International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.3, Issue No.3, pp : 276-279 1 March 2014

Error Detection and Correction by using Different Set Codes for Memory Applications

Kanagapriya.V[1], Kavitha.S [2] , KarthiKeyan.S [3], Sanjay.K [4] .
Department of Electronics and Communication Engineering,
Angel College of Engineering and Technology, Tirupur – 641665
[1]priyaece13@gmail.com,[2]vskarthi18@gmail.com,[3]skavi463@gmail.com, [4]t.k.sanjay@gmail.com

ABSTRACT:
Error correction codes are commonly used to protect memories from soft errors, which change the logical value of memory cells without damaging the circuit. As technology scales, memory devices become larger and more powerful error correction codes are needed. Majority logic decoding is efficient technique due to their capability to correct multi number of errors, less time and area consumption. The proposed error-detection and correction is achieved by using 2D parity check, checksum, CRC methods with majority logic decoding. This proposed method is more efficient than the existing method as it reduces the memory access time when there is no error in the data read. This proposed technique is implemented by using Xilinx design suite 12.1
Key terms: low density parity check, different set codes, error correction codes, Majority Logic Decoding

I.INTRODUCTION:

The impact of technology scaling small in dimensions, high combination densities, and low operating voltages which has come to a level that reliable of memories is put to difficulty. SRAM memory failure rates are increasing drastically, so posing a foremost reliable concern for many applications. For memories, it turned out that ECC codes are best way to mitigate memory soft errors. For terrestrial radiation environments where there is a low soft error rate (SER), codes like single error correction and double error detection (SEC–DED) are a good result, due to their small encoding and decoding complexity. However, as a consequence of augmenting integration densities, there is an increase in the number of soft errors and which produce the require for high error correction.

To avoid a high decoding complexity, the use of one step majority logic decodable codes was first proposed for memory applications. Further work One step majority logic decoding can be implemented serially with very simple circuitry 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 . capabilities. A sub-group of the low-density parity check (LDPC) codes, which be-longs to the family of the ML decodable codes, has been researched.

In this paper, we will focus on one specific type of LDPC codes, namely the difference-set cyclic codes (DSCCs), which is widely used in the Japanese tele text system or FM multiplex broadcasting systems. The main reason for using ML decoding is that it is very simple to implement and thus it is very practical and has low difficulty. The disadvantage of ML decoding is for a coded word of -bits, it takes cycles in the decoding progression, posing a large impact on system output. One way of coping with this problem is to implement parallel encoders and the decoders. This result would especially increases the complexity and the power consumption. The most of the memory read access will without errors, and the decoder is majority of the time functioning without reason.

This will have motivate the uses of a error detector module that checks if the codeword contains an error and then triggers the correction mechanism accordingly. In this case, only the faulty code word needs correction, and as a result the average reading memory accessing is speed up, by the expense of increased in hardware cost and the power utilization. For a code with block length N, majority logic decoding requires N iterations, so that as the code size grows, so does the decoding time. The remaining of this paper is organized as follows. Section I-IV gives an existing ML decoding. Section V and VI presents Proposed two dimensional parity check, CRC method with MLDD .

II. SUMMARY ABOUT MAJORITY LOGIC DECODING (MLD) SOLUTION

MLD is based on a number of parity check equations which are orthogonal to each other, so that, at each iteration, each codeword bit only participates in one parity checks equation, apart from the very first bit which contribute to every equations..

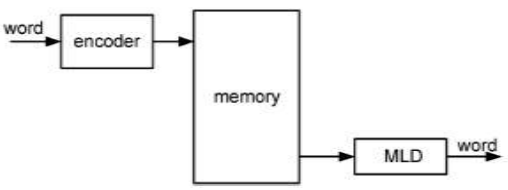

Fig. 1. Block Diagram of the MLD System

MLD was first mentioned for the Reed–Muller codes. Then, it was extended for all types of systematic linear block codes that can be totally orthogonalized on each codeword bit. A generic schematic of a memory system is depicted into Fig.1, as using of an ML decoder. primarily, the data words are encoded and then stored within the memory. After the memory is reads, the code words are fed through the ML decoder before sent to the output for additional processing. During this decoding processing the data words are corrected from all bit-flips that it might have suffered while being stored in the memory. where the code word is loaded into a cyclic shift register and the decoding started through calculates the parity check equations hard wired in the XOR matrix.

