INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Efficient Error Detection And Correction For Memory Applications Using Majority Logic Decoder And Detection

Ms. Susmitha B.C
Department of Electronics and Communication Engineering
B.G.S. Institute of Technology, B.G.Nagar, Mandya, India
Mr. Naveen K.B
Department of Electronics and Communication Engineering
B.G.S. Institute of Technology, B.G.Nagar, Mandya, India

Abstract:

Memory is a basic resource in every digital systems but nowadays, single event upsets (SEU) altering these memories by changing its states which is caused by ions or electro-magnetic radiations. Error-correcting code memory (ECC memory) is a type of computer data storage that can detect and correct the more common kinds of internal data corruption. This paper presents an error detection and correction method for Euclidean geometry Low density parity check (EG-LDPC) codes with majority logic decoding. Here the application is mainly focused on memories, since MLDD is used here due to its capability of correcting large errors. Even though they require a large decoding time that will more affecting on memory performance. This can be overcome by the proposed technique which significantly reduces memory access time and also it take three iterations instead of N iterations when there is no error in the data read and it uses majority logic decoder itself to detect failures which minimizes the area and power consumption.

Keywords: *Error correction codes, Euclidean geometry Low density parity check (EG-LDPC), Block codes, Majority logic decoder and detection (MLDD),*

1.Introduction

The memory is a very basic need for all systems, some types of memories such as ROM, SRAM, DRAM, flash memory etc is present in almost all system chips. The impact of technology scaling, smaller dimensions, higher integration densities, and lower operating voltages etc. [1], [2]. This has come to a level that reliability of memories is put into jeopardy, not only in extreme radiation environments like spacecraft and avionics electronics, but also at normal terrestrial environments [3], [1]. Memory is processed by which information is encoded, stored and retrieved. While retrieving the information which is encoded should be uncorrupted. So it is very important to protect memory against error. Some commonly used error identification techniques are Triple modular redundancy (TMR) and Error correction codes (ECCs).

1.1.Triple Modular Redundancy (TMR)

The TMR triplicates all the memory parts of the system and to choose the correct data using a voter. To utilize triple modular redundancy, a ship must have at least three chronometers. At one time, the cost of three sufficiently accurate chronometers was more than the cost of a smaller merchant vessel [4].

1.2.Error Correction Codes (ECCs)

ECC protects against undetected data corruption, and is used in memories, it also reduces the number of crashes in memory, ECC memory maintains a memory system immune to single-bit errors: the data that is read from each word is always the same as the data that had been written to it, even if a single bit actually stored, or more in some cases, has been flipped to the wrong state [1].

For example Single Error Correction (SEC) codes that can correct one error in a memory word are commonly used. Parity allows the detection of all single-bit errors (actually, any odd number of wrong bits). More advanced ECCs are also used when additional protection is needed. While the error correction capability of a code is important, it is also important to detect errors that cannot be corrected to avoid Silent Data Corruption (SDC). For that reason, codes that can also detect double errors are preferred. Those are known as Single Error Correction Double Error Detection (SEC-DED) codes [5].

The usual multi error correction codes, such as Reed- Solomon (RS) or Bose Chaudhuri-Hocquenghem (BCH) are not suitable for this task due to complex decoding algorithm. These are most sophisticated decoding algorithms, like complex algebraic decoders that can decode in fixed time and simple graph decoders. Both are very complex and increase computational cost [8]. Cyclic codes are linear block error-correcting codes that have convenient algebraic structures for efficient error detection and correction. Therefore cyclic codes are more suitable among ECC codes that meet the requirements of high error correction capability and low decoding complexity because of the majority decodable [7][9].

Euclidean geometry low density parity check (EG-LDPC) codes, a sub-group of the low density parity check (LDPC) codes, which belongs to the family of ML decodable codes, which is focused in this paper [6]. The main reason for using ML decoding is because of its low complexity and simple to implement. The main drawback of this ML decoding is that, for a code word of N-bits, it takes N cycles in the decoding process for both error and error free code words. This causes a big impact on system performance [8]. Error detection in a block codes can be implemented by computing syndrome and checking whether all bits are zero [10]. By calculating the syndrome, we can implement a fault detector for an ECC is but this also would add an additional complex functional unit.

