THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

Efficient Design of VLSI Architecture for Error Correction by using ML Decoder/Detector

S. N. Lalitha Parameswari
PG/VLSI Design, Karpagam University, Coimbatore, India
G. R. Mahendra Babu
Assistant Professor, Department of ECE, Karpagam University, Coimbatore, India

Abstract:
Error correction codes protect the memories from soft errors which cause data corruption. When additional protection is needed an advanced error correction codes are utilized. For correcting large number of soft errors, reduce decoding time and area consumption, majority logic decoder/detector codes are used. This paper presents an error detection method using majority logic decoding methodology for Euclidean Geometry Low Density Parity Check codes. The codes are synthesized using Xilinx 8.1 and Modelsim 6.3g.The proposed improved majority logic detector/decoder to perform silent data error correction in simple way using additional error correction technique and also reducing the delay time by detecting the errors in parallel and pipelining manner. Hence the decoding process uses less number of cycles which reduces the delay and also reduces the power consumption.

Key words*: Error correction codes, majority logic decoding, pipelining, low density parity check codes (LDPC), memory*

1. Introduction

Error correction codes are used to protect memories from soft errors. Soft errors will change the logical value of memory cells, but without damaging the circuit [1]. As technology emerges, memory devices become larger. It requires powerful error correction codes to correct the errors [2], [3]. Due to this the use of more advanced codes has been recently proposed [4]–[8]. These types of codes can correct a larger number of errors, but they require mixed decoders. To avoid this difficult decoding method, the different set codes and one step majority logic decodable codes were first proposed in [10], [1] for memory applications, decoding of low density parity check codes is done using majority logic decoding [9].

Further work on this topic was then presented in [5], [6], [8]. One step majority logic decoding can be implemented serially with very simple circuit [9], but requires long decoding times and it increases the delay. In a memory, this would increase the access time which is an important system constraint. But for few modules of codes are decoded using one step majority decoding [1].

Among those are some Euclidean geometry low density parity check (EG-LDPC) codes which were used in [4].A method was recently proposed to accelerate a serial implementation of majority logic decoding of EG-LDPC codes. The design behind the method is to use the first iterations of majority logic decoding to detect if the word decoded contains errors. If it is found there are no errors, then decoding process can be stopped. Decoding time is much more reduced because of stopping the iterations before fully completing. For a code with block length N, majority logic decoding which is implemented serially requires N iterations, so that the sizes of the code increase, so the decoding time also increase. In the proposed system, the errors are detected in parallel and in pipelining method. The detection of errors requires only single iteration where most of the errors are detected. The delay time is reduced for this proposed method is low compared to the prior technique.

2. Existing MLD Techniques

Majority logic decoding is a simple and effective scheme for decoding certain class of block codes. Majority logic decoding codes are cyclic codes. These codes are constructed based on finite geometrics such as Euclidean geometrics and projective geometrics.

Majority Logic Decoder (MLD) technique is generally based on number of parity check equations. A generic algorithm of this memory system is exposed in Fig .1. Initially the input is stored into the cyclic shift register and it shifted through all the bits. The intermediate values in each bit are given to the XOR matrix which is used to perform the checksum equations. The resulting sums are then forwarded to the majority gate for evaluating its correctness. If the number of 1's received is greater than the number of 0's which would mean that the current bit under decoding is wrong, so it move on the decoding process. It is used to produce the accurate result of the MLD. Otherwise, the bit under decoding would be correct and no extra operations would be needed on it and it is stored in the tri state buffer.

Decoding process involving the operation of the content of the registers is rotated and the above procedure is repeated till third cycle and it stops intermediate. For first three cycles of the decoding process, the estimate of the XOR matrix for all is '0', the

code word is determined to be error-free and forwarded to the output straightly. 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. At last the parity check sums should be zero if the code word has been correctly decoded. Finally the MLD method is used to detect the five bit errors and correct four bit errors effectively. More than five bit errors it produces the output but it did not show the errors presented in the input. This type of error is called the hushed data error. Drawback of this method is it is not detecting the silent data error and it consuming the area of the majority gate.

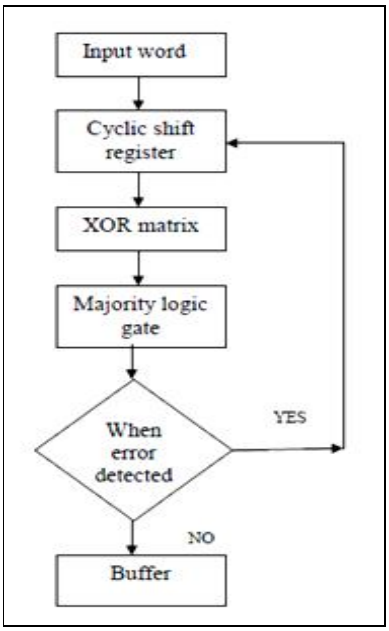

Figure 1: Flow chart of simple Majority Logic Decoding algorithm

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 type of code uses the check sum algorithm. The check sum algorithm is nothing but a numerical value is associated with the code word which is to be transmitted. At the receiver end the code word received has some numerical value. Then there is a comparison on the associated numerical values to detect the error.
The existing method is implemented using basic hardware. The decoding time is large for this serial methodology. Also the power consumption and area requirement are high. The MLD technique uses Serial One Step Majority Logic Decoder is used to detect the errors serially. The serial one step majority logic decoder algorithm for error detection and correction is exposed in Fig.2.

