ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Design and Implementation of LDPC for Memory Applications

Gopagoni Kiran
M.Tech (VLSI Design)
Arjun College of Technology & Sciences.

Mr.Lingaiah, M.E
Associate Professor & HoD
Arjun College of Technology & Sciences.

ABSTRACT

In the Advanced digital system design the security and reliability of Memory system are essential consideration. As a result of Technology, the memory devices becoming larger and we need more powerful error detection and correction codes to protect memories from soft errors. The paper mainly focused on the design of efficient Majority Logic Detector/Decoder (MLDD) for fault detection along with correction of fault for memory application, by considerably reducing fault detection time. The error detection and correction method is done by one step majority logic decoding and it is made of effective for Low Density Parity Check Codes (LDPC). Even though majority decodable codes can correct large number of errors, they need high decoding time for detection of errors and Majority Logic Decoding method may take same fault detecting time for both erroneous and error free code words, which in turn delays the memory performance. The proposed fault-detection method can detect the large errors in less decoding cycles (almost in three). When the data is error free, it can obviously reduce memory access time. The Technique keeps the area overhead minimal and low power consumption for large code word sizes.

INTRODUCTION

Now a days, digital communication has becoming essential part of life and a lot of data is being transferred. Many communication channels are leads to channel noise. Networks must be able to transfer data from one device to another with acceptable accuracy. For most applications, a system must guarantee that the data received are identical to the data transmitted. 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 major drawback of this method is increase the average latency of the decoding process because it depends on the size of the code, thus increases the memory access time. Another method is syndrome fault detector [11] which is an XOR matrix that calculates the syndrome based on the parity check matrix. This method results in a quite complex module, with a large amount of hardware and power consumption in the system. 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

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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].

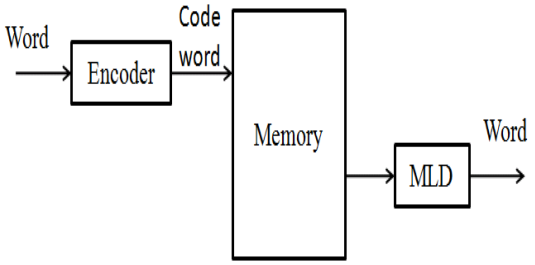


Fig 1: General memory system with MLD

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 Any time data are transmitted from one node to the next, they can become corrupted in passage. Many factors can alter one or more bits of message. Some application required a mechanism for detecting and correcting large number of errors. Data can be corrupted during transmission. Some application are required that errors can be detected and corrected. There are different methods to detect and correct errors to keep secure and accuracy data communication. Some applications can tolerate a small level of errors. For example, random errors in audio or video transmission may be tolerable, but when transfer text; we expect a very high level of accuracy.

Type of Errors

The error is soft because it will change the logic value of memory cells without damaging the circuit/device. The soft errors referred as a Single Event Upset (SEU).If the radiation event is of high energy, more than a single bit may be affected, i.e. Multi Bit Upset (MBU) [2].

For reliable communication, errors must be detected and corrected. Some multi error bit correction codes are BCH codes, Reed Solomon codes but in which 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 linear error correcting code, used to avoid a high decoding complexity [6]-[9]. In this paper, one specific type of low density parity check codes, namely LDPC codes [1] are used due to their 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 organized as follows. Section II gives an overview of existing ML decoding and MLDD system; Section III presents the novel improved ML decoder/detector (MLDD) using Low Density Parity Check Codes (EG-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.

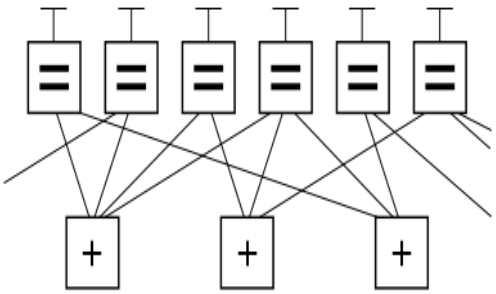
INTRODUCTION TO LDPC CODES

LDPC codes are defined by a sparse parity-check matrix. This sparse matrix is often randomly generated, subject to these parity constraints LDPC code construction is discussed later. These codes were first designed by Robert Gallager in 1962. Below is a graph fragment of an example LDPC code using Forney's factor graph notation. In this graph, *n* variable nodes in the top of the graph are

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

connected to ($n-k$) constraint nodes in the bottom of the graph. This is a popular way of graphically representing an (n, k) LDPC code. The bits of a valid message, when placed on the **T's** at the top of the graph, satisfy the graphical constraints. Specifically, all lines connecting to a variable node (box with an '=' sign) have the same value, and all values connecting to a factor node (box with a '+' sign) must sum, modulo two, to zero (in other words, they must sum to an even number).

