RESEARCH ARTICLE OPEN ACCESS

Error Locked Encoder and Decoder for Nanomemory Application

Y. Sharath[1], S.Srivani[2], O.M.Chandrika[3], B.Gopi Krishna[4]

Abstract

Memory cells have been protected from soft errors for more than a decade; due to the increase in soft error rate in logic circuits, the encoder and decoder circuitry around the memory blocks have become susceptible to soft errors as well and must also be protected. We introduce a new approach to design fault-secure encoder and decoder circuitry for memory designs. The key novel contribution of this paper is identifying and defining a new class of error-correcting codes whose redundancy makes the design of fault-secure detectors (FSD) particularly simple. We further quantify the importance of protecting encoder and decoder circuitry against transient errors, illustrating a scenario where the system failure rate (FIT) is dominated by the failure rate of the encoder and decoder. We prove that Euclidean Geometry Low-Density Parity-Check (EG-LDPC) codes have the fault-secure detector capability. Using some of the smaller EG-LDPC codes, we can tolerate bit or nanowire defect rates of 10% and fault rates of 10^{-18} upsets/device/cycle, achieving a FIT rate at or below one for the entire memory system and a memory density of 10^{11} bit/cm with nanowire pitch of 10 nm for memory blocks of 10 Mb or larger. Larger EG-LDPC codes can achieve even higher reliability and lower area overhead.

Key terms: Decoder, encoder, fault tolerant, memory, density parity-check , fault-secure detector and nanotechnology.

I. INTRODUCTION

Memory cells have been protected from soft errors for more than a decade; due to the increase in soft error rate in logic circuits, the encoder and decoder circuitry around the memory blocks have become susceptible to soft errors as well and must also be protected. We introduce a new approach to design fault-secure encoder and decoder circuitry for memory designs. Nanotechnology provides smaller, faster, and lower energy devices, which allow more powerful and compact circuitry; however, these benefits come with a cost, the nano scale devices may be less reliable. Thermal- and shot-noise estimations alone suggest that the transient fault rate of an individual nano scale device (e.g., transistor or nano wire) may be orders of magnitude higher than today's devices. As a result, we can expect combinational logic to be susceptible to transient faults, not just the storage and communication systems. Therefore, to build fault-tolerant nano scale systems, we must protect both combinational logic and memory against transient faults. In the present work we introduce a fault-tolerant nano scale memory architecture which tolerates transient faults both in the storage unit and in the supporting logic.

Memory types: Electronic space provided by silicon chips (semiconductor memory chips) or magnetic/optical media as temporary or permanent storage for data and/or instructions to control a computer or execute one or more programs. Two main types of computer memory are (1) Read only memory (ROM), smaller part of a computer's silicon (solid state) memory that is fixed in size and permanently stores manufacturer's instructions to run the computer when it is switched on. (2) Random access memory (RAM), larger part of a computer's memory comprising of hard disk, CD, DVD, floppies etc., (together called secondary storage) and employed in running programs and in archiving of data. Memory chips provide access to stored data or instructions that is hundreds of times faster than that provided by secondary storage.

Error Control Coding: Error detection and correction **or** error control**s** are techniques that enable reliable delivery of digital data over unreliable communication channels_(or storage medium). Error detection is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver. Error correction is the detection of errors and reconstruction of the original, error-free data.

The goal of error control coding is to encode information in such a way that even if the channel (or storage medium) introduces errors, the receiver can correct the errors and recover the original transmitted information. ECC stands for "Error Correction Codes" and is a method used to detect and correct errors introduced during storage or transmission of data. Certain kinds of RAM chips inside a computer implement this technique to correct data errors and are known as ECC Memory.

ECC Memory chips are predominantly used in servers rather than in client computers. Memory errors are proportional to the amount of RAM in a computer as well as the duration of operation. Since servers typically contain several Gigabytes of RAM and are in operation 24 hours a day, the likelihood of errors cropping up in their memory chips is comparatively high and hence they require ECC Memory. Memory errors that are not corrected immediately can eventually crash a computer. This again has more relevance to a server than a client computer in an office or home environment. When a client crashes, it normally does not affect other computers even when it is connected to a network, but when a server crashes it brings the entire network down with it. Hence ECC memory is mandatory for servers but optional for clients unless they are used for mission critical applications.