The resulting sums are then forwarded to the majority gate for evaluating its correctness .If the number of 1's received in is greater than the number of 0's, which would mean that the current bit under decode value is wrong and the signal to correct which would have been triggered. If not, the bits under decoding will be correct and no extra operations will needs on it. Then the next step the contents of the register is rotated and the above procedure is repeated until all codeword bits have processed.. This is a big impact on the performance of system, and depending on the size of the codeword. The example of a code words of 15 bits, then the decoding would takes 15 cycles, as would be extreme for most applications.

III.MLD WITH SFD METHOD

In order to improve the decoder performance, alternative de-signs may be used. One possibility is to add a fault detector by calculating the syndrome, so that only faulty codeword's are decoded. Since most of the codeword's will be error-free, no further correction will be needed, 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 (as shown in Fig. 2). The SFD is an XOR matrix that calculates the syndrome based on the parity check matrix. Each parity bit results in a syndrome equation. Therefore, the complexity of the syndrome calculator increases with the size of the code.

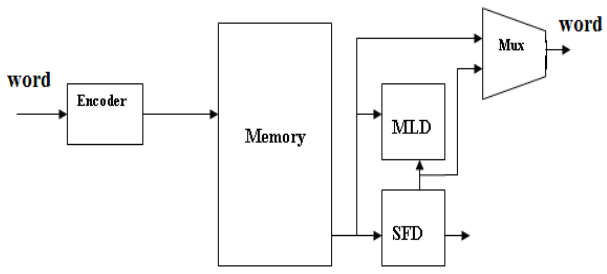

Fig 2 Schematic of the MLD with SFD

IV.EXISTENT ONE DIMENSIONAL PARITY CHECK

The parity check is the simplest and most popular error detection scheme. It appends the parity bit to the end of the data. Suppose that the information to be sent, *d* bits. In an even parity scheme, the sender simply includes one additional bit and chooses its value such that the total number of 1's in the (*d+1* bits) is even. The receiver need only count the number of 1's in the received *d+1* bits. If an odd number of 1-valued bits are found with an even parity scheme, the receiver knows that at least one bit error has occurred. Simple parity check can detect all single-bit errors.

It can also detect burst errors as long as the total number of bits changed is odd. This method cannot detect errors where the total number of bits changed is even. If any two bits change in transmission, the changes cancel each other and the data unit will pass a parity check even though the data unit is damaged.

V.PROPOSED TWO DIMENSIONAL PARITY CHECK

Figure 3shows a two-dimensional generalization of the single-bit parity scheme. Here, the *d* bits in *D* are divided into *i* rows and *j* columns. A parity value is computed for each row and for each column. both are sending along with the data. At the receiving end these are compared with parity bits calculated on the received data.

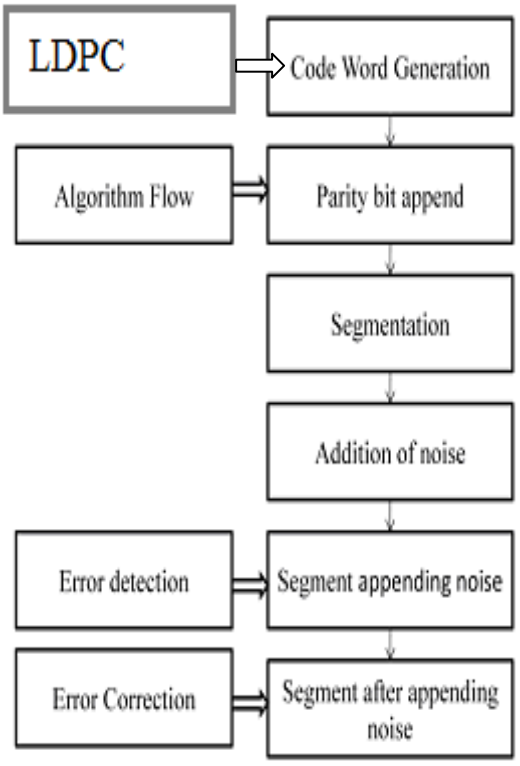

Fig.3Flowchart of the DC 2D parity flow

To generate a code word
c =m · G =m0g0 + m1g0 +……+ m**k-1gk-1**
G=P|Ik
H = [**I**n-k | **P**^T]
G-generator matrix
H-parity check matrix
Initialize
Data
C= m · G =m0g0 + m1g0 +……+ mk-1gk-1
In=Data&C
Segmentation.parity
If(seg.parity=1)
Tr.seg=In&1
Else In&0
Segment generation
End if
If(Rr=Tr)
Finish
Else
Allow ML decoding (repeat)to get(Rr=Tr)
end

Two-dimensional parity check increases the likelihood of detecting burst error.a redundancy of n bits can easily detect a burst errors of n bits, A burst error of more than n hits is also detected by this method with a very high probability. There is. however, one pattern of errors that remains elusive.The error correction can be achieved by using MLDD.