Ignoring any lines going out of the picture, there are eight possible six-bit strings corresponding to valid codeword's: (i.e., 000000, 011001, 110010, 101011, 111100, 100101, 001110, 010111). This LDPC code fragment represents a three-bit message encoded as six bits. Redundancy is used, here, to increase the chance of recovering from channel errors. This is a (6, 3) linear code, with $n = 6$ and $k = 3$. Once again ignoring lines going out of the picture, the parity-check matrix representing this graph fragment is show in below.



$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Fig 2: Parity Check Matrix

In this matrix, each row represents one of the three parity-check constraints, while each column represents one of the six bits in the received code word. In this example, the eight codeword's can be obtained by putting the parity-check matrix **H** into this form $[-P^T | I_{n-k}]$ through basic row operations:

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.

One Step Majority Logic Decoder

As described in earlier, 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.

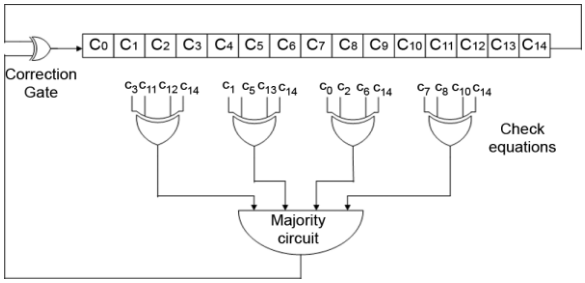


Fig 3: One step Majority Decoder for (15,7) LDPC codes

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

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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.

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, which can able to detect up to five bit flips in three decoding cycles. So the number of decoding cycles can be reduced to get improved performance. The schematic of majority logic decoder/detector is illustrated in figure3.

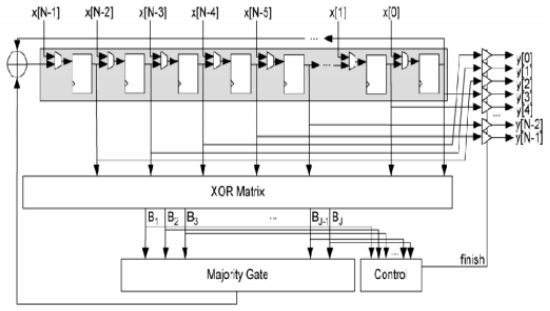


Fig 4: Schematic of Majority Logic Decoder / Detector (MLDD)

Initially the code word 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 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. 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 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. 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. 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).

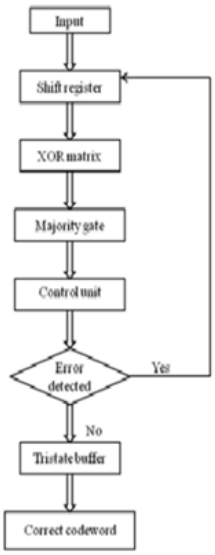


Fig 6 : MLDD Algorithm

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

PROPOSED SYSTEM (ENHANCED MLDD)

In general, the proposed version uses the same decoding algorithm as the one in plain ML decoder version. The advantage is that, proposed method stops intermediately in the third cycle when there is no error in data read, [2] as illustrated in Figure. 6, instead of decoding it for the whole codeword size of N. The xor matrix is evaluated for the first three cycles of the decoding process, and when all the outputs {Bj} is "0,"the codeword is determined to be error-free and forwarded directly to the output. On other hand, the proposed method would continue the whole decoding process to eliminate the errors [2] if the {Bj} contain at least a "1" in any of the three cycles

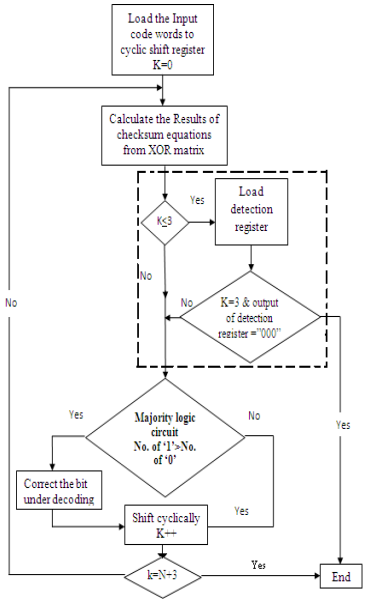


Fig 7: A detailed schematic of the proposed design for 15 bit code word is shown in Figure 7.

