IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)
e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 8, Issue 4 (Nov. - Dec. 2013), PP 07-12
www.iosrjournals.org

High Performance Error Detection with Different Set Cyclic Codes for Memory Applications

Karthik.N[1], T.N.Suresh[2]
[1](Department of EEE, Hindustan University, India)
[2](Department of EEE, Sri Venkateswara College of Engineering & Technology, India)

Abstract : *This paper presents an error detection method with majority logic decoding. The majority logic decodable words are suitable for memory application due to their capability to correct large number of errors, but they require a large decoding time that impacts the memory performance. So in the proposed fault detection method they significantly reduce the memory access time.*

Keywords: *Triple Modular Redundancy, Error Correction Codes, Majority Logic, Low Density parity checks, Difference Set Cyclic Codes, Syndrome Fault Detector, Majority Logic Detector/Decoder.*

I. INTRODUCTION

The impact of technology scaling—smaller dimensions, higher integration densities, and lower operating voltages—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. Especially, SRAM memory failure rates are increasing significantly, therefore posing a major reliability concern for many applications. Some commonly used mitigation techniques are:

- Triple Modular Redundancy (TMR);
- Error Correction Codes (ECCs).

TMR is a special case of the von Neumann method consisting of three versions of the design in parallel, with a majority voter selecting the correct output. As the method suggests, the complexity overhead would be three times plus the complexity of the majority voter and thus increasing the power consumption. For memories, it turned out that ECC codes are the best way to mitigate memory soft errors. Among the ECC codes that meet the requirements of higher error correction capability and low decoding complexity, cyclic block codes have been identified as good candidates, due to their property of being majority logic (ML) decodable. A subgroup of the low-density parity check (LDPC) codes, which belongs to the family of the ML decodable codes, namely the difference-set cyclic codes (DSCCs), which is widely used in the Japanese teletext system or FM multiplex broadcasting systems.

II. ERROR CORRECTING CODES

An error-correcting code is an algorithm for expressing a sequence of numbers such that any errors which are introduced can be detected and corrected based on the remaining numbers. The study of error correcting codes and the associated mathematics is known as coding theory. Error detection is much simpler than correction and one or more 'check' digits are commonly embedded in credit card numbers in order to detect mistakes. Early space probes like mariner used a type of error-correcting code called a block code and more recent space probe use convolution codes. Error correcting codes are also used in CD players, high speed modems, and cellular phones. Modems use error detection when they compute checksums, which are sums of the digits in a given transmission modulo some number.

III. BLOCK CODES

In coding theory, block codes refers to the large and important family of error-correcting codes that encode data in blocks. There is a vast number of examples for block codes, many of which have a wide range of practical applications. The main reason why the concept of block codes is so useful is that, it allows coding theorists, mathematicians and computer scientists to study the limitations of all block codes in a unified way. Such limitations often take the form of bounds that relate different parameters of the block code to each other such as its rate and its ability to detect and correct errors. Examples of block codes are Reed-Solomon codes, Hamming codes, Hadamard codes, Expander codes, Golay codes and Reed-Muller codes. These examples also belong to the class of linear codes, and hence they are called linear codes.

IV. DIFFERENT SET CYCLIC CODES

Codes exist which are capable of correcting large numbers of random errors. Such codes are rarely used in practical data transmission systems, however, because the equipment necessary to realize their capabilities-that

is, to actually correct the errors-is usually prohibitively complex and expensive. The problem of finding simply implemented decoding algorithms or, equivalently, codes which can be decoded simply with existing methods is perhaps the outstanding unsolved problem in coding theory today. Difference-set cyclic code is a new class of random-error-correcting cyclic code which has two desirable features as

- The binary members of the class are nearly as powerful as the best-known codes in the range of interest.
- They can be decoded with the simplest known decoding algorithm

V. Low-Density Parity Check (Ldpc) Codes

In information theory, a low-density parity-check (LDPC) code is a linear error correcting code.LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close to the theoretical maximum for a symmetric memory-less channel, the noise threshold defines an upper bound for the channel noise, up to which the probability of lost information can be made as small as desired. LDPC codes are finding increasing use in applications requiring reliable and highly efficient information transfer over bandwidth or return channel-constrained links in the presence of data-corrupting noise. Although implementation of LDPC codes has lagged behind that of other codes, notably turbo codes, the absence of encumbering software patents has made LDPC attractive to some.