Error-correcting codes

Any error-correcting code can be used for error detection. A code with minimum Hamming distance, d, can detect up to d-1 errors in a code word. Using minimum-distance-based error-correcting codes for error detection can be suitable if a strict limit on the minimum number of errors to be detected is desired. Codes with minimum Hamming distance d=2 are degenerate cases of error-correcting codes, and can be used to detect single errors. The parity bit is an example of a single-error-detecting code. The Berger code is an early example of a unidirectional error code that can detect any number of errors on an asymmetric channel, provided that only transitions of cleared bits to set bits or set bits to cleared bits can occur.

An error correcting code (ECC) or forward error correction (FEC) code is a system of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when a number of errors (up to the capability of the code being used) were introduced, either during the process of transmission, or on storage. Since the receiver does not have to ask the sender for retransmission of the data, a back-channel is not required in forward error correction, and it is therefore suitable for simplex communication such as broadcasting. Error-correcting codes are frequently used in lower-layer communication, as well as for reliable storage in media such as CDs, DVDs, hard disks, and RAM.

Hamming Codes

Hamming codes are an extension of this simple method that can used to detect and correct a larger set of errors. Hamming's development is a very direct construction of a code that permits correcting single-bit errors. He assumes that the data to be transmitted consists of a certain number of information bits u, and he adds to these a number of check bits *p* such that if a block is received that has at most one bit in error, then p identifies the bit that is in error (which may be one of the check bits). Specifically, in Hamming's code p is interpreted as an integer which is 0 if no error occurred, and otherwise is the 1-origined index of the bit that is in error. Let *k* be the number of information bits, and *m* the number of check bits used. Because the *m* check bits must check themselves as well as the information bits, the value of p, interpreted as an integer, must range from 0 to which are distinct values. Because *m* bits can distinguish cases, we must have

$$2^m \geq m + k + 1 \quad (1)$$

This is known as the Hamming rule. It applies to any single error correcting (SEC) binary FEC block code in which all of the transmitted bits must be checked. The check bits will be interspersed among the information bits in a manner described below. Because p indexes the bit (if any) that is in error, the least significant bit of p must be 1 if the erroneous bit is in an odd position, and 0 if it is in an even position or if there is no error. A simple way to achieve this is to let the least significant bit of p, p_0, be an even parity check on the odd positions of the block, and to put p_0 in an odd position. The receiver then checks the parity of the odd positions (including that of p_0). If the result is 1, an error has occurred in an odd position, and if the result is 0, either no error occurred or an error occurred in an even position. This satisfies the condition that p should be the index of the erroneous bit, or be 0 if no error occurred.

Reed-Solomon codes

Reed decoding algorithm was not known. A solution for the latter was found in 1969 by Berel and James Massey, and is since known as the Berlekamp-Massey decoding algorithm . In 1977, RS codes were notably implemented in the Voyager program in the form of concatenated codes. The first commercial application in mass-produced consumer products appeared in 1982 with the compact disc, where two interleaved RS codes are used.

The parameters of a Reed-Solomon code are
m = the number of bits per symbol
n = the block length in symbols
k = the uncoded message length in symbols
(n-k) = the parity check symbols (check bytes)
t = the number of correctable symbol errors
(n-k)=*2t* (for n-*k* even)
(n-k)-1 = *2t* (for n-*k* odd)

Therefore, an RS code may be described as an (n,k) code for any RS code where, $n \leq 2^m - 1$, and $n - k \geq 2t$.
Consider the RS (255,235) code. The encoder groups the message into 235 8-bit symbols and adds 20 8-bit symbols of redundancy to give a total block length of 255 8-bit symbols. In this case, 8% of the transmitted message is redundant data. In general, due to decoder constraints, the block length cannot be arbitrarily large. The block length for the PerFEC codes is bounded by the following equation

$$1 + 2t \leq n \leq 255$$

The number of correctable symbol errors (t), and block length (n) is set by the user.

The encoders and decoders for Hamming and Hsiao codes, for example, have low encoding and decoding complexity, but also have relatively low error- correcting capacity (e.g., Hamming is single error-correcting, double error-detecting). To achieve higher error- correcting capability, codes like Reed Solomon or BCH require more sophisticated decoding algorithms.

