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 4, September 2014

Implementation of Euclidian Geometry Low Density Parity Check (EG-LDPC) Codes for Error Detection by Using MLDD

B Ramanjaneya Reddy, O Homakesav

M.Tech, VLSISD, AITS, Kadapa, India

Assistant Professor, AITS, Kadapa, India

ABSTRACT: Nowadays, 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. As technology scales, memory devices become larger and more powerful error correction codes are needed. A method was recently proposed to accelerate a serial implementation of majority logic decoding of EG-LDPC codes. 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 errors, then decoding can be stopped without completing the remaining iterations, therefore greatly reducing the decoding time. For a code with block length, 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, there by achieving a large speed increase when is large. In it was shown that for EG-LDPC codes, all error combinations of up to five errors can be detected in the first three iterations. So in the proposed fault detection method significantly reduce the memory access time by using Majority logic decoder/detector.

KEYWORDS: Euclidean geometry low-density parity check (EG-LDPC) codes, Error correction codes, majority logic decoder/detector, Memory, Galois Field.

I. INTRODUCTION

An error-correcting code (ECC) or forward error correction (FEC) code is a system of adding redundant data, or *parity data*, to a message, such that it can be recovered by a receiver even when a number of errors (up to the capability of the code being used) were introduced, either during the process of transmission, or on storage. Since the receiver does not have to ask the sender for retransmission of the data, a back-channel is not required in forward error correction, and it is therefore suitable for simplex communication such as broadcasting. Error-correcting codes are frequently used in lower-layer communication, as well as for reliable storage in media such as CDs, DVDs, hard disks, and RAM.

In one step majority logic decoding, initially the code word is loaded into the cyclic shift register. Then the check equations are computed. 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 means that the current bit under decoding is wrong, and a signal to correct it would be triggered. Otherwise the bit under decoding is correct and no extra operations would be needed on it. In next, the content of the registers are rotated and the above procedure is repeated until codeword bits have been processed. Finally, the parity check sums should be zero if the codeword has been correctly decoded. In this process, each bit may be corrected only once. As a result, the decoding circuitry is simple, but it requires a long decoding time if the code word is large. Thus, by one-step majority-logic decoding, the code is capable of correcting any error pattern with two or fewer errors. For example, for a code word of 15-bits, the decoding would take 15 cycles, which would be excessive for most applications.

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 4, September 2014

II. MLDD

MLDD is a method to decode the repetition codes. Repetition code is one of the most basic error-correcting codes. The idea of the repetition code is to just repeat the message several times. The hope is that the channel corrupts only a minority of these repetitions. This way the receiver will notice that a transmission error occurred. Since the received data stream is not the repetition of a single message and moreover the receiver can recover the original message by looking at the received message in the data stream that occurs most often. A method was recently proposed to accelerate a serial implementation of majority logic decoding of EG-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 errors, then decoding can be stopped without completing the remaining iterations, therefore greatly reducing the decoding time. For a code with block length, 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 increase when is large. In it was shown that for EG-LDPC codes, all error combinations of up to five errors can be detected in the first 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 undetected. 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 that the additional area required to implement the scheme was only around 1% for large word sizes.

Fig.1 shows in LDPC based on the code rate result will be obtained after completion of the cycles. Here in MLD LDPC 63 code rate the result will be obtained after completion of 63 clock cycles, but in first stage output will be obtained after 65 cycles. This is because first three iteration it takes to load reset signal and DATAIN. Clock cycles will be counted based on counter in the design. In our design counter is an intermediate signal so that we have to drag the signal into output waveform to see the iterations. In LDPC if error data is forwarded as output then “ERROR” signal will be “HIGH” and if correct data will be forwarded then “ERROR” signal will be “LOW”.

