2581-4575

International Journal For Recent Developments in Science & Technology

Crossref

A Peer Reviewed Research Journal

IMPLEMENTATION ANALYSIS OF HYBRID FSM DECODING TECHNIQUE FOR SRAMS

[1]MOGALI GANGA RATNAM, [2] P.NAGARAJU M.Tech,(Ph.D.)

[1]M.TECH VLSI, DEPT OF E.C.E, KAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY, KORANGI, ANDHRAPRADESH, INDIA, 533461

[2]ASSOCIATE PROFESSOR, KAKINADA INSTITUTE OF ENGINEERING AND TECHNOLOGY, KORANGI, ANDHRAPRADESH, INDIA, 533461

ABSTRACT:

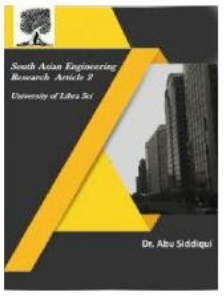
As a result of technology scaling and higher integration densities there may be variations in parameters and noise levels which will lead to larger error rates at various levels of the computations. As far as memory applications are concerned the soft errors and single event upsets are always a matter of problem. The paper mainly focuses on the design of an efficient Multi Detector/Decoder (MLDD) for fault detection along with correction of fault for memory applications, by considerably reducing fault detection time. The error detection and correction method is done by one step majority logic decoding and is made effective for Euclidean Geometry Low Density Parity Check Codes (EG-LDPC). Even though majority decodable codes can correct large number of errors, they need high decoding time for detection of errors and ML 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 fault in less decoding cycles (almost in three). When the data read is error free, it can obviously reduce memory access time. The technique keeps the area overhead minimal and power consumption low for large code word sizes.

Keywords: one step majority logic decoding; error correction codes (ECCs); Euclidean geometry low-density parity check (EG- LDPC); memory;control logic.

INTRODUCTION

Memories are the most universal component today. For more than a decade, memory cells have been protected from soft errors. Some type of embedded memory, such as ROM, SRAM, DRAM, flash memory etc is seen in almost all system chips. Now days, the memory failure rates are increasing due to the impact of technology scaling-smaller dimensions, high integration densities, lower operating voltages etc.[4],[5]. The ability to quickly determine that a bit has flipped is key to high reliability and high availability applications. Some commonly used error detecting techniques are Triple Modular Redundancy (TMR) and Error Correction

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal



Codes (ECCs).

The TMR triplicates all the memory parts of the system and to choose the correct data using a voter. This method have disadvantage of large area and complexity overhead of three times. Therefore the ECC became the best way to mitigate soft errors in memory [4].

The most commonly used ECC codes are Single Error Correction (SEC) codes that can correct one bit error in a memory word. Due to consequence of augmenting integration densities, there is an increase in soft errors which points the need for higher error correction capabilities [1], [3]. More advanced ECCs have been proposed for memory applications but even Double Error Correction (DEC) codes with a parallel implementation incur in a significant power consumption penalty. The usual multierror correction codes, such as Reed- Solomon (RS) or Bose Chaudhuri-Hocquenghem (BCH) are not suitable for this task due to complex decoding algorithm.

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, is focused here.

The advantages of ML decoding are that it is very simple to implement and thus it is very practical and has low complexity. The drawback of ML decoding is that, it needs as many cycles as the number of bits in the input signal, which is also the number of taps, N, in the decoder and also same decoding time for both error and error free code words. This is a great impact on the performance of the system, depending on the size of the code.

Another alternative is to first detect if there are errors in the word and only perform the rest of the decoding process when there are errors. This greatly reduces the average power consumption as most words will have no errors. Error detection in a block code can also be implemented by computing the syndrome and checking whether all its bits are zero [15]. 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 focus on using the MLD circuitry itself as an error detecting module therefore with no additional hardware the read operations could be accelerated.

The remainder of this paper is organized as follows. Section II gives an overview of existing ML decoding solutions. Section III presents the novel ML detector/decoder (MLDD)

using EG- LDPC cyclic codes. Section IV discusses the results obtained in respect to speedup, delay and power consumption. Finally, Section V discusses conclusions

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

and future work.