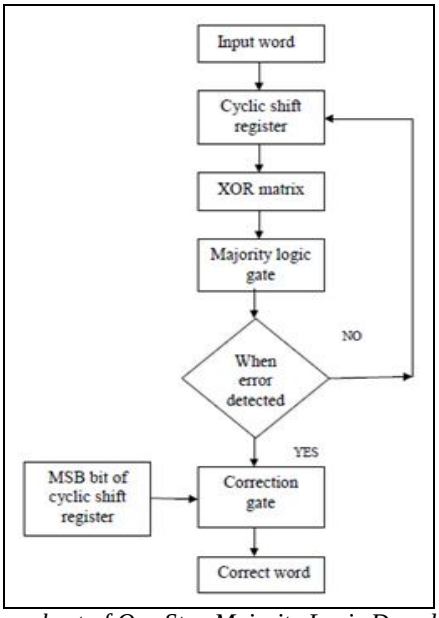

Figure 2: Flow chart of One Step Majority Logic Decoding algorithm

In this decoder 15 bit data is first stored in the cyclic shift register. Then the inputs are given to the XOR gates. The XOR gates required are four because the input is a 15 bit data. The bit to be detected should be given as one of the inputs for all the XOR gates. The XOR gates outputs are the check sum equations with some numerical data's. The check sum equations consist of 0's and 1's that are binary datas. Then the Majority circuit outputs the data which is in majority number of 1's. If the output of the one step majority circuit is majority number of '1' then the corresponding bit has the error else the bit is error free.
The output of the Majority circuit is given as one of the input to the correction gate. The bit which is under test is the other input to the correction gate. The corrected bit is stored into the shift register after the first cyclic shift. The entire process is called single iteration. Similarly three iterations are processed. First three iterations are required to detect all the errors of any number. Serial methodology requires three iterations to detect all possible errors. The parallel MLDD is used to overcome this drawback of serial MLDD which is the proposed technique. The errors of any number can be detected in the single iteration.

3. Area, Power and Delay Analysis
The comparison for area, power and delay for the existing serial one step MLDD is shown in Table I.

Design	Area	Power (mw)	Delay (ns)
Serial MLDD	1090	116.2	9.855

Table 1: Comparison for Area, Power and Delay

4. Proposed Majority Logic Detector/Decoding Technique
In this proposed technique the error detection process is done in parallel and in pipelining manner.

4.1. Parallel Processing
Single iteration is required for detection of any number of errors. Thus the delay time is reasonably low. Also the power consumption and the area requirement are low. The entire error in any bit of the given data is detected simultaneously in single iteration. But there is no cyclic shift as in serial error detection process. The logic blocks are same for the parallel MLDD as in the serial MLDD. But required is more number of majority gates and correction gates, each gate is assigned for a single bit. Error detection process is also done in pipelining manner for the parallel technique. In this process area requirement is further reduced compared to parallel processing. The memory schematic for parallel processing MLD is shown in fig 3.
In Parallel schematic each bit of the code word fed for error detection and correction consist of its parity check equation, correction gate and majority gate. Area increase in the parallel process because of using individual gates for each bit and power also increases as the area increases.

5. Area, Power and Delay Analysis
The comparison for area, power and delay for the proposed parallel MLDD is shown in Table II

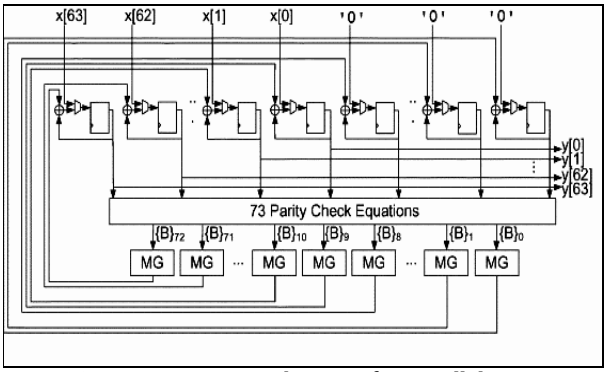

Figure 3: Memory schematic for parallel MLD

Design	Area	Power (mw)	Delay (ns)
Parallel MLDD	2400	130	3.212

Table 2: Comparison for Area, Power and Delay

5.1. Pipelining

The pipelining process is done for the proposed Parallel processing technique by adding registers. So that the delay is reduced compared to parallel processing, power and area is compared with the parallel technique increases. In pipelining technique, using partitioning method the input data is divided into several sets of inputs. The cut-set partitioning algorithm is used here. Since the inputs are partitioned, the area and power has been little bit increased compared to the Parallel technique. But the delay is reduced compared to the Serial and parallel techniques.