This paper proposes the MLD circuitry itself as an error detecting module therefore with no additional hardware the read operations could be accelerated. The results show that the properties of EG LDPC enable efficient fault detection.

2.Majority Logic Decoder (MLD)

Majority logic decoding is the relatively efficient with low complexity error-correcting technique [6]. MLD is based on the number of parity check equations which are orthogonal to each other, so that, at each iteration, each codeword 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 general steps of memory schematic with MLD is that word is first encoded and is then written to the memory [11]. After memory reads the word the it is passed to a majority logic detector block which detects and corrects the errors which occurred while the reading codeword. This generic schematic of memory system is duplicated in Fig.1 for the usage of an ML decoder.

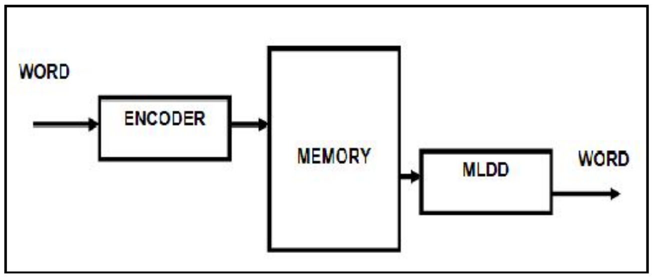

Figure 1: Memory system with MLD

In this type of decoding the data word is corrected from all bit-flips that it might have suffered while being stored in the memory. This type of decoder can be implemented in two ways. The first one is called the Type-I ML decoder, which determines the bits need to be corrected from the XOR combinations of the syndrome [8], The Type-II ML decoder that calculates the information of correctness of the current bit under decoding, directly out of the codeword bits. Both are quite similar, but when implementation is considered the Type-II uses less area, since it does not have syndrome calculation as an intermediate step. For this reason this paper is focused on Type-II implementation.

2.1.Existent Plain ML Decoder

The existent plain majority decoder has the Error control capability refers to mechanism to detect and correct the errors that occur in the codeword. The most common techniques for error control are as follows Error detection, Positive acknowledgement, retransmission after time out and acknowledgement and retransmission. As described befor, ML decoder is a simple and powerful decoder, capable of correcting multiple random bit-flips depending on the nimber of parity check equations. It consists of four parts:

- A cyclic shift register
- An XOR matrix
- A majority gate
- An xor

for correcting the codeword bit under decoding, as illustrated in Fig. 2.

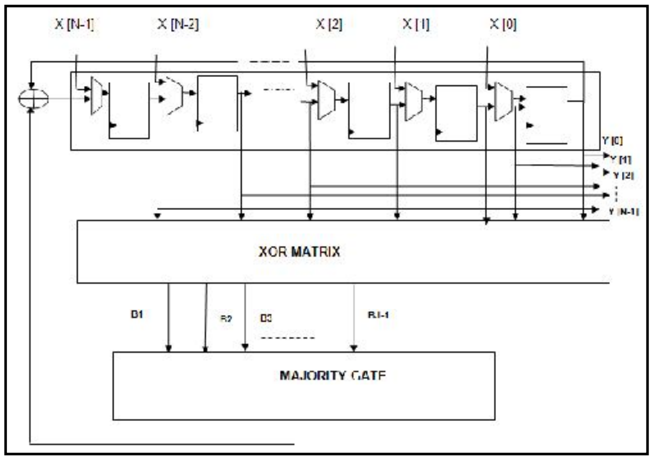

Figure 2: Schematic MLD method

The input codeword bits are stored in the cyclic shift register and shifted to all the registers. It circulates all codeword bits of the register around both MSB and LSB ends with no loss of information. The schematic of the MLDD method is shown in Fig 2. The cyclic shift register consists of two modules are D Flip flop and Multiplexers.

The in-between bits in each register are used to analyze the results from XOR matrix. In the Nth cycle, the results have attained the final register, and make the output bits. The codeword are corrupted by soft error which results in the wrong codeword. The codeword is passed to the shift register; the decoding process begins by calculating the parity check equation in the XOR matrix.