II. MAJORITY LOGIC DECODING (MLD) SOLUTONS

One-step majority-logic correction is a fast and relatively efficient error-correcting technique [6]. One-step-majority correctable ECC codes are limited which include type-I two-dimensional EG-LDPC.

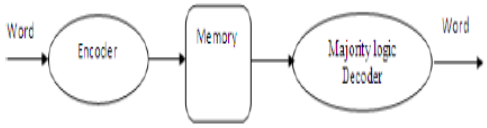


Figure 1: Representing the memory system schematic

The memory system schematic shown in Figure 1 show that the word is first encoded and is then written to the memory [2]. After the reading process of the memory it is passed to a majority logic detector block which detects and corrects the errors which occurred while the reading code word.

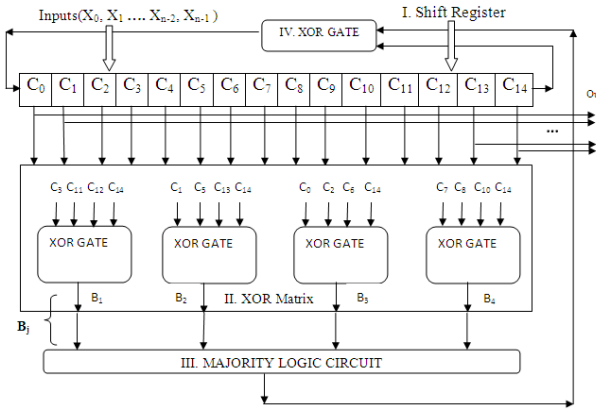


Figure 2: existed plain ML decoder

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, [9]. The Type-II ML decoder that calculates the information of correctness of the current bit under decoding, directly out of the codeword bits [6]. Both are quite similar, but when implementation is considered the Type-II uses less area, since it does not have a syndrome calculation as an intermediate step. For this reason the paper focus on this type II implementation.

A. Existent Plain ML Decoder

One-Step Majority-Logic Corrector: One-step majority logic correction is the process in which from the received codeword itself the correct values of each bit under decoding can directly found out. This method consists of mainly two steps- 1) Generating a specific set of linear sums of the received vector bits using the xor matrix 2) Determining the majority value of the computed linear sums. It is the majority logic output which determines the correctness of the bit under decoding. If the majority output is '1', then the bit is inverted, otherwise would be kept unchanged.

As described before, the ML decoder is powerful and simple decoder, which has the capability of correcting multiple random bit-flips depending on the number of parity check equations. It consists of four parts: 1) a cyclic shift register; 2) an XOR matrix; 3) a majority gate; and 4) an XOR for correcting the codeword bit under decoding. The circuit implementing a serial one-step majority logic corrector [6], [12] for (15, 7, 5) EG-LDPC code is shown in Figure 2.

The cyclic shift register is initially stored with the input signal x and shifted through

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

all the taps. The results {Bj} of the check sum equations from the XOR matrix is calculated from the intermediate values in each tap. In the Nth cycle, the result would reach the final tap, producing the output signal, which is the decoded version of input [2].

Figure 2. Serial one-step majority logic corrector for (15, 7, 5) EG-LDPC code

This is the situation of error free case. The input x might correspond to wrong data corrupted by a soft error or SEUs. The decoder is designed to handle this situation as follows.

From the parity check sum equations hardwired in the xor matrix the decoding starts at the very next moment after the codeword x are loaded into the cyclic shift register. The linear sum outputs {Bj} is then forwarded to the majority logic circuit which determines the correctness of the bit under decoding. If the majority of the Bj bits are "1" that is greater than the majority number of zeros then the current bit is erroneous and should be corrected, otherwise it is kept unchanged.

The process is repeated and contents of the shift registers are rotated up to the whole N bits of the codeword are processed. When all the parity check sums outputs are zero the codeword is correctly decoded. Further details on how this algorithm works can be found in [6], [12]. The whole algorithm [2] is depicted in Figure 3. The algorithm needs as many cycles as the number of bits in the input signal, which is number of taps, N, in the decoder and also needs same decoding time for both error and error free code words.