6. Area, Power and Delay Analysis

The comparison for area, power and delay for the proposed pipelining MLDD is shown in Table III

Design	Area	Power (mw)	Delay (ns)
Pipelining MLDD	2491	152	2.495

Table 3: Comparison for Area, Power and Delay

7. Preliminaries

Finite geometries have been used to derive many error-correcting codes [9], [11]. One example is EG-LDPC codes which are based on the structure of Euclidean geometries. One step majority logic decodable (MLD) is one of the subclass of EG-LDPC codes [9]. The results obtained can be summarized in the following hypothesis.

"Memory protection with DS-LDPC codes are read for word affected up to five bit flips and the error are detected in three decoding cycles "

The proposed technique was implemented in VHDL and synthesized, showing that for codes with large block size. This is because the existing majority logic decoding circuitry is reused to perform error detection and only some extra control logic is needed for error correction. The results obtained can be summarized in the following hypothesis.

"Memory protection with one step MLD EG-LDPC codes are read for word affected up to four bit flips and the error are detected in three decoding cycles "

8. Results

The proposed method has been applied to one step majority logic decoding EG-LDPC codes. 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, it is 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 of the errors are detected.

Parameters	Serial	Parallel	Pipelining
Delay	6.021 ns	3.212 ns	2.495 ns
Power	116.2 mw	130 mw	152 mw
Area	1090	2400	2491

Table 4: Comparison for Power, Area and Delay between theThree techniques

Table IV shows the comparison result of delay, power and area between serial, parallel and pipelining MLDD techniques. The delay reduction in parallel and pipelining techniques increases the memory applications. The simulations results help to determining the error occurrence. The simulation results show the error detection in single iteration and all the errors are detected. Thus the parallel and pipelining technique is more suitable for error detection and correction in memory applications.

9. Conclusion

In this brief, using parallel one step majority logic decoding technique the detection of errors single iteration is more practical than the serial manner. The EG-LDPC code was used to analyze errors up to four bit flips and DS-LDPC code was used to analyze errors up to five bit flips. The proposed method of parallel and its pipelining process detects any number of errors in a single iteration. Further in the future work includes extending the practical analysis to more number of errors with much more reduction of area and power.

10. References

1. Pedro Reviriego, Juan A. Maestro, and Mark F. Flanagan, (Jan 2013)"Error Detection in Majority Logic Decoding of Euclidean Geometry Low DensityParity Check (EG-LDPC) Codes," IEEE Trans. Very Large Scale Integr.(VLSI) Syst.. vol. 21, no. 1,
2. Bajura M.A, Y. Boulghassoul, R. Naseer, S. DasGupta, A. F.Witulski,J. Sondeen, S. D. Stansberry, J. Draper, L. W. Massengill, and J. N. Damoulakis, (Aug. 2007) "Models and algorithmic limits for an ECC-based approach to hardening sub-100-nm SRAMs," IEEE Trans. Nucl. Sci., vol. 54, no. 4, pp. 935–945,.

3. Baumann R.C, (Sep. 2005) "Radiation-induced soft errors in advanced semiconductor technologies," IEEE Trans. Device Mater. Reliab., vol. 5, no. 3,pp. 301–316,.
4. Ghosh.S and P. D. Lincoln, (2007). "Dynamic low-density parity check codes for fault-tolerant nano-scale memory," presented at the Foundations Nanosci. (FNANO), Snowbird, Utah,
5. Ghosh.S and 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.
6. Lin.S and D. J. Costello, (2004) ErrorControl Coding, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall,.
7. Liu.S, P. Reviriego, and J. Maestro, (Jan. 2012) "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,.
8. Naeimi.H and A. DeHon, (2007) "Fault secure encoder and decoder for Memory applications," in Proc. IEEE Int. Symp. Defect Fault Toler. VLSI Syst., , pp. 409–417.
9. Naeimi.H and A. DeHon, (Apr. 2009) "Fault secure encoder and decoder for nanomemory applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 4, pp. 473–486,.
10. Naseer.R and J. Draper, (2008) "DEC ECC design to improve memory reliability in sub-100 nm technologies," Proc. IEEE ICECS, pp. 586–589,.
11. Tang.H , J. Xu, S. Lin, and K. A. S. Abdel-Ghaffar ,(Feb.2005) "Codes on finite geometries," IEEE Trans. Inf. Theory, vol. 51, no. 2, pp. 572–596