International Journal of Engineering and Management Research e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-12, Issue-6, (December 2022)
www.ijemr.net https://doi.org/10.31033/ijemr.12.6.31

Simulation and Synthesis of Efficient Majority Logic Fault Detector Using EG-LDPC Codes to Reduce Access Time for Memory Applications

Suma.J[1], Mahesh B Neelagar[2], Shwetha N[3] and Niranjan L[4]
[1]Assistant Professor, Department of ISE, Sapthagiri College of Engineering, Bangalore, INDIA
[2]Assistant Professor, Department of ECE, VTU, Belagavi, Karnataka, INDIA
[3]Assistant Professor, Department of ECE, Dr. AIT, Bangalore, INDIA
[4]Assistant Professor, Department of ECE, CMRIT, Bangalore, INDIA

[3]Corresponding Author: shwethaec48@gmail.com

ABSTRACT

This paper presents an error-detection method for Euclidean Geometry low density parity check codes with majority logic decoding methodology in VHDL language and the output is verified with the help of Xilinx12.1. Majority logic decodable codes are suitable for memory applications due to their capability to correct a large number of errors. However, they require a large decoding time that impacts memory performance. The proposed fault-detection method significantly reduces memory access time when there is no error in the data read. The technique uses the majority logic decoder itself to detect failures, which makes the area overhead minimal and keeps the extra power consumption low. Starting from the original design of the ML decoder introduced, the proposed ML Detector/Decoder (MLDD) has been implemented using the Euclidean Geometry low density parity check codes. The proposed improved majority logic detector/decoder to perform data error correction in simple way using additional error correction technique and also reducing the delay time by detecting the errors in parallel manner. Hence the decoding process uses less number of cycles which reduces the delay.

Keywords-- Error Correction Codes, Euclidean Geometry Low-Density Parity Check (EG-LDPC) Codes, Majority Logic Decoding, Memory

I. INTRODUCTION

Memories are the most universal component today but they are prone to errors like soft and transient errors. Single Event Upsets (SEU) is the type of fault which alters these memories by changing its states which is caused by ions or electro-magnetic radiations. SEU occurs in digital circuits when an energized particle, or electron, causes a transistor to flip on or off from its correct state. This happens in microcircuits, including memory chips, communication devices, power circuits, and microprocessors. Such a flip of one bit can cause a computer or other electronic device to lockup or crash. Circuit components, including configuration memory cells, user memory, and registers, can also be affected. Some type of embedded memory, such as ROM, SRAM, DRAM, flash memory etc is seen in almost all system chips.

Soft errors that are predominantly caused by external radiation are also known as SEUs. They result in transient, inconsistent errors in data that are unrelated to components or manufacturing failures. Intrinsic noise and interference can also cause SEUs; however, design engineers can accommodate these causes. SEUs manifest themselves as either SBUs (single-bit upsets) or MBUs (multiple-bit upsets). "SBU" refers to the flipping of one bit due to the passage of a single energetic radiation particle, where the physical separation from any other flipped bit is at least two memory cells. MBU refers to the flipping of several elements due to the passage of one or more radiation particles. At the system level, designers can mitigate the increase in SER for SRAMs by using ECC (error-correction-code) detection and correction so that every addressable word of data stored in memory includes check information .The combination of data and check information is often called a check word. The consequence of a memory error is system-dependent. In systems without ECC an error can lead either to a crash or to corruption of data in large-scale production sites memory errors are one of the most common hardware causes of machine crashes. Memory errors can cause security vulnerabilities. Cyclic block codes have the property of being Majority Logic (ML) decodable. Therefore cyclic block codes have been identified as more suitable among the ECC codes that meet the requirements of higher error correction capability and low decoding complexity. Euclidean Geometry Low-Density Parity Check (EG-LDPC) codes, a subgroup of the Low-Density Parity Check (LDPC) codes, which belongs to the family of the ML decodable codes. The ML decoding are that it is very simple to implement and thus it is very practical and has low complexity. 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.

II. PROPOSED SYSTEM

Majority Logic Detector/Decoding (MLDD) technique is generally based on number of parity check equations. These check equations are the results of XOR gates. The inputs to the XOR gates are the fifteen bit data which are stored in the shift registers. A generic schematic of memory system is depicted in Fig .1.In this Fig the input data is sent to the encoder and stored in the memory. The decoding process is done in the MLDD.

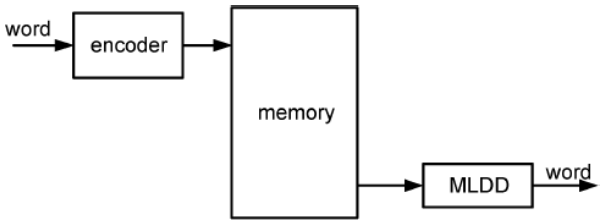

Figure 1: Memory system schematic of MLDD

The codeword used in this technique is the EG-LDPC code (Euclidean Geometry -Low Density Parity Check).It is the One Step Majority Logic Decodable code. This code uses the check sum algorithm. The check sum algorithm is nothing but a numerical value is associated with the code word to be transmitted. Then the code word received at the receiver end has some numerical value. There is a comparison on the associated numerical values to detect the error. The existing method is implemented using simple hardware. The decoding time for this method is more. Also the power consumption and area requirement are high. To detect the errors serially the MLDD technique uses Serial One Step Majority Logic Decoder. The serial one step majority logic decoder is depicted in Fig.2.

