FAULT DETECTION IN MEMORY APPLICATION USING MAJORITY LOGIC DECODER/DETECTOR

Jothi Maragathavalli. R,*

Prof.V. Gopi, M.E, (Ph.d)**

Abstract-

In recent years, memory cells have become more liable to soft errors. This must be protected with effective error correction codes. Majority logic decodable codes are suitable for memory applications because of their capability to correct large number of errors. Conversely, they increase the average latency of the decoding process because it depends upon the code size that impacts memory performance. Another method of decodable logic is Majority Logic Decoder/Detector which reduces the decoding time, memory access time and area utilization .The drawback of this method is that it does not detects the silent data error and it consumes the major area of the majority gate. Low-Density Parity-Check (LDPC) codes are used for error correction, because of their fault-secure detector capability. In this paper, an enhanced majority logic decoder/detector (MLDD) is proposed to detect silent data error (SDE) using additional logic and in order to reduce the area of the majority gate, by using sorting network. Thus, the proposed Method reduces the decoding time, area and also power consumption.

Index terms - Error correction codes, Soft errors, Memory, Majority logic decoder/detector, Low-density parity check (LDPC) codes, Silent Data Error (SDE), Sorting network.

* PG Scholar/Department of ECE, PSN College of Engg. and Technology, Tirunelveli, India.

** The Dean ECE, PSN College of Engg. and Technology, Tirunelveli, India.

I. INTRODUCTION

The reliability and security of memories are essential considerations in the modern digital system design. Soft error occurs when a radiation event causes enough of a charge disturbance to Reverse or flip the data state of a memory cell, register [2]. As technology scales, memory devices become larger and more powerful error correction codes are needed to protect memories from soft errors [3], [4]. The error is "soft" because it will change the logic value of memory cells without damaging the circuit/device. The soft error is also referred to as a Single Event Upset (SEU). If the radiation event is of high energy, more than a single bit may be affected, creating a Multi Bit Upset (MBU) [2].

For consistent communication, errors must be detected and corrected. Some multi error bit correction codes are BCH codes, Reed Solomon codes, but in which the algorithm is very difficult. These codes can correct a large number of errors, but need complex decoders [10], [11]. Among the error correction codes, cyclic block codes have higher error detection capability, low decoding complexity and that are majority logic (ML) decodable. A low-density parity-check (LDPC) code is a linear error correcting code, used to avoid a high decoding complexity [6]-[9] In this paper, one specific type of low density parity check [LDPC] codes, [1] are used due to their fault secure detector capability, higher reliability and lower area overhead. The LDPC codes which are one step majority logic decodable. Various error detection techniques are used to avoid the soft error [10]. One of the methods is majority logic decoder which used to detect and correct the error in simple way. This method uses the first iteration of majority logic decoding to detect the error present in the word. If there are no errors, then the decoding process can be stopped without completing the remaining iterations [1]. The main reason for using Majority Logic Decoding (MLD) is that it is very easy to implement and has a low complexity [11].

The parallel encoders and decoders have been implemented to overcome the drawback of majority logic decoder in which it takes N number of cycles to detect the error [11]. In this paper, the Majority Logic Decoder/Detector (MLDD) method [11] used to detect the error in memory device itself, so the data corruption during processing has been eliminated easily to improve the system performance. The MLDD is used the control unit for detecting the error. This method did not detect the silent data error [12].

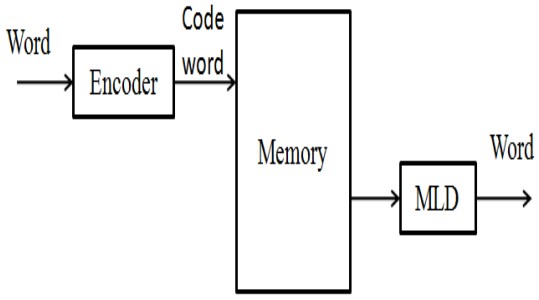

Fig 1: General memory system with MLD

The general schematic of memory system implemented with majority logic decoder is depicted in figure 1. Initially the data words are encoded and then stored in the memory. When the memory is read, the code word is then fed through the Majority Logic Decoder (MLD) before sent to the output. In this decoding process, the code word is corrected from all bit flips it might have suffered while being stored in the memory. The proposed enhanced MLDD method uses additional error detection technique to detect the silent data error (SDE) in MLDD. To produce the accurate result of MLDD, this addition logic is used to detect the error which is not detected by the first three iteration of the MLDD. To reduce the number of gates in the majority gate, a sorting network is used. Thus reduces the area of the majority gate and also the power consumption. The remainder of this paper is organised as follows. Section II gives an overview of existing ML decoding and MLDD system; Section III presents the novel enhanced ML decoder/detector (MLDD) using Low Density Parity Check Codes (LDPC); Section IV discusses the results obtained for the different methods in respect to the performance, area and power consumption; Finally, Section V discusses Conclusions.