A detailed schematic of the proposed design for 15 bit code word is shown in Figure 7. The figure shows the basic ML decoder with a 15-tap shift register, an XOR array to calculate the orthogonal parity check sums and a majority logic circuit which will decide whether the current bit under decoding is erroneous and the need for its inversion.

The plain ML decoder [2] shown in Figure 1 is also having the same schematic structure up to this stage. The additional hardware [2] intended for fault detection illustrated in Figure 8 are: a) the control logic unit and b) the output tristate buffers. The control unit triggers a finish flag when there is no errors are detected in data read. The output tristate buffers are always in high impedance state until the control unit sends the finish signal so that the current values are forwarded to the output y from the shift register.

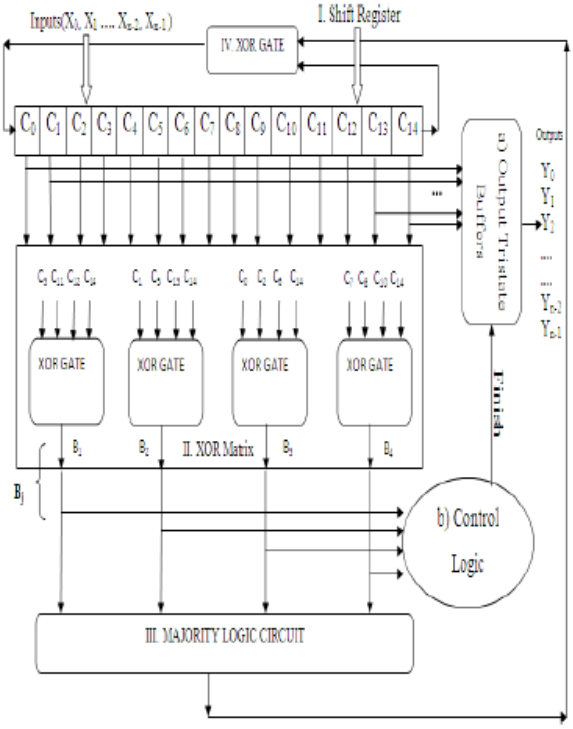


Fig 8: Schematic of Control Logic

The control logic schematic [2] is illustrated in Figure 9. The detection process is managed by the control unit. For distinguishing the first three iterations of the ML decoding, a counter is used here which counts up to three cycles. The control unit evaluates the output from xor matrix Bj by giving it as input to the OR 1 gate. This output 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 OR 2 gate and finally all are evaluated in the third cycle in the OR 2 gate. If the result is "0," a finish signal is send by the FSM which indicates that the processed word is error-free. The ML decoding process runs until the end, if the result is "1".

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

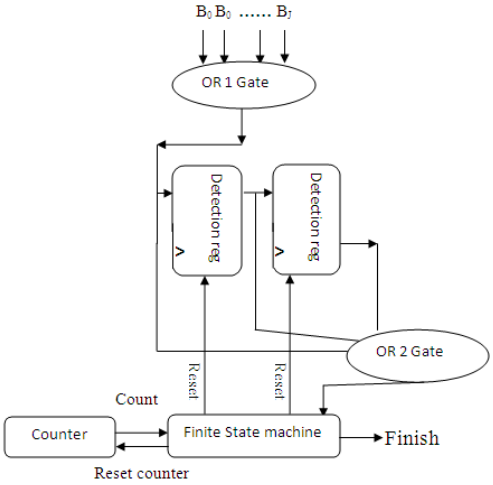


Fig 9: Schematic of the Control Unit

MODIFIED MLDD

Since we are using a separate module for fault detection, there will be a slight area overhead. This area overhead can be overcome by using sorting network in the majority gate. The EG LDPC code used here is only for 15 bits, it have only outputs four outputs from xor matrix. Therefore the above structure of sorting network in Figure 10 (a) can be used. It takes only four input bits and the vertical lines shown here is comparator Figure 10 (b) which has two inputs and it will compare and larger bit is given to the top output and smaller to bottom respectively.