It is the One Step Majority Logic Decodable code. This code uses the check sum algorithm .The check sum algorithm is nothingbut a numerical value is associated with the code word to be transmitted. Then the code word received at the receiver end has some numerical value. There is a comparison on the associated numerical values to detect the error. The existing method is implemented using simple hardware. The decoding time for this method is more. Also the power consumption and area requirement are high. To detect the errors serially the MLDD technique uses Serial One Step Majority Logic Decoder. The serial one step majority logic decoder is depicted in Fig.2.

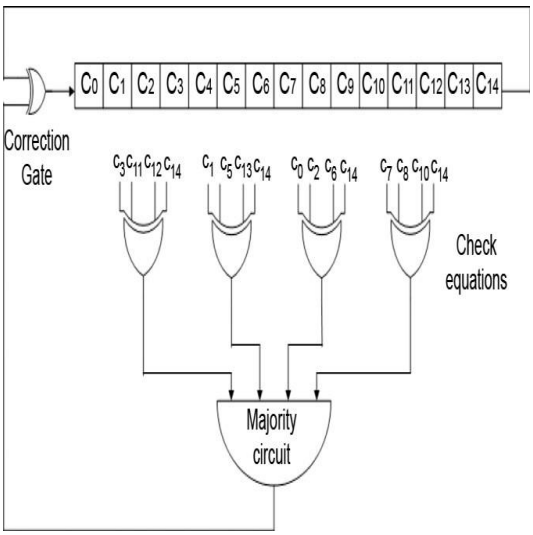

Figure 2: Serial one-step majority logic decoder for the (15,7) EG-LPDC code

In this decoder 15 bit data is first stored in the cyclic shift register. Then the inputs are assigned to the XOR gates. Since there is 15 bit data the XOR gates required are four. The bit to be detected should be given

as one of the inputs for all the XOR gates. The outputs of the XOR gates are the check sum equations. The check sum equations consist of binary data. Then the Majority circuit outputs the data which is in major number. If the output of the majority circuit is „1" then the corresponding bit has the error else thebit is error free. The schematic of the MLDD is shown in Fig.3. It consists of cyclic shift registers, XOR matrix, Majority gate, control unit, tri state buffers and XOR gate. In this schematic the control unit has the important characteristic. It stores the result of three iterations. After receiving the results of three iterations it sends "finish" signal to the tri state buffer which indicates the data is error free. The tri state buffer is in high state until it receives the finish signal. After receiving the finish signal it goes low and outputs the current data from cyclic shift register. The output of the buffer is the detected data.

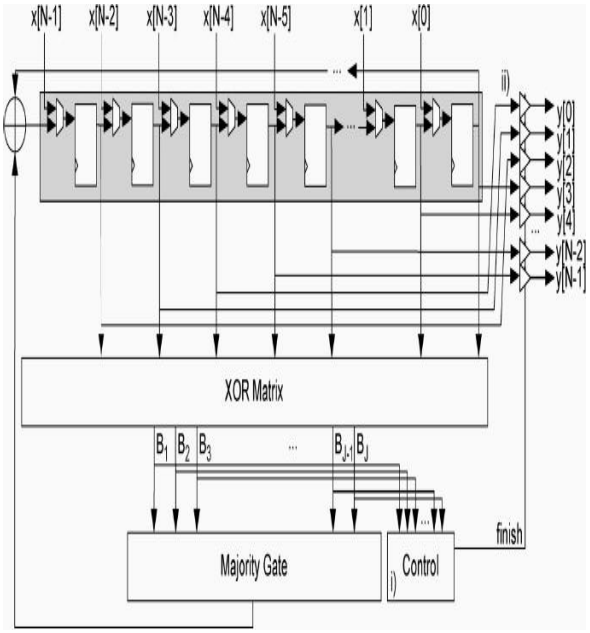

Figure 3: Schematic of MLDD with Control unit

In the proposed technique the error detection process is done in parallel and in pipelining manner. The iteration required for detection is only one. Thus the delay time is comparatively low. Also the power consumption and the area requirement is low. Considering the Fig.2. The entire 15 bit data is detected simultaneously in single iteration. But the cyclic shift is same as in serial error detection process. Error detection process is also done in pipelining manner. In this process area requirement is further reduced compared to parallel processing.

III. ALGORITHMS

1. MLD Algorithm

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 candidates, due to their property of being Majority Logic (ML) decodable. A subgroup of the Low-Density Parity Check (LDPC) codes, which belongs the main reason for using ML decoding is that it is very simple to implement and thus it is very practical and has low complexity. The drawback of ML decoding is that, for a coded word of N-bits, it takes N cycles in the decoding process, posing a big impact on system performance to the family of the ML decodable codes. One way of coping with this problem is to implement parallel encoders and decoders. This solution would enormously increase the complexity and, therefore, the power consumption. As most of the memory reading accesses will have no errors, the decoder in most of the time is working for no reason. This has motivated the use of a fault detector module that checks if the codeword contains an error and then triggers the correction mechanism accordingly. In this case, only the faulty code words need correction, and therefore the average read memory access is speeded up, at the expense of an increase in hardware cost and power consumption. A similar proposal has been presented in for the case of flash memories. The simplest way to implement a fault detector for an ECC is by calculating the syndrome, but this generally implies adding another very complex functional unit. This paper explores the idea of using the ML decoder circuitry as a fault detector so that read operations are accelerated with almost no additional hardware cost.