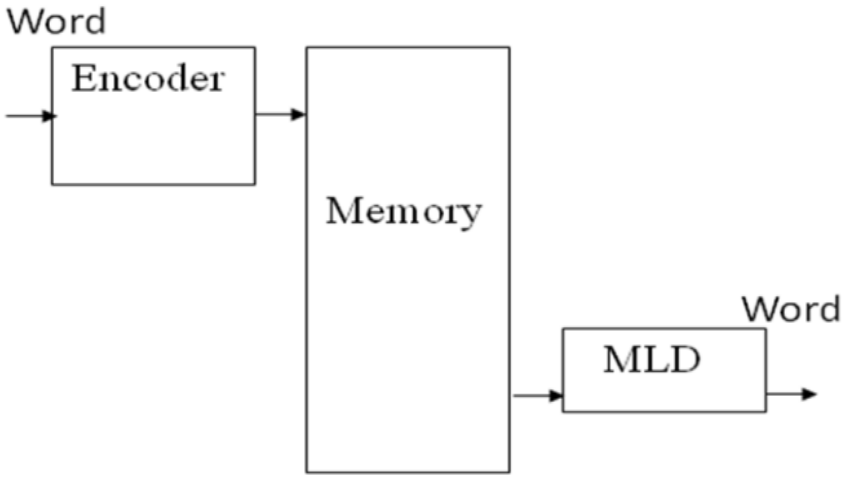


Fig. 1 Memory system schematic with MLD

Fig 3 shows initially the input is stored into the cyclic shift register and it shifted through all the taps. The intermediate values in each tap are given to the XOR matrix which is used to perform the checksum equations. The resulting sums are then forwarded to the majority gate for evaluating its correctness. If the number of 1’s received is greater than the number of 0’s which would mean that the current bit under decoding is wrong, so it move on the decoding process of t It is used to produce the accurate result of the MLDD. Otherwise, the bit under decoding would be correct and no extra operations would be needed on it. Decoding process involving the operation of the content of

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 4, September 2014

the registers is rotated and the above procedure is repeated and it stops intermediately in the third cycle. If in the first three cycles of the decoding process, the evaluation of the XOR matrix for all is "0," the code word is determined to be error-free and forwarded directly to the output. If the error contains in any of the three cycles at least a "1," it would continue the whole decoding process in order to eliminate the errors. Finally, the parity check sums should be zero if the code word has been correctly decoded. Finally the MLDD method is used to detect the five bit errors and correct four bit errors effectively. More than five bit errors it produces the output but it did not show the errors presented in the input.