The resulting values are forwarded to the majority gate module for calculating the correctness. If the number of 0’s is lesser than number of 1’s, that indicates the current bit of the codeword is wrong. To mitigate this problem, trigger signal is passed to correct the codeword. Otherwise there is no extra operation is needed which means the current bit is error-free.

The codeword in the register are rotated and the above process is taken place to check all the codeword. If there are N bits, the ML decoder process takes N iteration.

This algorithm needs N iteration for N bit codeword input. The main demerits are that performances of the MLD scheme based on the codeword size.

2.2 Plain MLD with Syndrome Fault Detector (SFB)

Syndrome vector method overcomes the demerits of Majority Logic Decoder (MLD) method. The faulty codeword are decoder, by adding the fault detector which calculates the syndrome value. This will not affect the performances of the system because most of the codeword are error-free. The main drawback of this system increases the complexity to the design. Based on parity check equation, the XOR matrix calculates the syndrome value. This increases the complexity of the syndrome value vector based on the size of the codeword.

An error in the codeword is identified when the syndrome vector value is ‘1’, then the ML decoder is used to correct the wrong codeword. Otherwise it forwards the codeword to the output, without correcting cycles. In this method, the performance is improved the performances of the system but additional module which increases the complexity to the design. Further, it increases the power complexity and reduces the performances of the system. It will increase the power consumption. Syndrome vector is oldest technology, which is used to detect the error in the codeword. Syndrome decoder is linear decoder. Hamming code is one of the examples of syndrome decoder.

Thus the proposed MLDD method overcomes the demerits of above existing method. If the 73-bit codeword is error-free, then output will be processed in the three iteration. It helps to obtain the result in three cycles.

3.Proposed Majority Logic Decoder/Detector

The proposed ML detector/decoder (MLDD) has been implemented using the Euclidean Geometry LDPC [6]. The EG-LDPC are based on the structure of Euclidean geometries, among EG-LDPC codes there is an subclass of codes that is one step majority logic decodable (MLD) [12]. The below Fig. 3 shows the memory system schematic of MLDD.

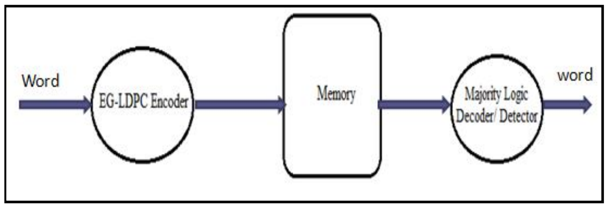

Figure 3: Schematic of a Memory system with MLDD

This method is very practical to generate and check all possible error combinations for codes with small words and affected by a small number of bit flips. When the size of code and the number of bit flip increases, it is difficult to exhaustively test all possible combinations. Therefore the simulations are done in two ways, the error combinations are exhaustively checked when it is feasible and in the rest of the causes the combinations are checked randomly. Since it is convenient to first describe the chosen design and also for simplicity, let us assume that the hypothesis is true, that only three cycles are needed to detect all errors affecting up to four bits[12] in EG LDPC Codes.

3.1.Encoder

The encoder structure of a systematic code calculates the parity function of each parity bit based on the information bits. Each parity function is a xor gate. Therefore the encoder circuitry consists of (n − k) xor gates.

An n-bit codeword c, which encodes a k-bit information vector i is generated by multiplying the k-bit information memory. The encoder vector consists of two parts, the first part consists of information bits and second part is the parity bits, where each parity bit is simply an inner product of information vector and a column of X, from G=[1:X]. The encoder circuit to compute the parity bits of the (15, 7, 5) EG-LDPC code is shown in the Fig 4. In this figure the information vectors are (i0, i1,......i6) and will be copied to (c0,...,c6) bits of the encoder vector, c. The rest of encoded vector (c7,...c14), that is the parity bits are the linear sums (XOR) of the information bits.