2. MLDD Algorithm

The Difference Set Cyclic Codes (DSCC) used in the majority logic decoding technique has the drawback of detecting the errors for N clock cycles. This drawback is overcome in Majority Logic Detector/Decoding (MLDD) technique. The code used in this algorithm is Euclidean Geometry- Low Density Parity Check Codes (EG-LDPC). Itis One-Step Majority Logic Decodable technique. The algorithm embedded here is Check Sum algorithm where a numerical value is always associated with the data to be transmitted over the memory. So that the data received at the receiver side is verified by using the numerical value. This process is feasible for three decoding cycles where most of the errors are detected.

Flow Chart of MLDD Algorithm

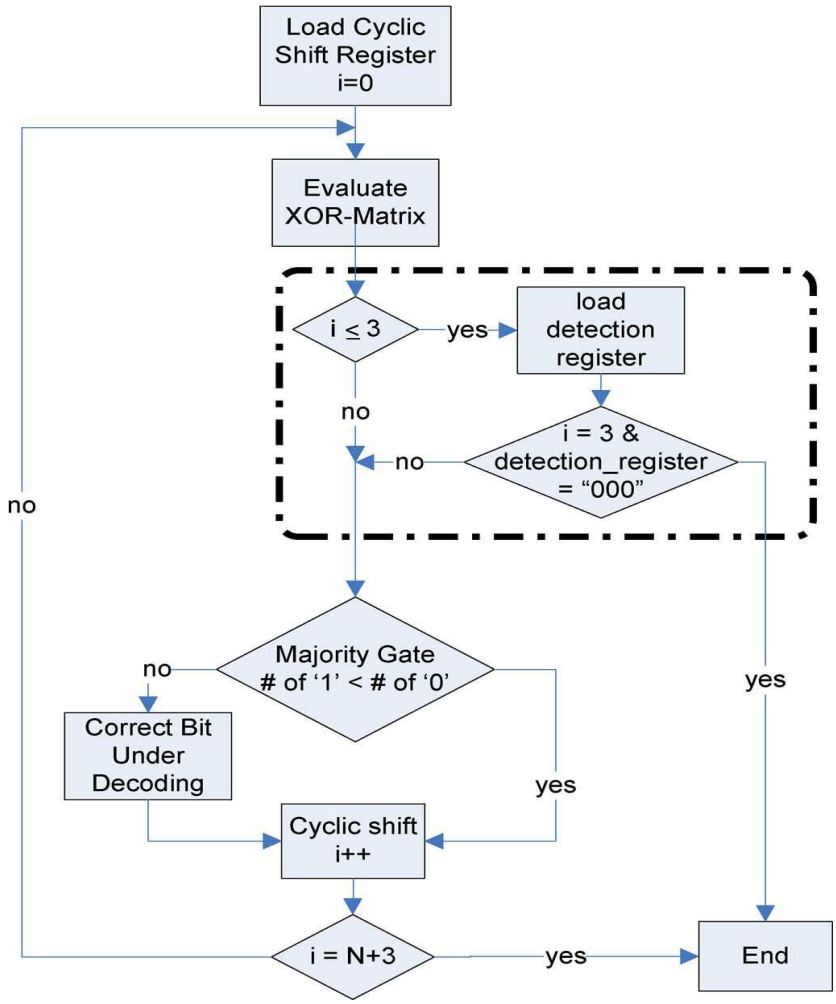

Figure 4: Flow Diagram Of The MLDD Algorithm

3. Low Density Parity Check (LDPC)

A Low-Density Parity-Check (LDPC) code is a linear error correcting code, a method of transmitting a message over a noisy transmission channel, and is constructed using a sparse bipartite graph. Low Density Parity-Check code (LDPC) is an error correcting code used in noisy communication channel to reduce the probability of loss of information With LDPC, this probability can be reduced to as small as desired, thus the data transmission rate can be as close to Shannon's limit as desired. LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close to the theoretical maximum (the Shannon limit).for a symmetric memory-less channel. The noise threshold defines an upper bound for the channel noise, up to which the probability of lost desired LDPC codes are finding increasing use in applications requiring reliable and highly efficient information transfer over bandwidth or return channel constrained links in the presence of data corrupting noise An LDPC code beat six turbo codes to become the error correcting code in the new DVB-S2 standard for the satellite transmission of digital television. LDPC beat convolution turbo codes as the FEC scheme because of

its lower decoding complexity. The name comes from the characteristic of their parity-check matrix which contains onlya few 1‟s in comparison to the amount of 0‟s. Their main advantage is that they provide a performance which is very close to the capacity for a lot of different channels and linear time complex algorithms for decoding.

4. Euclidean Geometry Low-Density Parity Check (EG-LDPC) Codes

Euclidean Geometry Low-Density Parity Check (EG-LDPC) codes, a subgroup of the Low-Density Parity Check (LDPC) codes, which belongs to the family of the ML decodable codes, is focused here. 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. Let EG be a Euclidean Geometry with points and J lines. EG is a finite geometry that is shown to have the following fundamental structural properties are

- Every line consists of points
- Any two points are connected by exactly one line
- Every point is intersected by lines
- Two lines intersect in exactly one point or they are parallel i.e., they do notintersect

IV. SIMULATION RESULTS

Simulation results are shown for three blocks such as for encoder, memory, majority logic decoder/detector with control and for final serial majority logic decoder.