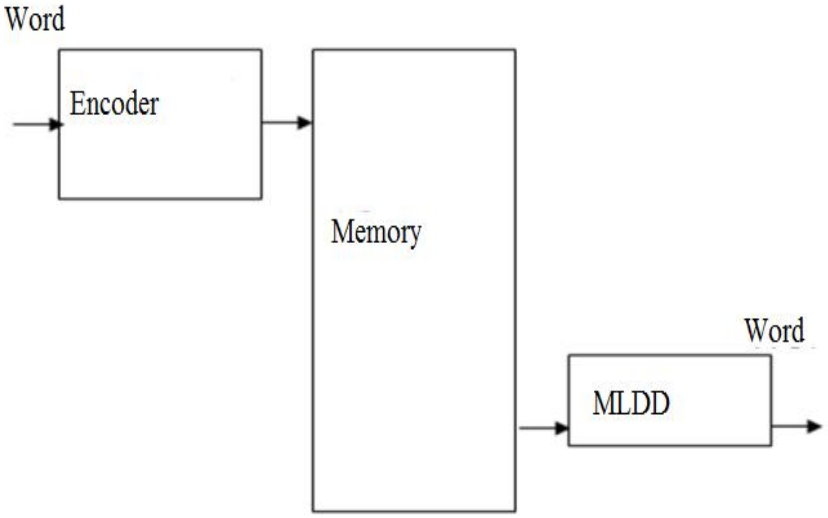


Fig. 2 Memory system schematic with MLDD

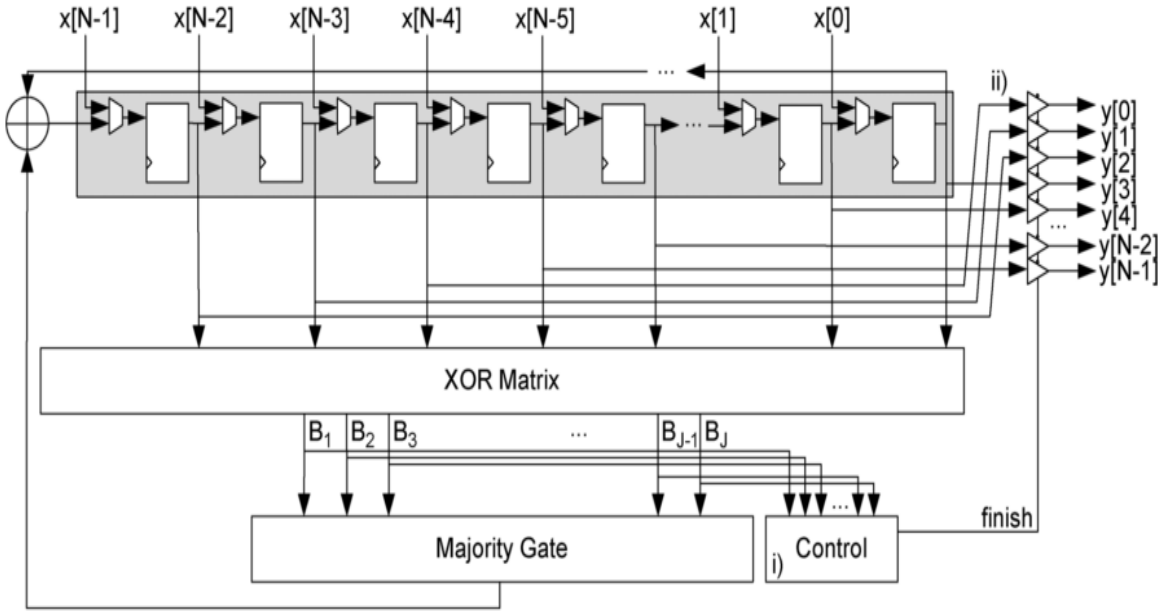


Fig. 3 Schematic Of The Proposed MLDD With Control Logic

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

International Journal of Innovative Research in Computer and Communication Engineering

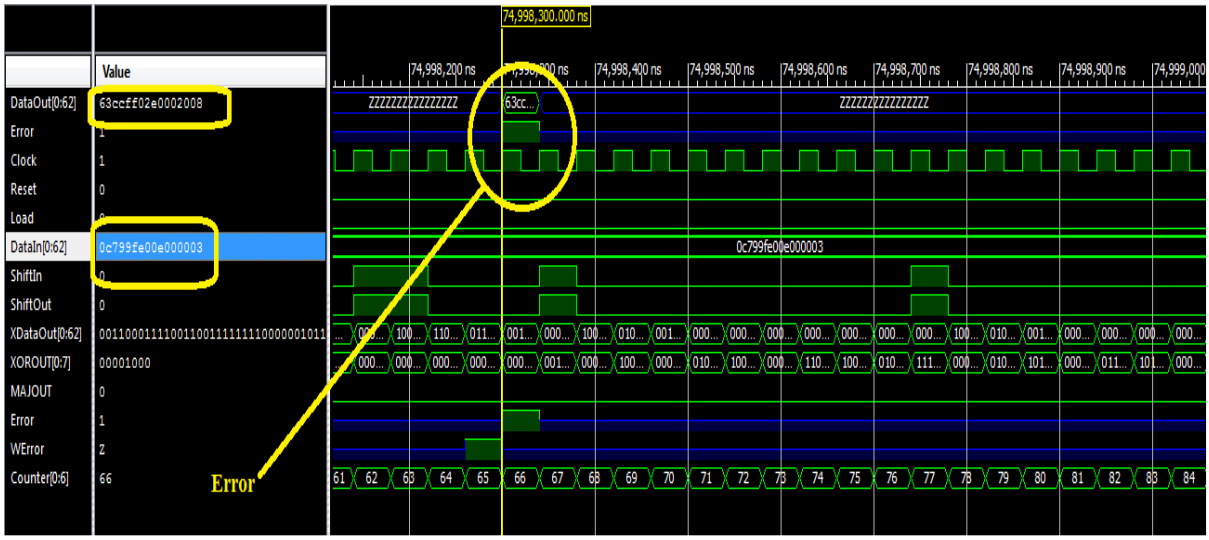
(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 4, September 2014