LDPC CODES

LOW-density parity-check (LDPC) codes were first discovered by Gallager in the early 1960s and have recently been rediscovered and generalized .It has been shown that these codes achieve a remarkable performance with iterative decoding that is very close to the Shannon limit. Consequently, these codes have become strong competitors to turbo codes for error control in many communication and digital storage systems where high reliability is required.LDPC codes can be constructed using random or deterministic approaches. In this report, we focus on a class of LDPC codes known as Euclidean Geometric (EG) LDPC codes, which are constructed deterministically using the points and lines of a Euclidean geometry [1, 16]. The EG LDPC codes that we consider are cyclic and consequently their encoding can be efficiently implemented with linear shift registers. Minimum distances for EG codes are also reasonably good and can be derived analytically. Iteratively decoded EG LDPC codes also seem to not have the serious error- floors that plague randomly-constructed LDPC codes; this fact can be explained by the observation made in that EG LDPC codes do not have pseudo-code words of weight smaller than their minimum distance. For these reasons, EG LDPC codes are good candidates for use in applications like optical communications that require very fast encoders and decoders and very low bit error-rates.

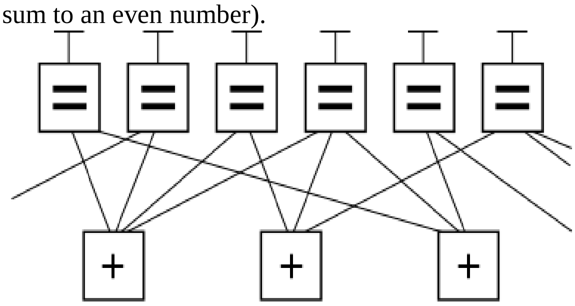
In information theory , a **low-density parity-check** (**LDPC**) **code** is a linear error correcting code, a method of transmitting a message over a noisy transmission channel, and is constructed using a sparse bipartite graph. LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close (or even arbitrarily close on the BEC) to the theoretical maximum (the Shannon limit) 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. Using iterative belief propagation techniques, LDPC codes can be decoded in time linear to their block length.

LDPC codes are finding increasing use in applications where reliable and highly efficient information transfer over bandwidth or return-channel constrained links in the presence of data-corrupting noise is desired. Although implementation of LDPC codes has lagged that of other codes, notably turbo codes, the absence of encumbering software patents has made LDPC attractive to some. LDPC codes are also known as **Gallager codes**, in honor of Robert G. Gallager, who developed the LDPC concept in his doctoral dissertation at MIT in 1960.

Function

LDPC codes are defined by a sparse parity-check matrix. This sparse matrix is often randomly generated, subject to the sparsity constraints. These codes were first designed by 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 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, module two, to zero (in other words, they must sum to an even number).



Ignoring any lines going out of the picture, there are 8 possible 6-bit strings corresponding to valid code words (i.e., 000000, 011001, 110010, 101011, 111100, 100101, 001110, 010111). This LDPC code fragment represents a 3-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

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

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 codeword.

In this example, the eight codewords can be obtained by putting the parity-check matrix **H** into this form $\left[-P^T | I_{n-k}\right]$ through basic row

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

From this, the generator matrix **G** can be obtained as $\left[I_k | P\right]$ (noting that in the special case of this being a binary code P = − P), or specifically

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

Finally, by multiplying all eight possible 3-bit strings by **G**, all eight valid codewords are obtained. For example, the codeword for the bit-string '101' is obtained by

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

Decoding:

As with other codes, optimally decoding an LDPC code on the binary symmetric channel is an NP-complete problem, although techniques based on iterative belief propagation used in practice lead to good approximations. In contrast, belief propagation on the binary erasure channel is particularly simple where it consists of iterative constraint satisfaction. For example, consider that the valid codeword, 101011, from the example above, is transmitted across a binary erasure channel and received with the first and fourth bit erased to yield ?01?11. Since the transmitted message must have satisfied the code constraints, the message can be represented by writing the received message on the top of the factor graph.

In this example, the first bit cannot yet be recovered, because all of the constraints connected to it have more than one unknown bit. In order to proceed with decoding the message, constraints connecting to only one of the erased bits must be identified. In this example, either the second or third constraint suffices. Examining the second constraint, the fourth bit must have been 0, since only a 0 in that position would satisfy the constraint. This procedure is then iterated. The new value for the fourth bit can now be used in conjunction with the first constraint to recover the first bit as seen below. This means that the first bit must be a 1 to satisfy the leftmost constraint. Thus, the message can be decoded iteratively. For other channel models, the messages passed between the variable nodes and check nodes are real numbers, which express probabilities and likelihoods of belief.