VI. Majority Logic Decoding (Mld)

MLD is based on a 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 correctness of the current bit under decoding.

VII. Existing System

PLAIN ML DECODER

The ML decoder is a simple and powerful decoder, capable of correcting multiple random bit-flips depending on the number 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

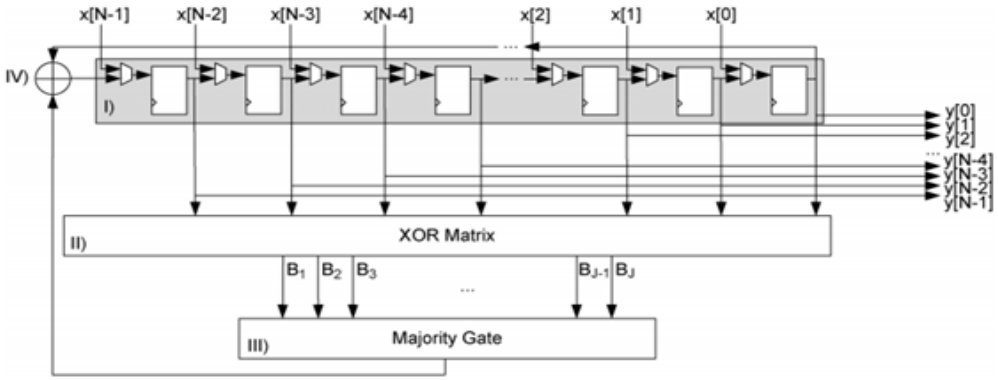


Fig. 1: Plain ML Decoder

PLAIN MLD WITH SYNDROME FAULT DETECTOR

In order to improve the decoder performance, alternative designs may be used. One possibility is to add a fault detector by calculating the syndrome, so that only faulty codeword's are decoded. Since most of the codeword's will be error-free, no further correction will be needed, and therefore performance will not be affected. Although the implementation of an SFD reduces the average latency of the decoding process, it also adds complexity to the design.

The SFD is an XOR matrix that calculates the syndrome based on the parity check matrix. Each parity bit results in a syndrome equation. Therefore, the complexity of the syndrome calculator increases with the size of the code.

The input signals x 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 checksum equations from the XOR matrix. In the Nth cycle, the result has reached the final tap, producing the output signal y.

As an initial step, codeword x 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. Finally, the parity check sums should be zero if the codeword has been correctly decoded.