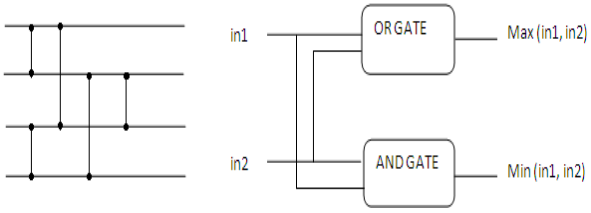


Fig10 : (a)Sorting network-four input (b) Schematic of one comparator

This clearly provides a performance improvement respect to the traditional method which is the existing MLD. [2]. The proposed method mostly would only take three cycles for decoding(five, if we consider the other two for input/output) since most of the words would be error free and would need to perform the whole decoding process only for those words with errors (which should be a minority).

RESULTS AND DISCUSSION

The UART BIST architecture simulation was done through the Xilinx ISE using VERILOG HDL. The data address-bit verification can also to be done through this simulation and the waveform could be verified by using the XILINX.

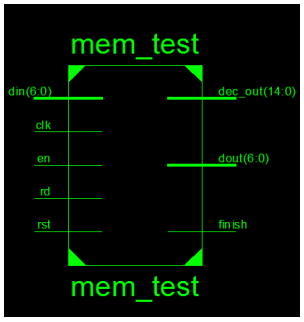


Fig 11: Block diagram for proposed

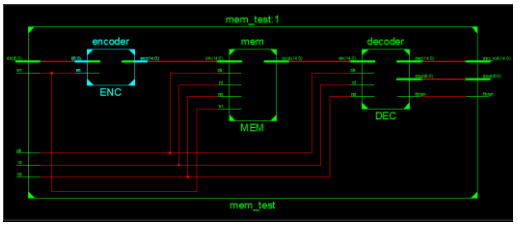


Fig 12: RTL for Proposed

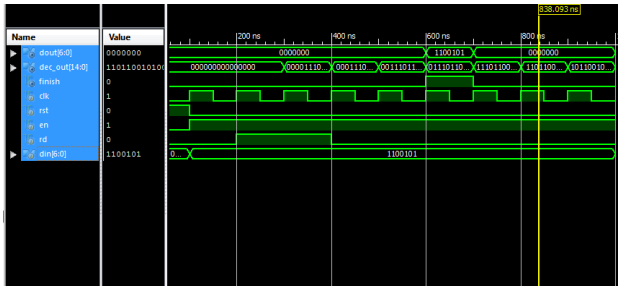


Fig 13: Waveform for proposed

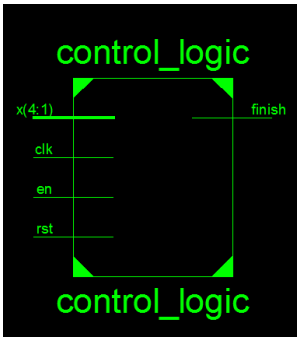


Fig 14: Block diagram for Control logic

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

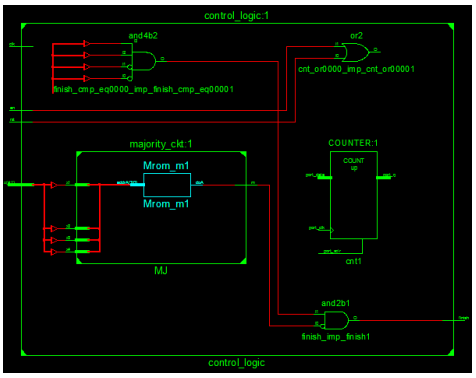


Fig 15: RTL for Control logic

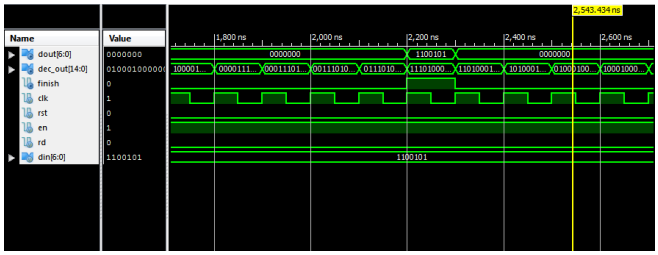


Fig 16: Waveform for Control logic

CONCLUSION

This papers presents the UART based BIST Architecture using VERILOG HDL. Most of the researchers have been used to implement this testing algorithm in VERILOG for stable, compact and reliable transmission. The structural details have been recognized and it can be integrated into the chip could be easier. The UART transmission could be relatively used in all the devices for the reliable transmission of data's from the structure where it could be converter and tested as a bit files generation. This design function can be adopted as a technical preserving data's for communication. The BIST controller as a device uses as an efficient bit generation for the chip implementation.

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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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