PROPOSED MULTI-DETECTOR/DECODER

A novel version of the MD decoder for improving performance is presented here. With reference to the original ML decoder, the proposed MD detector/decoder (MDD) has been implemented using the Hamming and parity check (HMPC) and general decoder. This proposed design uses much more easy way implementation for detecting and correcting.

The proof of the hypothesis that all error will be detected in eight cycles is very simple from the mathematical point of view. It is 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 flips 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 cases the combinations are checked randomly.

A. Design structure of the encoder

The encoder and corrector are two different operation that is used which randomly checking the memory but in this case we have considered a generalized encoder (3:8 or convolution encoder) which can encode the data (as shown in figure). As per the corrector we have the hamming parity check where each

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

code data is divided into sub sectors which enables to check the parity based on the division and then compared to its original value. Considering the original value will equal to the expected corrected value results in correction successfully.

B. Design Structure of Decoder

The decoder and detector structure have been shown in the figure. The decoder design is a complex design based on the (BCH decoder/Hamming decoder) which comprises of the received signal and error position from the detector. So hence based on the position and no of the errors found in the given sequence of the data is decoded accordingly.

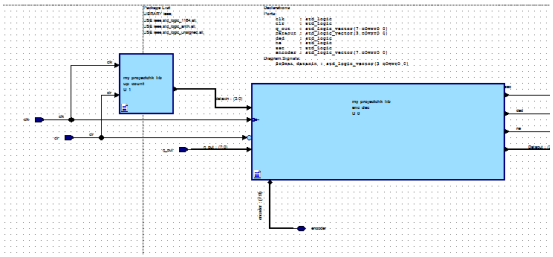


Figure 3: representing the correction and detection of memory

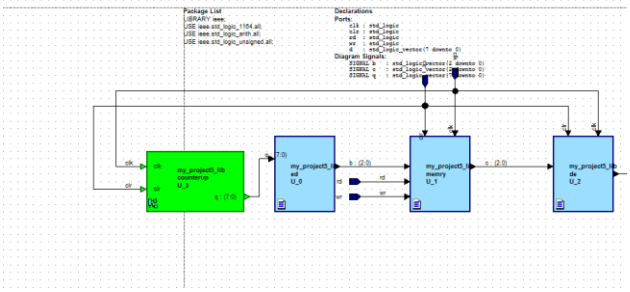


Figure4: Representing the encoding and decoding of the memory.

Simulation Results:

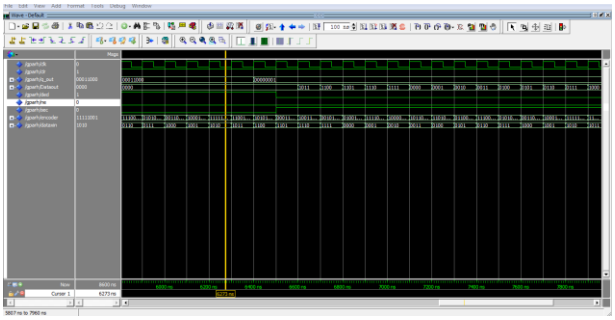


Figure5: Representing the encoding sequence

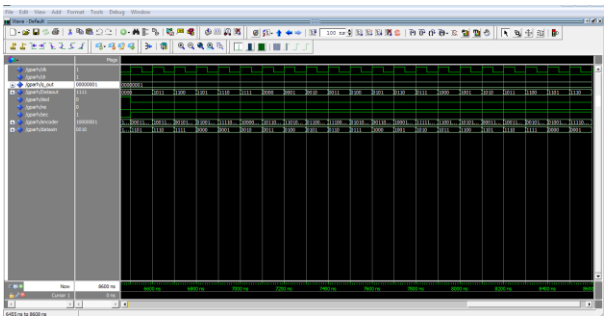


Figure6: Representing the encoding at different time frame

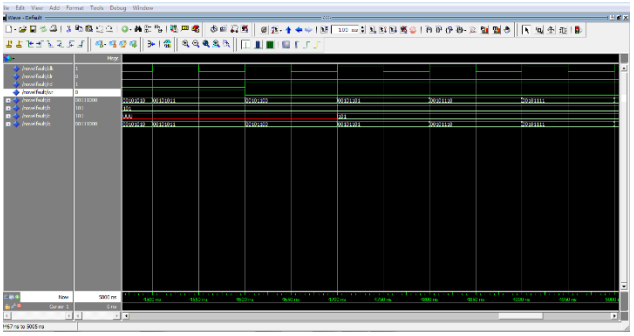


Figure7: Representing the decoding of the memory.