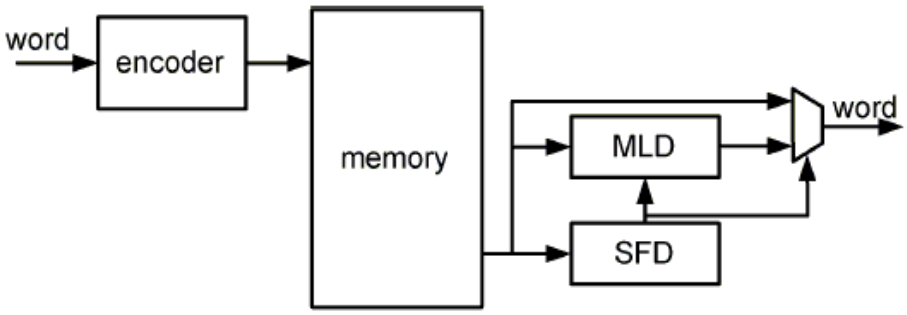


Fig. 2: Plain MLD with SFD

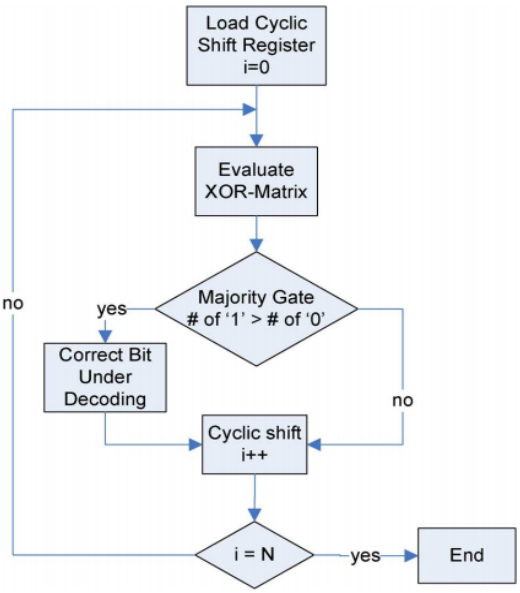


Fig. 3: Existing System Flowchart

After the initial step, in which 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 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 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. Finally, the parity check sums should be zero if the Codeword has been correctly decoded.

Disadvantages Of The Existing System

- The plain ML Decoder needs as many cycles as the number of bits in the input signal, which is also the number of taps, , in the decoder. This is a big impact on the performance of the system, depending on the size of the code . For example, for a codeword of 73 bits, the decoding would take 73 cycles, which would be excessive for most applications.

- The syndrome fault detector results in a quite complex module, with a large amount of additional hardware and power consumption in the system.

VIII. Proposed System

This section presents a modified version of the ML decoder that improves the designs presented before. Starting from the original design of the ML decoder introduced in, the proposed ML detector/decoder (MLDD) has been implemented using the difference-set cyclic codes (DSCCs). This code is part of the LDPC codes, and, based on their attributes, they have the following properties

• 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

Since performance is important for most applications, we have chosen an intermediate solution, which provides a good reliability with a small delay penalty for scenarios where up to five bit-flips may be expected and it is based on the following hypothesis:

Given a word read from a memory and affected by up to five bit-flips, all errors can be detected in only three decoding cycles.

In general, the decoding algorithm is still the same as the one in the plain ML decoder version. The difference is that, instead of decoding all codeword bits by processing the ML decoding during cycles, the proposed method stops intermediately in the third cycle.

The figure below shows the proposed MLDD which consists of following parts:

- a cyclic shift register
- an XOR matrix
- a majority gate an XOR for correcting the codeword bit under decoding
- control unit
- tristate buffer