II. EXISTING SYSTEM

This section deals with the existing decoding methodologies used for error detection. In error detection and correction, majority logic decoding is a method to decode repetition codes, based on the assumption that the largest number of occurrences of a symbol was the transmitted symbol. Majority logic decoder is based on a number of parity check equations which are orthogonal to each other [11]. So the majority result of these parity check equations decide the correctness of the current bit under decoding.

A. One Step Majority Logic Decoder

As described in previous, Majority-logic decoder is a simple and effective decoder capable of correcting multiple bit flips depending on the number of parity checksum equations. It consists of four parts: 1) a cyclic shift register; 2) an XOR matrix; 3) a majority gate; 4) an EXOR gate for error correction, as illustrated in figure 2. Fig 2:

In one step majority logic decoding [1], 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.

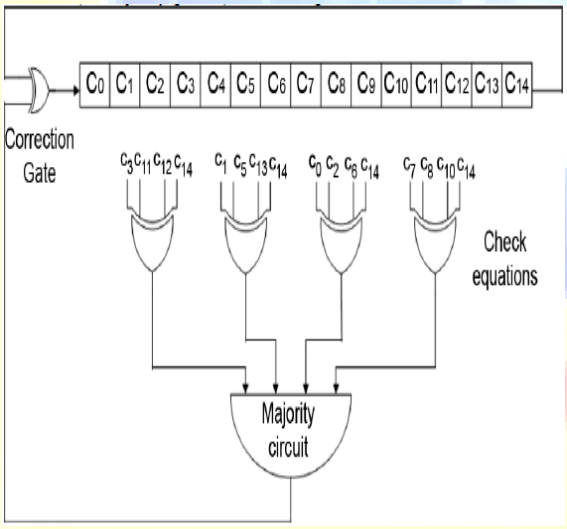

Fig 2: One step Majority Logic Decoder for (15, 7) LDPC Codes

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.

B. Majority Logic Decoder/Detector (MLDD)

In order to overcome the drawback of MLD method, majority logic decoder/detector was proposed, in which the majority logic decoder itself act as a fault detector. In general, the decoding algorithm is still the same as the majority logic decoder. The difference is that instead of decoding all codeword bits, the

MLDD method 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. In conclusion the MLDD method is used to detect the five bit errors and correct four bit errors effectively. If the code word contain more than five bit error, it produces the output but it did not show the errors presented in the input. This type of error is called the silent data error.

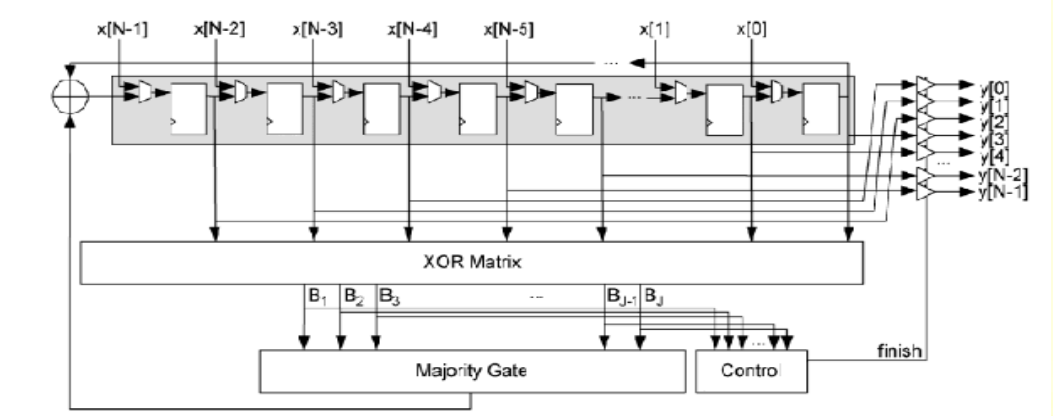

Fig 3: Schematic of Majority Logic Decoder/Detector (MLDD)

Drawback of this method is did not detecting the silent data error and it consuming the area of the majority gate. The schematic for this memory system is shown in figure 5. It is very similar to the one shown in fig. 1; additionally the control unit was added in the MLDD module

to manage the decoding process (to detect the error). Overall operation of the MLDD is illustrated in figure 3.