This result can be validated by multiplying the corrected codeword **r** by the parity-check matrix **H**

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

Because the outcome **z** (the syndrome) of this operation is the 3 × 1 zero vector, the resulting codeword **r** is successfully validated.

II. RELATED WORK

Particularly, we identify a class of error-correcting codes (ECCs) that guarantees the existence of a simple fault-tolerant detector design. This class satisfies a new, restricted definition for ECCs which guarantees that the ECC codeword has an appropriate redundancy structure such that it can detect multiple errors occurring in both the stored codeword in memory and the surrounding circuitries. We call this type of error-correcting codes, are fault-secure detector capable ECCs (FSD-ECC). The parity-check Matrix of an FSD-ECC has a particular structure that the decoder circuit, generated from the parity-check Matrix, is Fault-Secure. In this paper, we present our novel, restricted ECC definition for our fault-secure detector capable codes. Before starting the details of our new definition we briefly review basic linear ECCs.

In this proposed system the encoder is protected with parity-prediction and parity checker. The decoder is protected by adding a code checker (detector) block. If the code checker detects a non-codeword, then the error in the decoder is detected.

The restricted ECC definition which guarantees a fault-secure detector capable ECC is as follows. Let C be an ECC with minimum distance d. C is FSD-ECC if it can detect any combination of overall (d – 1) or fewer errors in the received codeword and in the detector circuitry.

FSD-ECC:

The restricted ECC definition which guarantees a FSD-ECC is as follows.
Definition I Let be an ECC with minimum distance is FSD-ECC if it can detect any combination of overall or fewer errors in the received codeword and in the detector circuitry.

Theorem I: Let c be an ECC, with minimum distance d . C Is FSD-ECC if any error vector of weight $0 < e < d - 1,$ has syndrome vector of weight at least $d - e$.
Note: The following proof depends on the fact that any single error in the detector circuitry can corrupt at most one output (one syndrome bit). This can be easily satisfied for any circuit by implementing the circuit in such a way that no logic element is shared among multiple output bits; therefore, any single error in the circuit corrupts at most one output (one syndrome bit).

Proof: The core of a detector circuitry is a multiplier that implements the vector-matrix multiply of the received vector and the parity-check matrix to generate the syndrome vector. Now if e errors strike the received codeword the syndrome weight of the error pattern is at least $d - e$ from the assumption.

Furthermore, the maximum number of tolerable errors in the whole system is $d - e$ and e errors already exist in the encoded vector, therefore the maximum number of errors that can strike in the detector circuitry is $d - 1 - e$. From the previous note, these many errors can corrupt at most $d - 1 - e$ syndrome bit, which in worst case leaves at least one nonzero syndrome bit and therefore detects the errors.

The difference between FSD-ECC and normal ECC is simply the demand on syndrome weight. That is, for error vector of weight $e > 0$, a normal ECC demands nonzero syndrome weight while FSD-ECC demands syndrome weight of $\geq d - e.$

Euclidean Geometry Code Review

The construction of Euclidean Geometry codes based on the lines and points of the corresponding finite geometries. Euclidean Geometry codes are also called EG-LDPC codes based on the fact that they are low-density parity-check (LDPC) codes. LDPC codes have a limited number of 1's in each row and column of the matrix; this limit guarantees limited complexity in their associated detectors and correctors making them fast and light weight.

Low-Density Parity-Check (LDPC) codes are a class of recently re-discovered highly efficient linear block codes. They can provide performance very close to the channel capacity (the theoretical maximum) using an iterated soft-decision decoding approach, at linear time complexity in terms of their block length. LDPC codes were first introduced by Robert G. Gallager in his PhD thesis in 1960.LDPC codes are now used in many recent high-speed communication standards, such as DVB-S2 (Digital video broadcasting), WiMAX (IEEE 802.16e standard for microwave communications), High-Speed Wireless LAN (IEEE 802.11n),10GBase-T Ethernet (802.3an) and G.hn/G.9960 (ITU-T Standard for networking over power lines, phone lines and coaxial cable). A Low Density Parity Check Code (LDPC) is one where the parity check matrix is binary and sparse, where most of the entries are zero and only a small fraction are 1's. In its simplest form the parity check matrix is constructed at random subject to some rather weak constraints on H.