The segments are added using inverted arithmetic to get the sum and corresponding output is processed based on two dimensional mod-2 addition, and then segmented two dimensional mod-2 addition is sent along with data segments. All the received segments has to be computed. The ML decoding algorithm is a hard-decision message-passing algorithm for LDPC codes. A binary hard decision about each received bit is made by the detector and this is passed to the decoder.

Suppose now that a single bit error occurs in the original *d* bits of information. With this **two-dimensional parity** scheme, the parity of both the column and the row containing the flipped bit will be in error. The receiver can thus not only *detect* the fact that a single bit error has occurred, but can use the column and row indices of the column and row with parity errors to actually identify the bit that was corrupted and *correct* that error.

VI)PROPOED LFSR BASED MLS FLOW(CRC)

CRC is based on binary division. In CRC, instead of adding bits to achieve a desired parity, a sequence of redundant bits, called the CRC or the CRC remainder, is appended to the end of a data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number. At its destination, the incoming data unit is divided by the same number.

If at this step there is no remainder the data unit is assumed to be intact and is therefore accepted. A remainder indicates that the data unit has been damaged in transit and therefore must be rejected. and it is allowed Majority logic decoding.

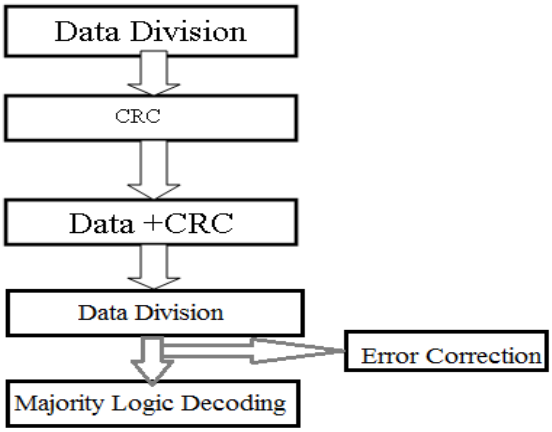

Fig 4.Flowchart of CRC

The concept of the CRC as an error-detecting code gets complicated when an implementer or standards committee turns it into a practical system.While cyclic redundancy checks form part of several standards, they are not themselves standardized to the point of adopting one algorithm of each degree worldwide:

Compute M(x) * Xk
equivalent to adding **k** zeros
example: M(x) **= 1000, G(x) of degree 2**
X3 * X2 = X5 =T(x) (10000)
Divide T(x) by G(x)
Find remainder R(x) = T(x) / G(x)
Replace the Zeros that are added to the message Polynomial with the remainder forms D(x), D(x)is exactly divisible by G(x) **if(D(x)/g(x)=zero)**
Transmit D(x)
elseIf(D(x)/g(x) not equal to zero)
Allow ML decoding till (D(x)/g(x)=)
End if
End process
Fig 5:algorithm of CRC

VII. RESULTS

A. *Memory*

The memory read access delay of the plain MLD is directly dependent on the code size, i.e., a code with length 72 needs 72 cycles, etc. Then, two extra cycles need to be added for

I/O. On the other hand, the memory read access delay of the proposed MLDD is only dependent on the word error rate (WER). If there are more errors, then more words need to be fully decoded.

B.*Area*

The previous subsection showed that the performance of the proposed design Based on two dimensional MLDD is much faster than the plain MLD version, but slightlylower than the design with syndrome calculator (SFD).As mentioned several times, this is compensated with a clear savings in area. The conclusions on the area results are given as follows. The MLDD version has a very similar performance to SFD; however it requires a much lower area overhead, ranging from 10.16% to 0.4 Minimum input arrival time before clock:

Thus the simulation result shows the change of segments with respect to noise added in the data. And it has to be analysed via ML decoding