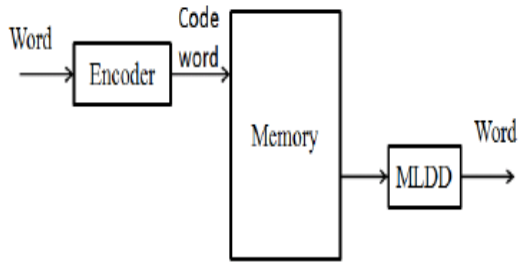

Fig 4: Schematic of memory with MLDD

III. PROPOSED SYSTEM (ENHANCED MLDD)

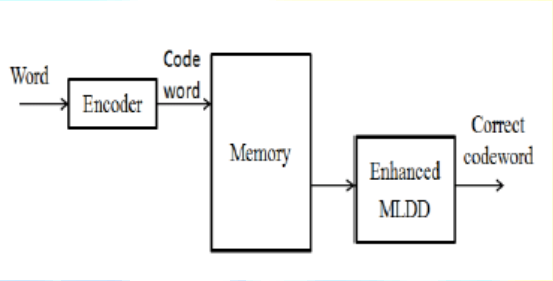

Fig 5: Memory schematic of an Enhanced MLDD

This section presents an enhanced version of the ML decoder/detector that improves the designs presented before, by

detecting the silent data error. Memory schematic of an enhanced MLDD is illustrated in fig 5.The silent data error detection using enhanced MLDD algorithm performs the decoding as in the MLDD with some modifications. When the MLDD having more than 5 errors will be detected and corrected by the enhanced MLDD method. The MLDD is used the control unit for detecting the error. If it has any error in this iteration it will be perform with the modified algorithm is illustrated in Figure 7. It is used to avoid silent data corruption of the MLDD output. This would increase the error detection capabilities at the expense of the error-correction capabilities. In this algorithm up to four errors will be done as in the MLDD algorithm. If it has more than four errors will detected by after third iteration. Then correction will be done by after nth iteration.

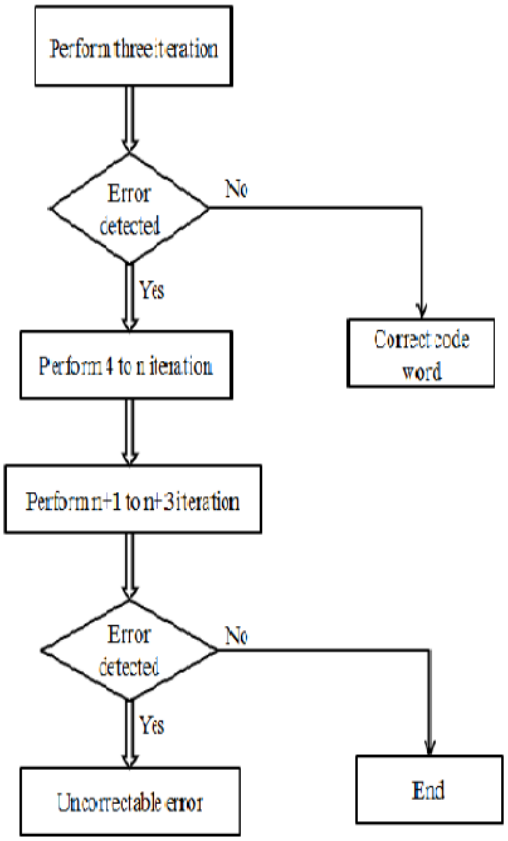

Fig 7: Enhanced MLDD algorithm

A. Sorting network

A sorting network is an abstract mathematical model of a network of wires and comparator modules that is used to sort a sequence of numbers. Each comparator connects two wires and sorts the values by outputting the smaller value to one wire, and the larger to the other. The main difference between sorting networks and comparison sorting algorithms is that with a sorting network the sequence of comparisons is set in advance, regardless of the outcome of previous comparisons. This independence comparison sequences is useful for parallel execution of the algorithms.

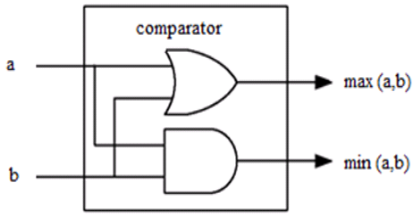

Fig 8(a): Comparator circuit

A sorting network consists of wires and comparators that will correctly sort all possible inputs into ascending order. . So it used to reducing the gates and their interconnections of the majority gate. Each wire carries with it a value, and each comparator takes two wires as input and output. When two values enter a comparator, the comparator emits the lower value from the top wire, and the higher value from the bottom wire. Using sorting network number of gates reduced in the majority gate. Initially it compares the inputs using comparator circuit. Comparator consist of AND gate and then OR gate for selecting maximum and minimum value shown in figure 8 (a). OR gate producing maximum value will be placed in top of the wire and the AND gate producing minimum value will be placed in bottom of the wire in the comparator circuit.