Simulation Result for Encoder

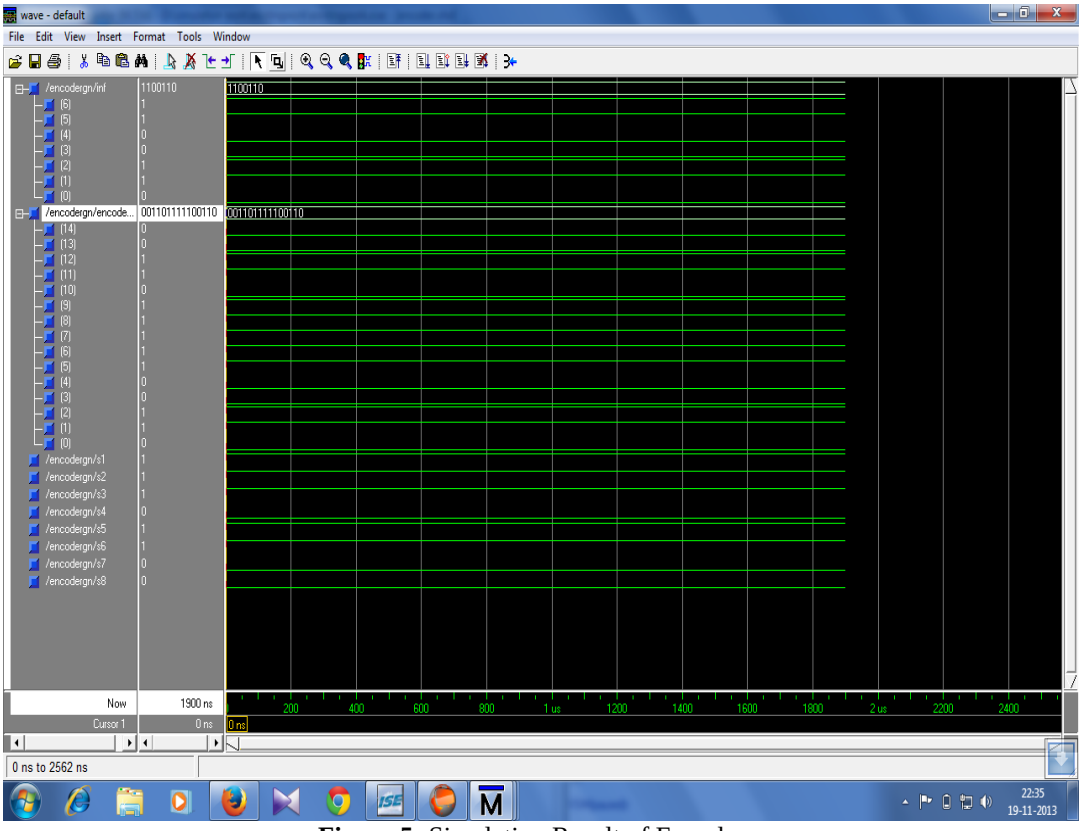

Figure 5: Simulation Result of Encoder

Figure 5 shows the simulation result of an Encoder. The data input to given to the memory is encoded first through this encoder block. The input to the encoder is the 7 bit information and output is the 15 bit information. The signal output is form s_1 to s_8.

Simulation Result of Memory

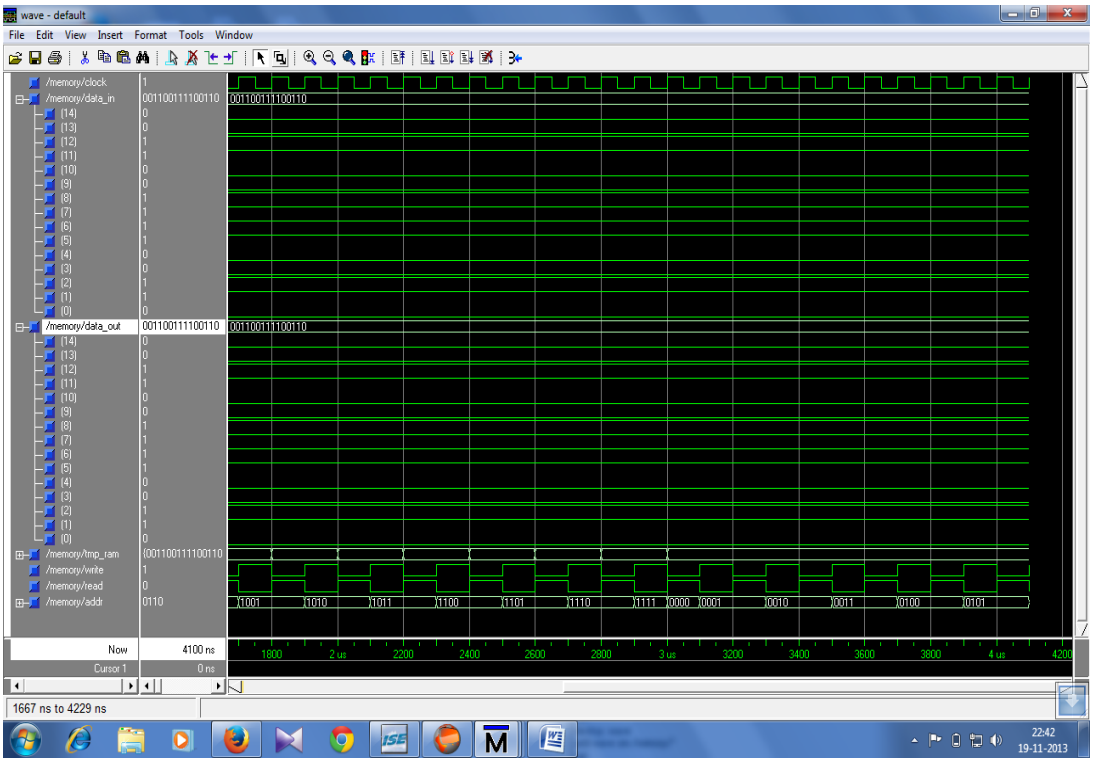

Figure 6: Simulation Result of Memory

Figure 6 shows the simulation result of the memory. The data in and data out of the memory is the 15 bit information taken from the encoder. The temporary RAM is used as memory element. The clock event is used to read from the memory and write data to the memory. The memory address of the particular data is also specified.