$$\begin{array}{c} \\ i_0 \\ i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \\ \\ \end{array}
\begin{array}{c}
\begin{array}{ccccccccccccccc} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} \end{array} \\
\left[\underbrace{\begin{array}{ccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}}_{I}
\underbrace{\begin{array}{cccccccc}
1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1
\end{array}}_{X}\right]
\end{array}$$

Figure 4:Generator matrix for the (15, 7, 5) EG-LDPC code

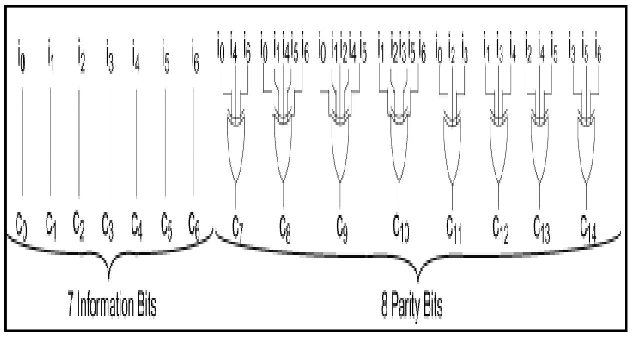
i_0 i_1 i_2 i_3 i_4 i_5 i_6
C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8 C_9 C_{10} C_{11} C_{12} C_{13} C_{14}
7 Information Bits
8 Parity Bits

Figure 5: Structure of an encoder circuit for the (15, 7, 5) EG-LDPC code

The information bits are fed into the encoder to encode the information vector, and the fault secure detector of the encoder verifies the validity of the encoded vector. If the detector detects any error, the encoding operation must be redone to generate the correct codeword. The codeword is then stored in the memory. During memory access operation, the stored codeword‘s will be accessed from the memory unit. Codeword‘s are susceptible to transient faults while they are stored in the memory.

3.2 Majority Logic Decoder and Detector Structure

The modified version of majority logic detector that overcomes the disadvantages of Majority logic decoder and syndrome vector with Majority logic decoder method. MLDD is straightforward, power decoder and capable of correcting several random bit-flips that depending in the number of the parity check equation.

In the MLDD method, the 15-bit codeword input is encoded and decoded. If codeword does not contain any error, then the output will be processed in three iterations. The advantages of this method are as follows

- Ability to correct large number of errors.
- Sparse encoding, decoding and checking circuits synthesizable into simple hardware
- Modular encoder and decoder blocks that allow an efficient hardware implementation
- Systematic code structure for clean partition of information and code bits in the memory.

In MLDD method, 15-bit codeword is used. There will be 15 iteration, if there is error in codeword. Otherwise the output will be obtained after three iteration. Schematic of

MLDD method shown in Fig 7 MLDD modules are as follows Cyclic shift register, XOR-matrix, Majority gate, Control unit.

The codeword is ‘n’ bit encoded block of bits. It contains information bits and parity bits. A block of ‘k’ bits are encoded to become a block of ‘n’ bits called codeword. The cyclic shift register with the output codeword of Most Significant Bit (MSB) is connected to Least Significant Bit (LSB) as codeword input. After the cyclic shift, all the codeword bits retain in the register, but their respective bit position changes. It circulates all codeword bits of the register around both MSB and LSB ends with no loss of information. The algorithm for MLDD method is to detect the error and correct. The flow chart for MLDD method as shown in Fig 7.

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. As stated before, input x might correspond to wrong data corrupted by a soft error. To handle this situation, the decoder would behave as follows. After the initial step, in which the codeword is loaded into the cyclic shift register, the decoding starts by calculating the parity check equations hardwired in the XOR matrix. The resulting sums {Bj} are then forwarded to the majority gate for evaluating its correctness. If the number of 1’s received in {Bj} is greater than the number of 0’s, that would mean that the current bit under decoding is wrong, and a signal to correct it would be triggered. Otherwise, the bit under decoding would be correct and no extra operations would be needed on it. In the next step, the content of the registers are rotated and the above procedure is repeated until all codeword bits have been processed. Instead of decoding all codeword bits by processing the ML decoding during N-cycles, the proposed method stops intermediately in the third cycle. Finally, the parity check sums should be zero if the N-codeword has been correctly decoded. The whole algorithm is depicted in Fig. 6.