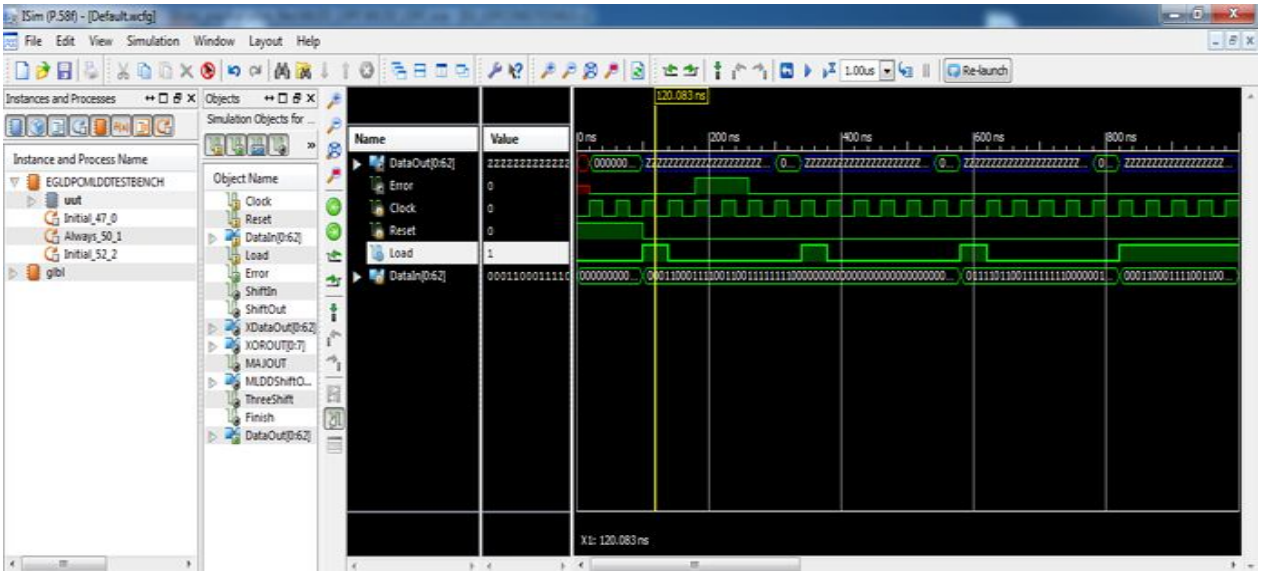
Table-I
ONE STEP MLDD AND MLD EG-LDPC CODES

Method	N	K	No of Cycles
MLD	63	32	63
MLDD	63	32	5

III. SIMULATION RESULT



Simulation waveform of MLD



Simulation waveform of MLDD

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 4, September 2014

IV. CONCLUSION

In this project we will take the no of bit (63) it will take to detect the error by using 63 cycles in MLD. But the proposed method has shows the error it will take around 4 cycles. So finally this project will be reduce the no of cycles based on the simulation results.

REFERENCES

[1] S. Ghosh and P. D. Lincoln, 2007, "Low-density parity check codes for error correction in Nano scale memory," SRI Computer Science Laboratory Tech. Rep.CSL-0703.
[2] Y. Kato and T. Morita, ―Error correction circuit using difference-set cyclic code,‖ in *Proc. ASP-DAC*, 2003, pp. 585–586.
[3] H.Naeimi and A.DeHon, "Fault secure encoder an decoderfor Nano-Memory applications," *IEEE Trans. Very LargeScale Integr.(VLSI) Syst."* , vol. 17, no. 4, pp. 473–486, Apr.2009.
[4] E. J.Weldon, Jr., ―Difference-set cyclic codes,‖ *Bell Syst. Tech. J.*, vol.45, pp. 1045–1055, 1966.
[5] 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.
[6] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004.