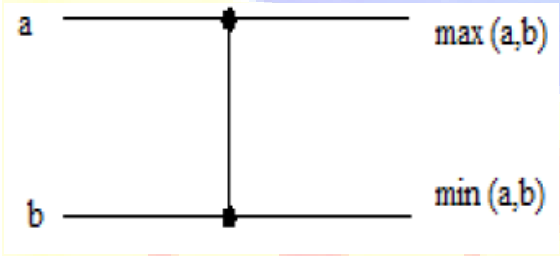

Fig 8(b): 2-bit sorter

Each of the vertical lines represents one comparator which compares two bits and assigns the larger one to the top output and the smaller one to the bottom. Those value given to the AND gate for getting the minimum value and given to the OR gate for selecting the maximum value. Shown in figure 8 (b).

IV. RESULTS AND DISCUSSIONS

The design is simulated using modelsim and Xilinx. The power consumed and area utilization is calculated. The enhanced MLDD with sorting network shows reduced power consumption and area compared existing method. Then the performance of MLDD and enhanced MLDD has shown below. Using modelsim shows the output waveform of MLDD and enhanced MLDD with and without silent data error and then the memory access time of both MLDD and enhanced MLDD were shown. Using Xilinx get the power consumption and area utilization of both MLDD and enhanced MLDD.

TABLE 1: PERFORMANCE RESULTS FOR MLDD AND ENHANCED MLDD

MODEL	NO OF CYCLES FOR I/O	NO OF CYCLES FOR ERROR DETECTION	NO OF CYCLES FOR NO ERROR	NO OF CYCLES FOR ERROR PRESENTED	NO OF CYCLES FOR SILENT DATA ERROR DETECTION
MLDD	2	3	5	78	Not detected
ENHANCED MLDD	2	3	5	78	78

TABLE 2: COMPARISONS OF POWER AND AREA UTILIZATION

Design	Power (in watts)	Area (in number of slices)
MLDD	1.776	131
Enhanced MLDD using sorting network	0.823	115

V. CONCLUSION

The enhanced MLDD technique is used to detect any pattern of more than five bit-flips is designed. The comparison between the MLDD to detect any pattern up to five bit-flips and enhanced MLDD to detect more than five bit-flips were done. They can be simulated by using modelsim 6.4a and Xilinx 13.2 software and their performances were analyzed. Thus the performance of the proposed method was improved compared to the MLDD approach. Finally, the decoding cycles for the detection and correction of errors is slightly increased comparing to MLDD approach. The area and power consumed by enhanced MLDD is less compared to the MLDD approach by the use of sorting network.

REFERENCES

[1] 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. Very Large Scale Integration (VLSI) Systems, Vol. 21, No. 1, January 2013.

[2] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," IEEE Trans. Device Mater. Reliab., vol. 5, no. 3, pp. 301–316, Sep. 2005.

[3] M. A. Bajura, Y. Boulghassoul, R. Naseer, S.DasGupta, A. F.Witulski, J. Sondeen, S. D. Stansberry, J. Draper, L. W. Massengill, and J. N.Damoulakis, "Models and algorithmic limits for an ECC-based approach to hardening sub-100-nm

[4] SRAMs," IEEE Trans. Nucl. Sci., vol. 54, no. 4, pp. 935–945, Aug. 2007.

[5] R. Naseer and J. Draper, "DEC ECC design to improve memory reliability in sub-100 nm Technologies," Proc. IEEE ICECS, pp. 586–589, 2008.

[6] S. Ghosh and P. D. Lincoln, "Dynamic low-density parity check codes for fault-tolerant nano-scale memory," presented at the Foundations Nanosci. (FNANO), Snowbird, Utah, 2007.

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

[8] H. Naeimi and A. DeHon, "Fault secure encoder and decoder for memory applications," in Proc. IEEE Int. Symp. Defect Fault Toler. VLSI Syst., 2007, pp. 409–417.

[9] B. Vasic and S. K. Chilappagari, information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low- density parity-check codes," IEEE Trans. CircuitsSyst. I, Reg. Papers, vol. 54, no. 11, Nov. 2007.

[10] 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, pp. 473–486, Apr. 2009.

[11] S. Lin and D. J. Costello, Error Control Coding, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004.

[12] S. Liu, P. Reviriego, and J. Maestro, "Efficient majority logic 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] H. Tang, J. Xu, S. Lin, and K. A. S. Abdel-Ghaffar, "Codes on finite geometries," IEEE Trans. Inf. Theory, vol. 51, no. 2, pp. 572–596, Feb.2005.