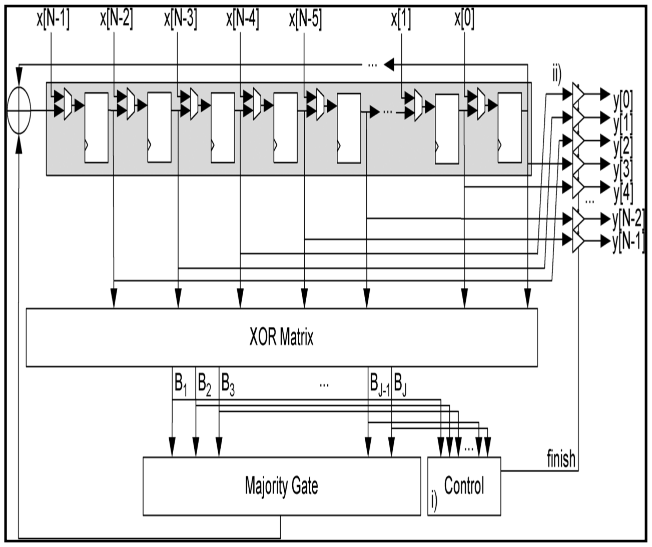

Figure 6: Schematic of the proposed MLDD for 15 bit codeword

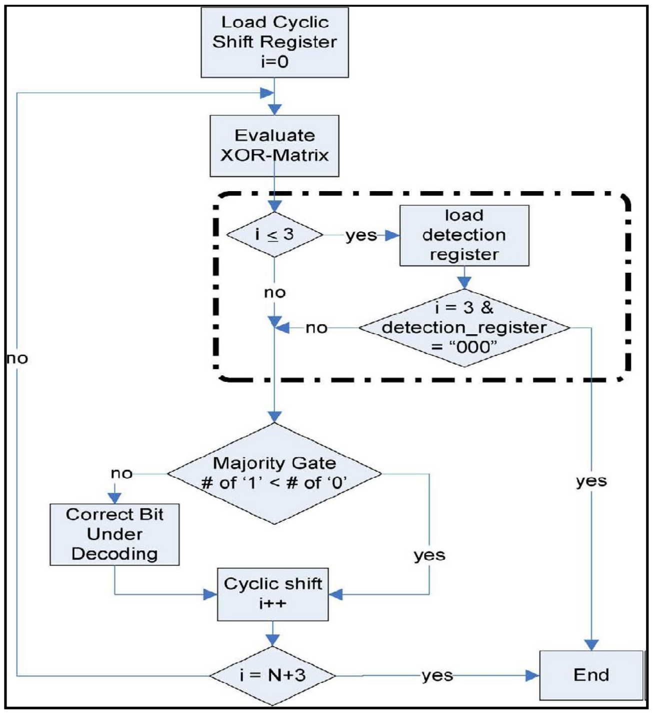
Load Cyclic Shift Register i=0
Evaluate XOR-Matrix
i ≤ 3
yes
load detection register
no
i = 3 & detection_register = "000"
no
no
yes
Majority Gate # of '1' < # of '0'
no
Correct Bit Under Decoding
yes
Cyclic shift i++
i = N+3
yes
End

Figure 7: Flow diagram of MLDD Algorithm

The control schematic is illustrated in Fig 8. The detection process is managed by the control unit. For distinguishing the first three iterations of the ML decoding, a counter unit evaluates the output from xor matrix Bj by giving it as input to the first OR gate. This out value is fed to two shift registers which has the results of the previous stages stored in it. The values are shifted accordingly. The third coming input is directly forwarded to the second or gate and finally all are evaluated in the third cycle in the OR2 gate. If the result is “0”, a finish signal is send by FSM which indicates that the processed word is error-free. The ML decoding process runs until the end, if the result is “1”. This clearly provides a performance improvement respect to the traditional method. Most of the words would only take three cycles (five, if we consider the other two for input/output) and only those with errors (which should be a minority) would need to perform the whole decoding process.