Simulation Result of One-Step Majority LogicDecoder

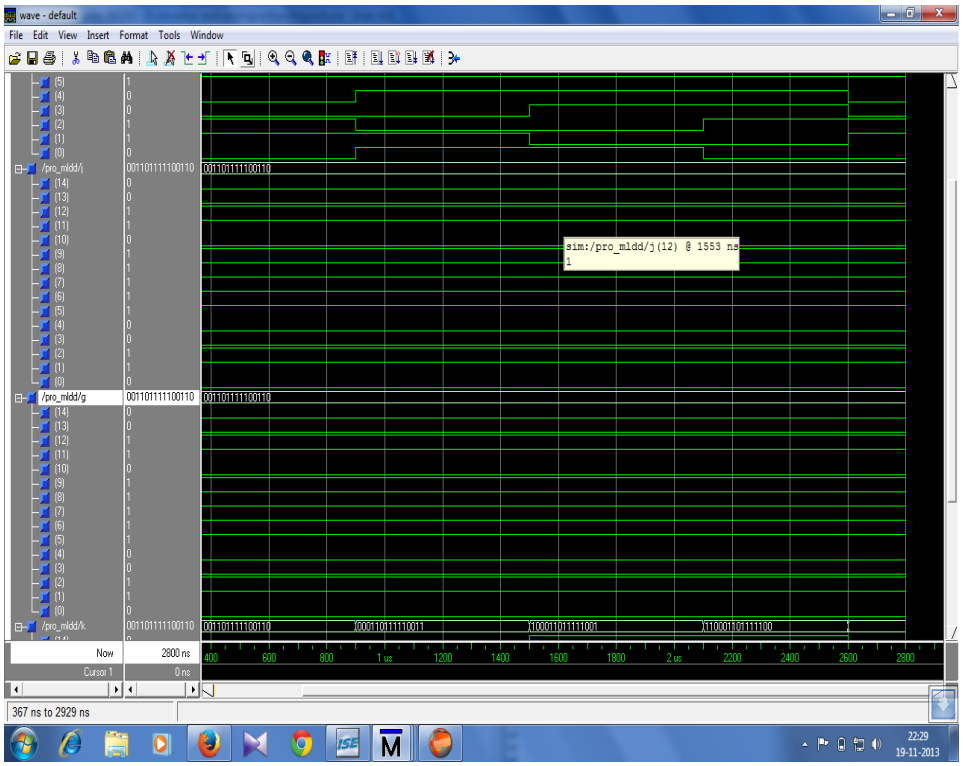

Figure 7: Simulation Result of One-Step MLD.

Figure 7 shows the output of the Serial Majority logic decoder. In this decoder the data read from the memory is given as serial input to the decoder. The data is verified whether it is with error or without error by serial cyclic shifting. This method is called One Step Majority Logic Decoding.