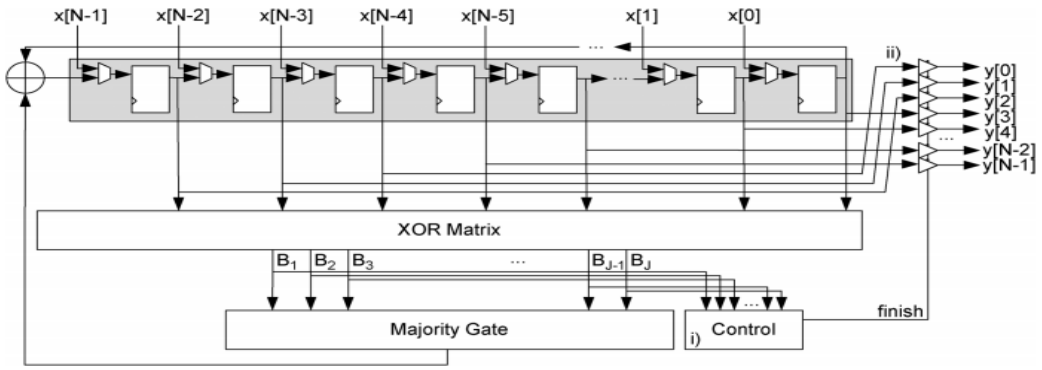


Fig. 4: Proposed System

The figure shows the basic ML decoder with an N-tap shift register, an XOR array to calculate the orthogonal parity check sums and a majority gate for deciding if the current bit under decoding needs to be inverted. Those components are the same as the ones for the plain ML decoder. The additional hardware to perform the error detection is the control unit which triggers a finish flag when no errors are detected after the third cycle and the output tristate buffers. The output tristate buffers are always in high impedance unless the control unit sends the finish signal so that the current values of the shift register are forwarded to the output y.

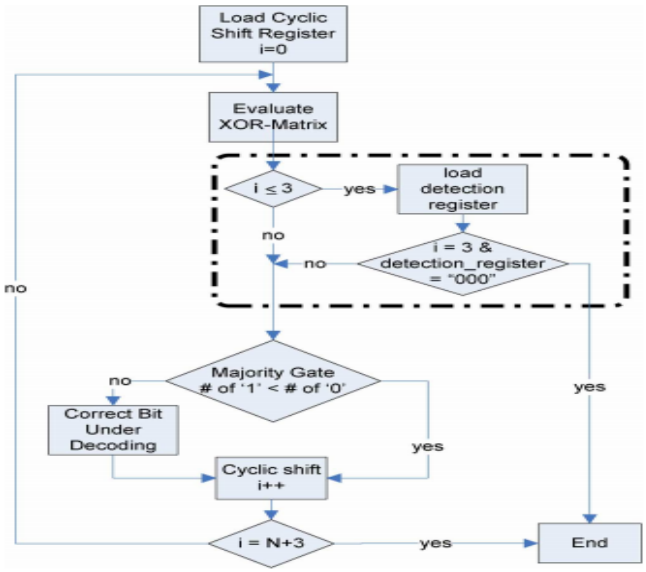


Fig. 5: Proposed System Flowchart

In general, the decoding algorithm is still the same as the one in the plain ML decoder version. The difference is that, instead of decoding all codeword bits by processing the ML decoding during cycles, the proposed method stops intermediately in the third cycle, as illustrated in Flow chart. If in the first three cycles of the decoding process, the evaluation of the XOR matrix for all is "0," the codeword is determined to be error-free and forwarded directly to the output. If they contain in any of the three cycles at least a "1," the proposed method would continue the whole decoding process in order to eliminate the errors.