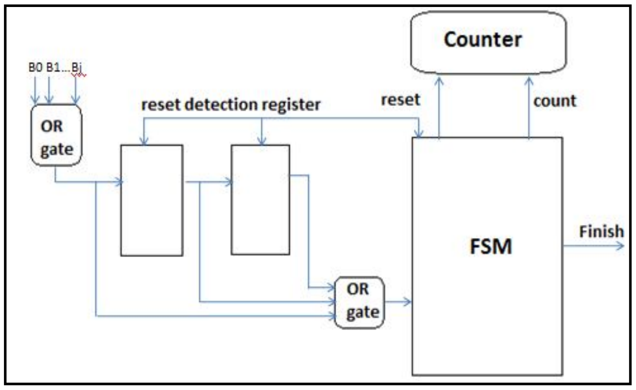

Figure 8: Schematic of MLDD Control unit

4.SIMULATION RESULTS

The existence decoding method can be used to reduce the decoding latency when no error has been detected in the first three iterations. This technique reduces the decoding latency significantly as most words will have no errors. But when the error detects at the t+1 affected bits it cannot be corrected the result is shown in the below Fig 9.

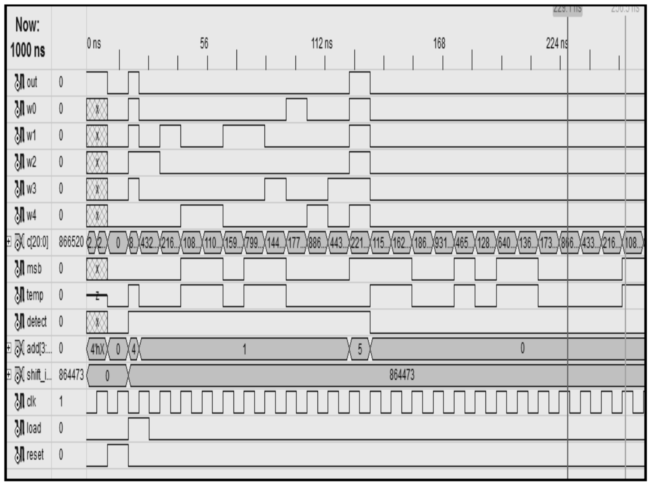

Figure 9: Uncorrected error

The simulation results of the proposed MLDD of Encoder and Control unit is shown in below Fig 10 and Fig 11 respectively.

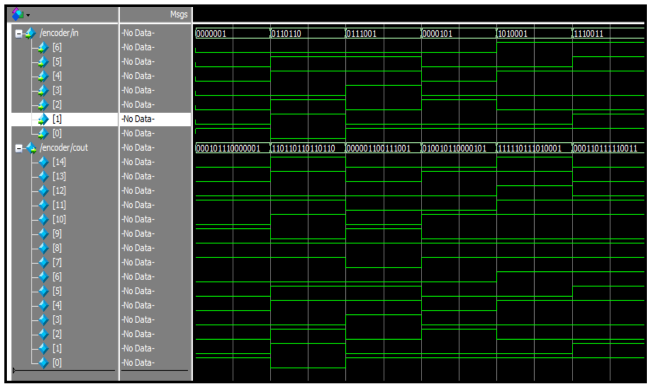

Figure 10: Simulation result for proposed encoder

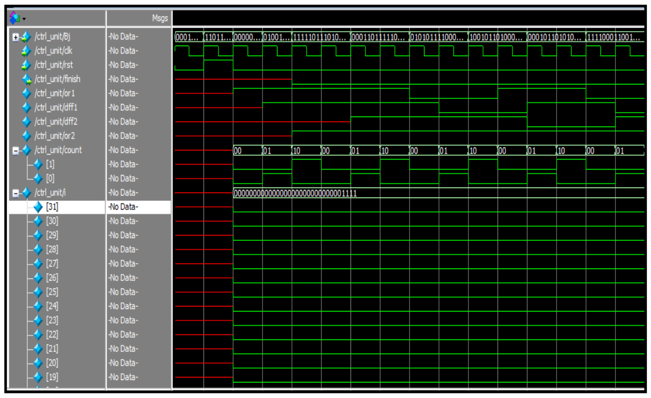

Figure 11:Simulation result for proposed MLDD control unit

5.Application

The potential applications areas of proposed MLDD include the following

- Nano memory
- Wireless communication systems
- Internet
- Deep space telecommunications
- Satellite broadcasting
- Data storage