Simulation Result of Parallel MLDD

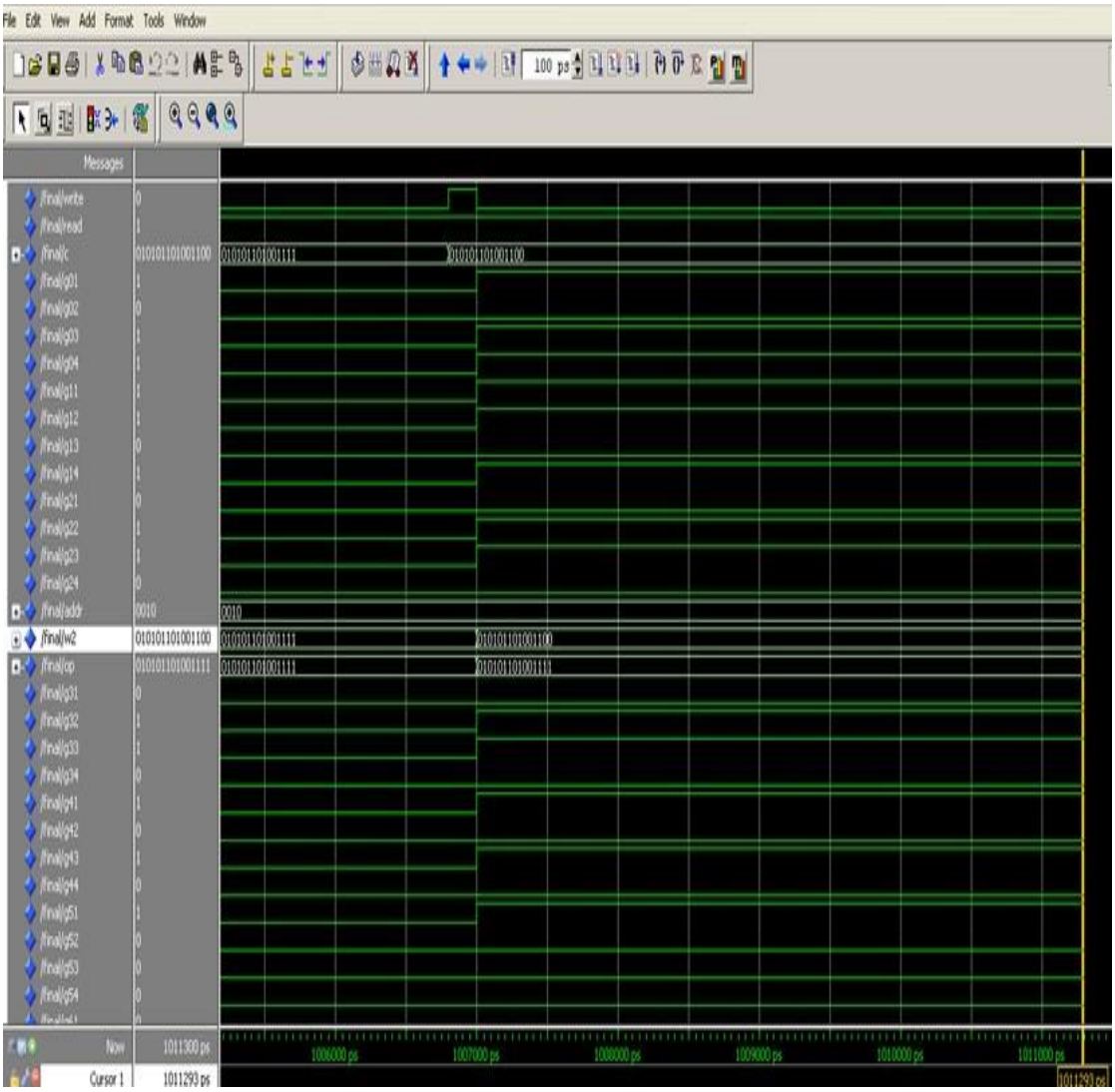

Figure 8: Simulation Result of Parallel MLDD

Figure 8. shows the output of the parallel Majority logic decoder. In this decoder the data read from the memory is given as parallel input to the decoder. All codeword bits are decoded at the same time. The data is verified whether it is with error or without error in single iteration.

Power Analyzer

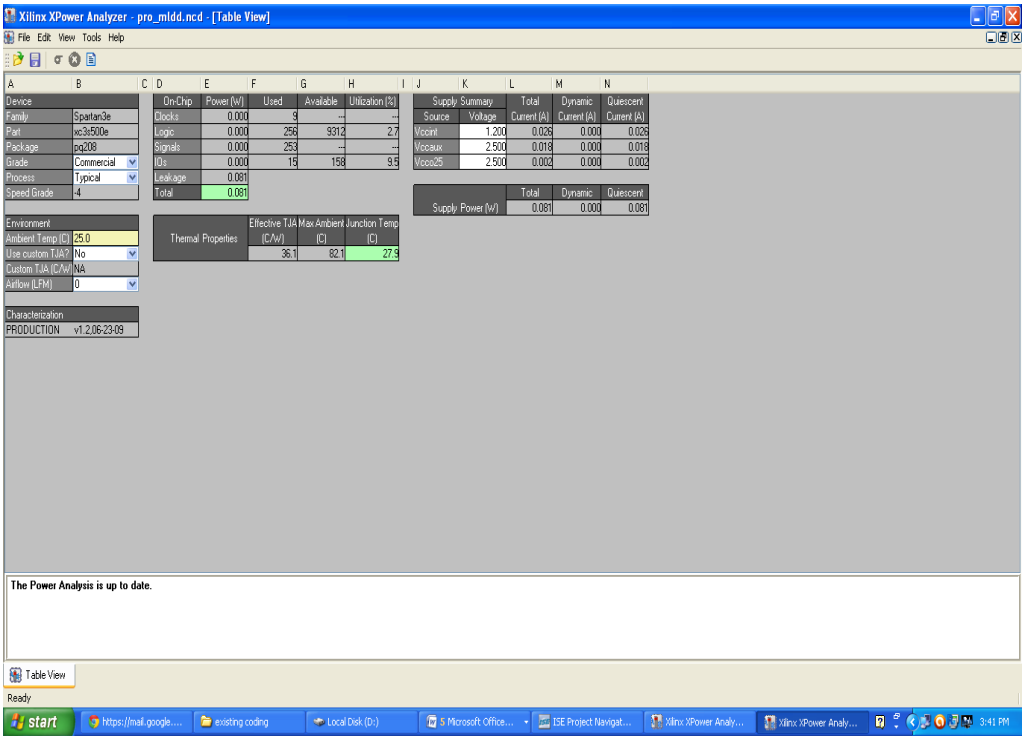

Figure 9: Power consumed by serial MLD

Figure 9 shows the total power consumed by the serial majority logic decodingtechnique.