International Journal For Recent Developments in Science & Technology
2581-4575
Crossref
A Peer Reviewed Research Journal

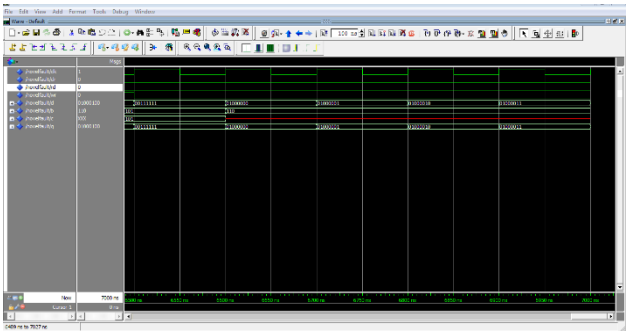


Figure8: Representing the decoding of the memory.

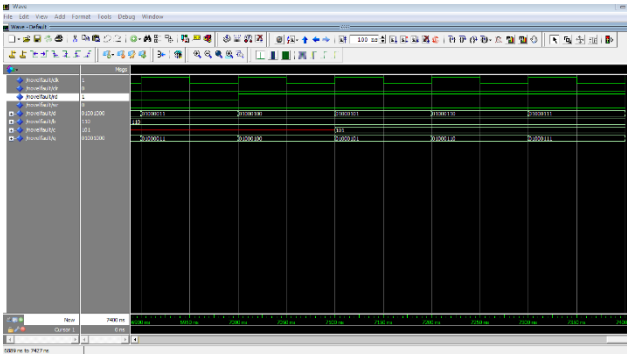


Figure 9: Representing the data processing using encoding and decoding the memory.

Table 1: Representing the Comparison of existing and proposed HFSMD memory

SNO	PARAMETERS	Existing memory	PROPOSED MEMORY WITH HFSMD
1	AREA	19.3 %	10.17%
2	POWER	8.856W	3.03W
3	DELAY	182.87 ns	94.58 ns
4	Throughput	96.85mbps	134.2mpbs
5	Latency	176.85 ns	95 ns

Conclusions:

The paper focuses on the design of a Hybrid FSM Decoder/Detector (HFSMDD) for fault detection along with correction of fault, suitable for memory applications, with reduced fault detection time.

From the simulation results, (A codeword of size 8 is chosen here for designing), when compared to the existing HFSMDD, The proposed HFSMDD has comparatively less delay of 12.578 ns and can detect the presence of errors in just 8 cycles even for multiple bit flips.It has found that for error detection and correction (for codeword of 15), when comparing to the existing technique, a speed up of about 1100 ns is obtained when there is no errors in data read access. It's because the fault detection needs only three cycles and after the detection of an error free condition, the codeword is passed to the output without further corrections. This is a great saving of time since most of the situations the memory read access does not make errors. Therefore there is a considerable reduction in the memory access time.The proposed HFSMDD have about 10% low power consumption than the existing MLD technique, since the proposed design detects the faults in just three cycles. Therefore a large no. of clock cycles (here 12 clock cycles) are saved and hence considerable reduction in power is achieved. HFSMDD error detector is designed as it is independent of the code word size and inference about area is that for large values of code word size, the area overhead of the HFSMDD actually decreases with respect to the plain HFSMDD technique. i.e., for large values of code word size both areas are practically the same. Therefore the proposed HFSMDD will be an efficient design for fault detection and correction

2581-4575
International Journal For Recent Developments in Science & Technology
Crossref
A Peer Reviewed Research Journal


The future research is to focus on the application oriented implementation of HFSMDD to memories and also by changing the internal architecture of majority gate we can obtain a more efficient, low power and low area HFSMDD.

REFERENCES

[1] N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, Dependability in Electronic Systems: Mitigation of Hardware Failures, Soft Errors, and Electro-Magnetic Disturbances. New York, NY, USA: Springer-Verlag, 2010.