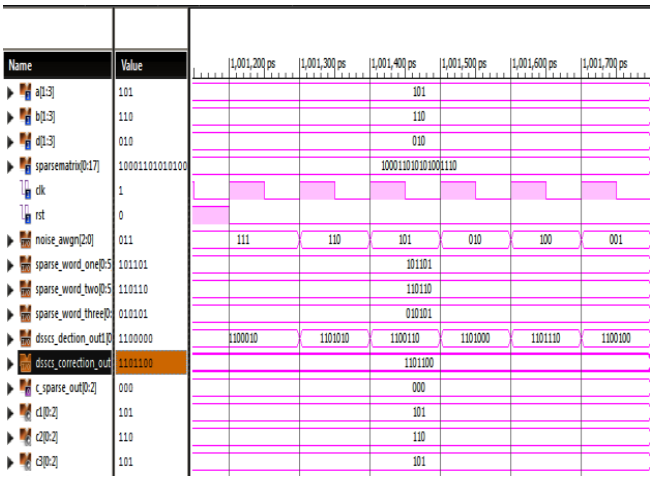

Fig 6:output of 2D parity check

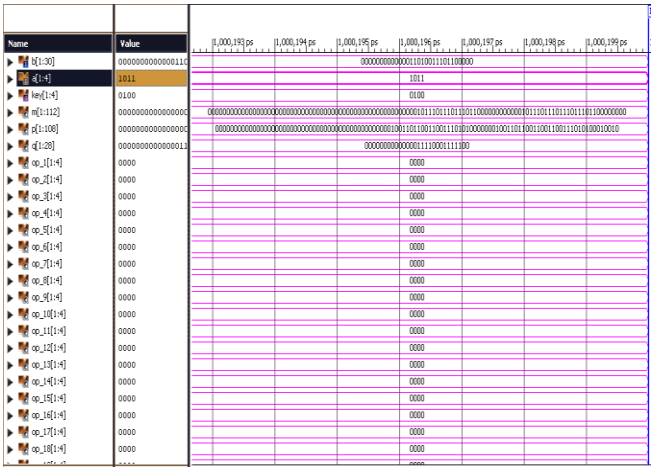

Fig 7: output of CRC

VIII. CONCLUSION

In this paper, a two dimensional method and CRC has been presented which effectively helps to correct errors caused by Multiple Cell Upsets as well as SEU in memories. Additionally, it helps to accelerate the decoding and effectively reduced the area and memory access time than the previously proposed methods .

This makes its area overhead quite reduced compared with other traditional approaches such as the syndrome calculation (SFD). The proposed scheme has been validated by VHDL simulation using a large number of error combinations and memory access time also reduced compared to one dimensional parity check.

ACKNOWLEDGMENT

The authors would like to thank to Anna University and ANGEL groups of institutions for their Valuable support.

REFERENCES

i. *Bane Vasic, Shashi kiran, 'An Information Theoretical Framework for Analysis and Design of Nanoscale Fault-Tolerant Memories Based on Low-Density Parity-Check Codes', IEEE Transactions on circuits and systems,vol.54,no.11,November 2007*

ii. *Costas A. Argyrides, Pedro Reviriego, Dhiraj K. Pradhan, and Juan Antonio Maestro,' Matrix-Based Codes for Adjacent Error Correction',IEEE Transactions on Nuclear Science,vol.4,no.5,August 2010.*

iii. *Charles W. Slayman, 'Cache and Memory Error Detection, Correction, and Reduction Techniques for Terrestrial Servers and Workstations',IEEE transactions on device and material reliability,vol 5 no.3,September 2005*

iv. *Ganesan umanesan, 'A Class of Random Multiple Bits in a Byte ErrorCorrecting and Single Byte Error Detecting Codes', IEEE Transactions on computers', vol. 52, no. 7, july 2003*

v. *Guiqiang Dong,,Ningde Xie, and Tong Zhang,'Enabling NAND Flash Memory Use Soft-Decision Error Correction Codes at Minimal Read Latency Overhead', IEEE Transactions on circuits and systems. —, Vol.no. 60, no. 9, September 2013*

vi. *Junho Cho and Wonyong Sung, Efficient Software-Basedcomputer,vol.58,no.7.july 2009*

vii. *Liu.S, Sorrenti.G, Reviriego .P, Casini. F, Maestro .J. A, Alderighi.M, and Mecha.H,' Comparison of the Susceptibility to Soft Errors of SRAM-Based FPGA Error Correction Codes Implementations',IEEE Transactions of nuclear science ,Vol. 59, No. 3, June 20121.*

viii. *Pedro Reviriego, Salvatore Pontarelli, Juan Antonio Maestro, and Marco Ottavi,' Reducing the Cost of Single Error Correction With Parity Sharing',IEEE Transactions and materials reliability,Vol.13,No.3,September ,2013.*

ix. *Priyanka P. Ankolekar, Roger Isaac, and Jonathan W. Bredow,' Multibit Error-Correction Methods for Latency-Constrained Flash Memory*

x. *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 Systems,Vol.20,No.1,January 2012.*