Synthesis of Area

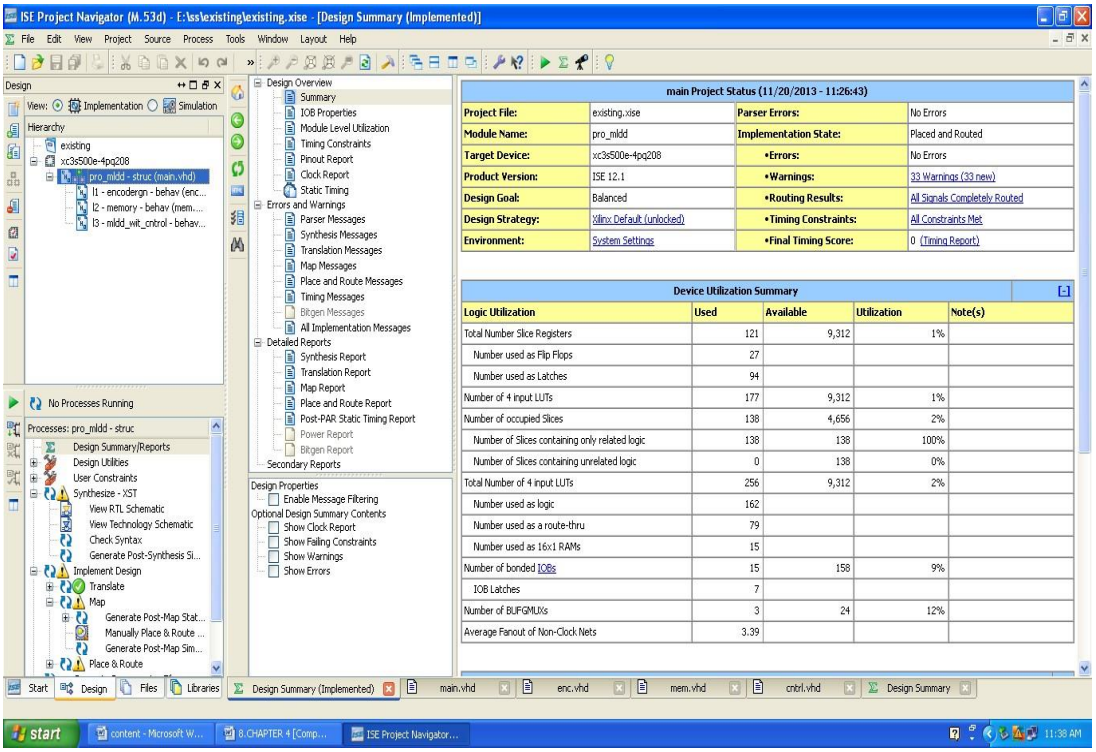

Figure 10: Area consumption for the serial MLD

Figure 10 shows the area requirement of the serial majority logic decodingtechnique

Timing Constraint

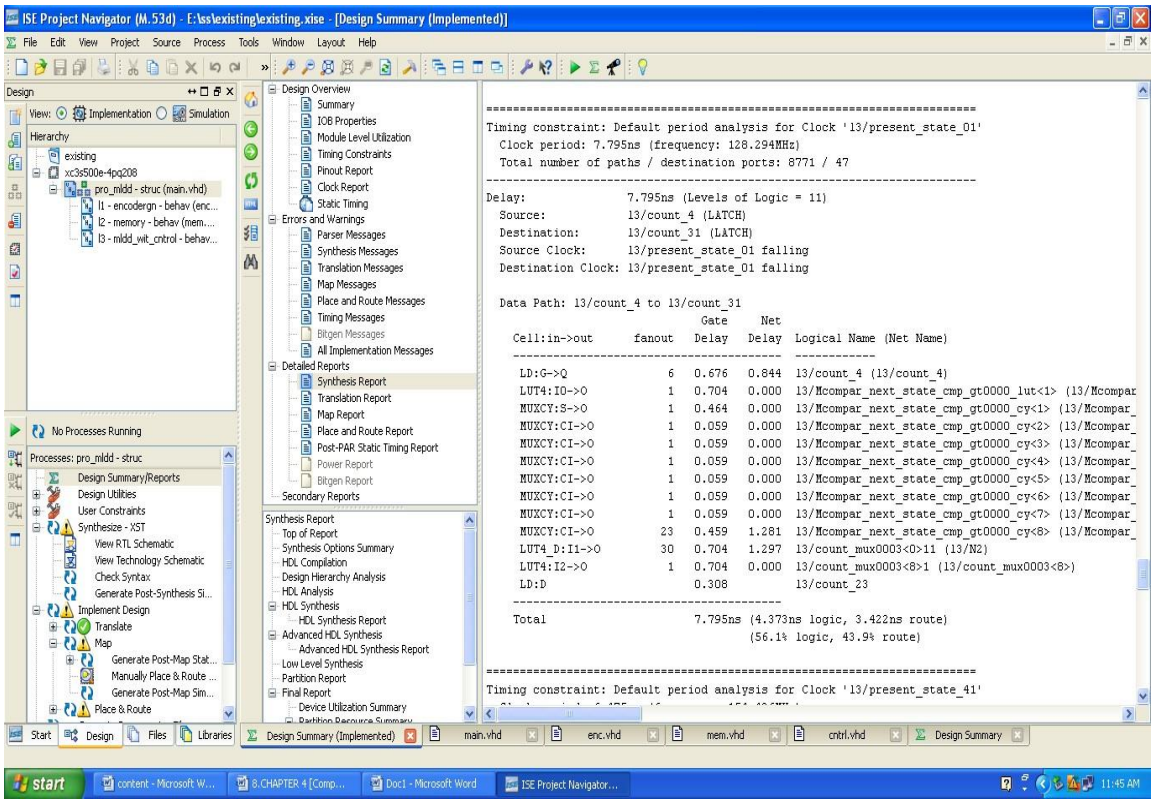

Figure 11: Delay Constraint for the Serial MLD

Figure 8.8 shows the delay time of the serial majority logic decoding technique forthe error correction. The proposed method has been applied to one step majority logic decoding EG-LDPCcodes. The results are presented with the help of VHDL simulation. For codes with minimum words and affected by a minimum number of bit flips, it is possible to generate and check all possible error combinations. As the code size increases and the number of bit flips increases, itis no longer possible to exhaustively test all possible combinations. Therefore the simulations are done in two ways, by exhaustively checking all error combinations when it is feasible and by checking randomly generated combinations in the rest of the cases. The simulation results show the error detected in only one iteration and most ofthe errors are detected.