CONTROL UNIT

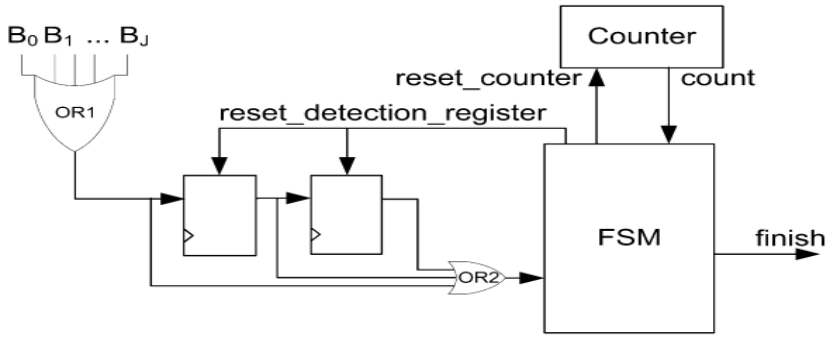


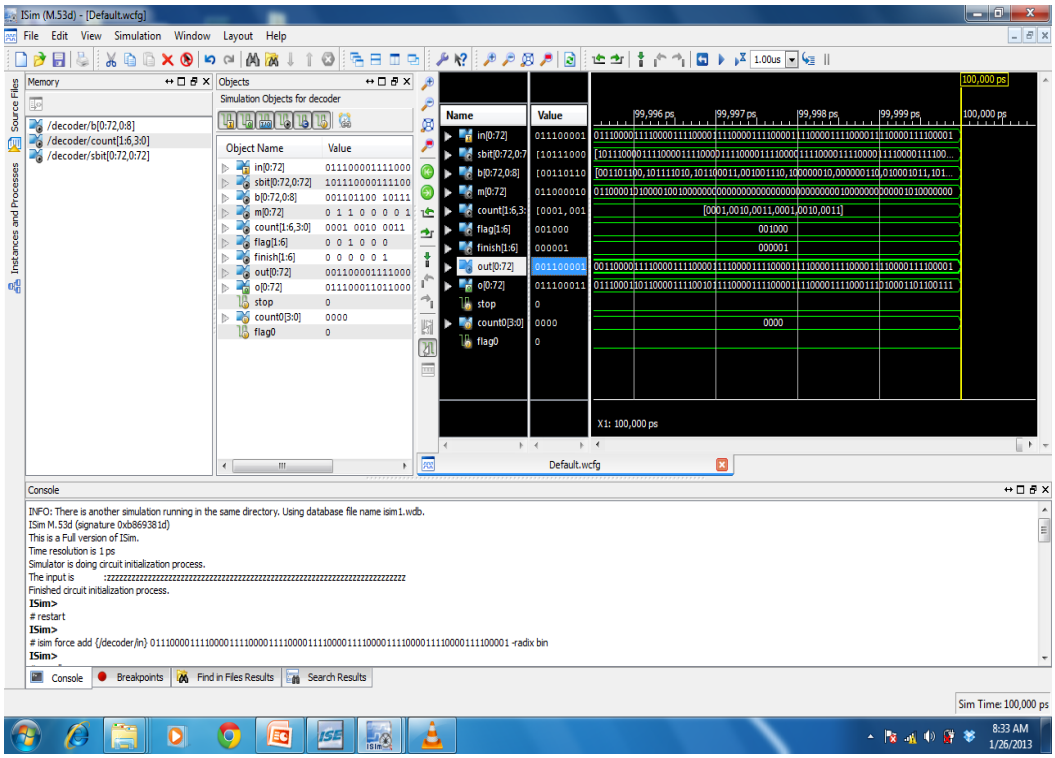
Fig. 6: Control Unit

The control unit manages the detection process. It uses a counter that counts up to three, which distinguishes the first three iterations of the ML decoding. In these first three iterations, the control unit evaluates the by combining them with the OR1 function. This value is fed into a three-stage shift register, which holds the results of the last three cycles. In the third cycle, the OR2 gate evaluates the content of the detection register. When the result is "0," the FSM sends out the finish signal indicating that the processed word is error-free. In the other case, if the result is "1," the ML decoding process runs until the end.

ADVANTAGES OF THE PROPOSED SYSTEM

- Ability to correct large number of bit flips.
- It takes lesser decoding cycles.
- Uses less memory and low power consumption.

IX. Simulation Results



X. Conclusion

In this paper, a fault-detection mechanism, MLDD, has been presented based on ML decoding using the DSCCs.In the proposed technique it 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. On the other hand, the MLDD error detector module has been designed in a way that is independent of the code size. This makes area overhead quite reduced compared with other traditional approaches such as the syndrome calculation (SFD)

Acknowledgements

The author wish to thank the Management of Hindustan University, Padur Chennai for their support and encouragement to carry out this work.

References

[1]. E. J. Weldon, Jr., "Difference-set cyclic codes," Bell Syst. Tech. J., vol.45, pp. 1045–1055, 1966.
[2]. Y. Kato and T. Morita, "Error correction circuit using difference-set cyclic code," in Proc. ASP-DAC, 2003, pp. 585–586.
[3]. S. Lin and D. J. Costello, Error Control Coding, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004
[4]. 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.
[5]. 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.
[6]. 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.