6.Conclusion And Future Work

In this paper, a fault-detection mechanism, MLDD, has been presented based on ML decoding using the EG-LDPC codes. The simulation test results show that the proposed technique is able to detect any pattern of up to five bit-flips in the first three cycles of the decoding process. This improves the performance of the design with respect to the traditional MLD approach. The performances of the proposed MLDD method is faster when compared to existing method MLD and MLD with syndrome vector. On the other hand, the MLDD error detector module has been designed in a way that is independent of the code size. This makes its area overhead quite reduced compared with other traditional approaches such as the syndrome calculation (SFD).

The proposed detects the faults in just three cycles. Therefore a large number of clock cycles are saved and hence considerable reduction in power is achieved. In future we can implement the proposed conventional logic gates can be replaced by reversible logic gate in order to reduce the present power consumption in conventional logic and the size of message bits and codeword can be increase further like (73, 36, 19), since LDPC codes reaches the Shannon limit and codeword consists of less number of one’s.

7.Reference

1. R.C. Baumann, “Radiation-induced soft errors in advanced semiconductor technologies,” IEEE Trans. Device Mater. Reliabil., vol.5, no3, pp. 301-316, sep. 2005.
2. C.W. Slayman, “Cache and Memory error Detection, correction, and reduction techniques for terrestrial servers and workstations,” IEEE Trans. Device Mater. Reliabil., vol.5, no3, pp. 301-316, sep. 2005.
3. C. W. Slayman, “Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations,” IEEE Trans. Device Mater. Reliabil., vol. 5, no. 3, pp. 397–404, Sep. 2005.
4. "Re: Longitude as a Romance". Irbs.com, Navigation mailing list. 2001-07-12. Retrieved 2009-02-16.
5. R. Naseer and J. Draper, “ DEC ECC design to improve memory reliability in sub-100nm technologies,” in Proc IEEE ICECS, 2008, pp.586-586.
6. H. Naeimi and A. DeHon, “Fault secure encoder and decoder for Nano Memory applications,” IEEE Trans. Device Mater. Reliabil., vol. 5, no. 3, pp. 397-404, sep. 2005.
7. I. S. Reed, “A class of multiple-error-correcting codes and the decoding scheme,” IRE Trans. Inf. Theory, vol. IT-4, pp. 38–49, 1954.
8. S. Lin and D. J. Costello, Error Control Coding, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004.
9. J. L. Massey, Threshold Decoding. Cambridge, MA: MIT Press, 1963.
10. P. Ankolekar, S. Rosner, R. Isaac, and J. Bredow, “Multi-bit error correction methods for latency-contrained flash memory systems,” IEEE
11. Trans. Device Mater. Reliabil., vol. 10, no. 1, pp. 33–39, Mar. 2010.
12. Shih-fu Liu,pedro Revingo, and Juan Antonio Meastro “Efficient majority fault detection with difference set codes for memory applications”, IEEE Trans. Very Large Scale Integr.(VLSI)Syst., vol.20, no. 1, pp.148-156, Jan.2012
13. 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 Trans On Very Scale Integration (Vlsi) system 1
14. Y. Kato and T. Morita, “Error correction circuit using difference-set cyclic code,” in Proc. ASP-DAC, 2003, pp. 585–586.

15. D.Subalakshmi and Major. P.S.Raghavendran "Error Identification and Correction for Memory Application using Majority Logic Decoder and Detector," in International Journal of Computer Applications (0975 – 8887) Volume 64– No.10, Feb 2013
16. J. A. Maestro and P. Reviriego, " Reliability of soft-error correction protected memories", IEEE Trans, on Reliability, vol. 58, no. 1, pp. 193-201, Mar. 2009.