[5]R.C.Baumann, "Radiation-Induced Soft Errors in Advanced Semiconductor Technologies,"IEEE Transactions On Device Material Reliability,2005
[6]Charles W. Slay man, "Cache and Memory Error Detection, Correction, and Reduction Techniques for Terrestrial Servers and Workstations,"IEEE Transactions On Device Material Reliability,2005
[7]S. Lin and D. J. Costello, "Error Control Coding" ,2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004
[8]H. Tang, J. Xu, S. Lin, and K. A. S. Abdel-Ghaffar, "Codes on finite geometries," IEEE Trans. Inf. Theory, vol. 51, no. 2, 2005.
[9]P. Ankolekar, S. Rosner, R. Isaac, and J. W. Bredow, "Multi-bit error correction methods for latency-constrained flash memory systems," IEEE Trans. Device Mater. Rel., vol. 10, no. 1, Mar. 2010.
[10]J.A.Maestro and P>Reviriego, "Reliability of soft-error correction protected memories:,IEEE Trans,on Reliability,vol.58,Mar.2009