[2] J. L. Autran et al., "Soft-errors induced by terrestrial neutrons and natural alpha-particle emitters in advanced memory circuits at ground level," Microelectron. Rel., vol. 50, no. 9, pp. 1822–1831, Sep. 2010.

[3] A. L. Silburt, A. Evans, I. Perryman, S. J. Wen, and D. Alexandrescu, "Design for soft error resiliency in Internet core routers," IEEE Trans. Nucl. Sci., vol. 56, no. 6, pp. 3551–3555, Dec. 2009. [4] A. Evans, S.-J. Wen, and M. Nicolaidis, "Case study of SEU effects in a network processor," in Proc. IEEE Workshop Silicon Errors Logic-Syst. Effects (SELSE), Mar. 2012, pp. 1–7. [5] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," IBM J. Res. Develop., vol. 28, no. 2, pp. 124–134, Mar. 1984. [6] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," IEEE J. Solid-State Circuits, vol. 41, no. 3, pp. 712–727, Mar. 2006.

[7] F. Yu, R. H. Katz, and T. V. Lakshman, "Efficient multimatch packet classification and lookup with TCAM," IEEE Micro, vol. 25, no. 1, pp. 50–59, Jan./Feb. 2005. [8] P. Bosshart et al., "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN," in Proc. ACM SIGCOMM, 2013, pp. 99–110.

[9] I. Syafalni, T. Sasao, and X. Wen, "A method to detect bit flips in a soft-error resilient TCAM," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 37, no. 6, pp. 1185–1196, Jun. 2018.

[10] S. Pontarelli, M. Ottavi, A. Evans, and S. Wen, "Error detection in ternary CAMs using Bloom filters," in Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE), Mar. 2013, pp. 1474–1479.

[11] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "NetFPGA SUME: Toward 100 Gbps as research commodity," IEEE Micro, vol. 34, no. 5, pp. 32–41, Sep./Oct. 2014.

[12] M. Irfan and Z. Ullah, "G-AETCAM: Gate-based area-efficient ternary content-addressable memory on FPGA," IEEE Access, vol. 5, pp. 20785–20790, 2017.

[13] W. Jiang, "Scalable ternary content addressable memory implementation using FPGAs," in Proc. ACM ANCS, San Jose, CA, USA, Oct. 2013, pp. 71–82.

[14] Z. Ullah, M. K. Jaiswal, and R. C. C. Cheung, "E-TCAM: An efficient SRAM-based architecture for TCAM," Circuits,

2581-4575
International Journal For Recent Developments in Science & Technology
Crossref
A Peer Reviewed Research Journal

Syst., Signal Process., vol. 33, no. 10, pp. 3123–3144, Oct. 2014.

[15] A. Ahmed, K. Park, and S. Baeg, "Resource-efficient SRAM-based ternary content addressable memory," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 4, pp. 1583–1587, Apr. 2017.

[16] I. Ullah, Z. Ullah, and J.-A. Lee, "Efficient TCAM design based on multipumping-enabled multiported SRAM on FPGA," IEEE Access, vol. 6, pp. 19940–19947, 2018.

[17] Ternary Content Addressable Memory (TCAM) Search IP for SDNet: SmartCORE IP Product Guide, PG190 (v1.0), Xilinx, San Jose, CA, USA, Nov. 2017.

[18] V. Gherman and M. Cartron, "Soft-error protection of TCAMs based on ECCs and asymmetric SRAM cells," Electron. Lett., vol. 50, no. 24, pp. 1823–1824, 2014.

[13] C.Tjhai, M. Tomlinson, M. Ambroze, and M. Ahmed, "Cyclotomic idempotent- based binary cyclic codes," Electron. Lett., vol. 41, no. 6, Mar. 2005.

[14] T Shibuya and K. Sakaniwa, "Construction of cyclic codes suitable for iterative decoding via generating idempotents," IEICE Trans. Fundamentals, vol. E86-A, no. 4, pp. 928-939, 2003.

[15] P .Ankolekar, S. Rosner, R. Isaac, and J. Bredow, "Multi-bit error correction methods for latency-contrained flash memory systems," IEEE Trans. Device Mater. Reliabil., vol. 10, no. 1, pp. 33-39, Mar. 2010.

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