The power and delay for serial and parallel MLDD is shown in table 1

Table I

	Serial MLD technique	Parallel MLD technique
Power	0.087 watts	0.081 watts
Delay	7.795 ns	6.376 ns

IV. CONCLUSION AND FUTURE SCOPE

The detection of errors during the first iterations of Serial One Step Majority Logic Decoding of EG-LDPC codes and one step majority logic corrector when implemented serially provides compact implementation and when implemented in parallel minimize correction latency has been studied. The objective was to reduce the decoding time by stopping the decoding process when no errors are 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. Error detection mechanism, Parallel MLDD has been presented based on the ML decoding used to the EG-LDPC. Single iteration is required for detection of any number of errors. Thus the delay time is reasonably low. These

results extend the ones recently presented for EG-LDPC codes, making the modified one step majority logic decoding more attractive for memory applications. Thus the error detection process using parallel technique is more efficient for the memory application. 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. Also detection of errors can be done through pipelining which reduces the delay time and power consumption. 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 decoding time and error detection capability.

REFERENCES

[1] S. Liu, P. Reviriego & J. Maestro. (2012). Efficient majority logic fault detection with difference-set codes for memory applications. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, *20*(1), pp. 148–156.

[2] S. Ghosh & P. D. Lincoln. (2007). Low-density parity check codes for error correction in nanoscale memory. *SRI Computer Science Lab., Menlo Park, CA, Tech. Rep. CSL-0703*.

[3] R. C. Baumann. (2005. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Trans. Device Mater. Reliab.*, *5*(3), pp. 301–316.

[4] H. Naeimi & A. DeHon. (2007). Fault secure encoder and decoder for memory applications. In: *Proc. IEEE Int. Symp. Defect Fault Toler. VLSI Syst.*, pp. 409–417.

[5] B. Vasic & S. K. Chilappagari. (2007). 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*, 54(11), pp. 2438–2446.

[6] S. Lin & D. J. Costello. (2004). *Error control coding*. (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall.

[7] H. Tang, J. Xu, S. Lin & K. A. S. Abdel- Ghaffar. (2005). Codes on finite geometries. *IEEE Trans. Inf. Theory*, *51*(2), pp. 572–596.

[8] M. A. Bajura, Y. Boulghassoul, R. Naseer, S.Das Gupta, A. F. Witulski, J. Sondeen, S. D. Stansberry, J. Draper, L. W. Massengill & J. N. Damoulakis. (200). Models and algorithmic limits for an ECC-based approach to hardening sub-100-nm SRAMs, *IEEE Trans. Nucl. Sci.*, *54*(4), pp. 935–945.

[9] R. Naseer & J. Draper. (2005). DEC ECC design to improve memory reliability in sub-100 nm technologies. *Proc. IEEE ICECS*, pp. 586–589.

[10] R. C. Baumann. (2005). Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Trans. Device Mater. Reliabil*, *5*(3), pp. 301–316.

[11] M. A. Bajura et al. (2007 Aug). Models and algorithmic limits for an ECC-based approach to hardening sub-100-nm SRAMs. *IEEE Trans. Nucl. Sci.*, *54*(4), pp. 935–945.

[12] R. Naseer & J. Draper. (2008). DEC ECC design to improve memory reli-ability in sub-100 nm technologies. In: *Proc. IEEE ICECS*, pp. 586–589.

[13] C. W. Slayman. (2005). Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations. *IEEE Trans. Device Mater. Reliabil*, *5*(3), pp. 397–404.

[14] B. Vasic & S. K. Chilappagari. (2007). An information theoretical frame-work for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes. *IEEE Trans. Circuits Syst. I, Reg. Papers*, *54*(11), pp. 2438–2446.

[15] H. Naeimi & A. DeHon. (2009). Fault secure encoder and decoder for nano memory applications. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, *17*(4), pp. 473–486.

[16] Y. Kato & T. Morita. (2003). Error correction circuit using difference-set cyclic code. In: *Proc. ASP-DAC*, pp. 585–586.

[17] P. Ankolekar, S. Rosner, R. Isaac &d J. Bredow. Multi-bit error correction methods for latency contrained flash memory systems. *IEEE Trans*.

[18] Shwetha, N., Niranjan, L., Chidanandan, V. & Sangeetha, N. (2021), Smart driving assistance using Arduino and proteus design tool. *Expert Clouds and Applications*, 647–663. https://doi.org/10.1007/978-981-16- 2126-0 51.

[19] Shwetha, N., Niranjan, L., Chidanandan, V. & Sangeetha, N. (2021). Advance system for driving assistance using Arduino and proteus design tool. *Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*. Available at: https://doi.org/10.1109/icicv50876.2021.9388620.