Let EG be a Euclidean Geometry with n points and J lines. EG is a finite geometry that is shown to have the following fundamental structural properties

1) Every line consists of ρ points;
2) Any two points are connected by exactly one line;
3) Every point is intersected by γ lines;
4) Two lines intersect in exactly one point or they are parallel;
i.e., they do not intersect.

Let H be a $J \times n$ binary matrix, whose rows and columns corresponds to lines and points in an EG Euclidean geometry, respectively, where $h_{i,j} = 1$ if and only if the ith line of EG contains the jth point of EG, and $h_{i,j} = 0$ otherwise.

A row in H displays the points on a specific line of EG and have weight ρ .A column in H displays the lines that intersect at a specific point in EG and have weight γ . The rows of H are called the incidence vectors of the lines in EG, and the columns of H are called the intersecting vectors of the points in EG. Therefore, H is the incidence matrix of the lines in EG over the points in EG. It is shown in [15] that H is a LDPC matrix, and therefore the code is an LDPC code.

A special subclass of EG-LDPC codes, type-I 2-D EG-LDPC, is considered here. It is shown in [15] that type-I 2-D EG-LDPC has the following parameters for any positive integer $t \geq 2$

- Information bits, $k = 2^{2t} - 3^{t}$;
- Length, $n = 2^{2t} - 1$;
- Minimum distance, $d_{\min} = 2^{t} + 1$;
- Dimensions of the parity-check matrix, $n \times n$;
- Row weight of the parity-check matrix, $\rho = 2^{t}$;
- Column weight of the parity-check matrix, $\gamma = 2^{t}$.

It is important to note that the rows of H are not necessarily linearly independent, and therefore the number of rows do not necessarily represents the rank of the H matrix. The rank of H is $n-k$ which makes the code of this matrix linear code. Since the matrix is $n \times n$, the implementation has n syndrome bits instead of $n-k$. The $(2^{2t}-1)\times(2^{2t}-1)$, parity-check matrix H of an EG Euclidean geometry can be formed by taking the incidence vector of a line in EG and its $2^{2t}-2$ cyclic shifts as rows; therefore this code is a cyclic code.

FSD-ECC Proof for EG-LDPC

In this section, we prove that EG-LDPC codes have the FSD-ECC property.

Theorem II: Type-I 2-D EG-LDPC codes are FSD-ECC.

Proof: Let be an EG-LDPC code with column weight and minimum distance d .We have to show that any error vector of weight $0 < e \leq d-1$ corrupting the received encoded vector has syndrome vector of weight at least $d-e$.

Now a specific bit in the syndrome vector will be one if and only if the parity-check sum corresponding to this syndrome vector has an odd number of error bits present in it. Looking from the Euclidean geometry perspective, each error bit corresponds to a point in the geometry and each bit in the syndrome vector corresponds to a line. Consequently, we are interested in obtaining a lower bound on the number of lines that pass through an odd number of error points. We further lower bound this quantity by the number of lines that pass through exactly one of the error points. Based on the definition of the Euclidean geometry, γ lines pass through each point; so error points potentially impact γe lines. Also at most one line connects two points. Therefore, looking at the e error points, there are at most $\binom{e}{2}$ lines between pairs of error points. Hence, the number of lines passing through a collection of these points lower bounded by $\gamma e - \binom{e}{2}$. Out of this number, at most $\binom{e}{2}$ lines connect two or more points of the error points. Summarizing all this, the number of lines passing through exactly one error point, which gives us the lower bound on the syndrome vector weight, is at least $\gamma e - 2\binom{e}{2}$.

From the code properties introduced in Section A and knowing that $d = \gamma + 1$, we can derive the following inequality

$$|s(c_e)| \geq \gamma e - 2\binom{e}{2} = e(\gamma + 1 - e) = e(d-e)$$

$\geq d-e$ When $e > 0$

The previous inequality says that the weight of the syndrome vector of a codeword with e errors is at least $d-e$ when $e > 0$ which is the required condition of Theorem (I). Therefore, EG-LDPC is FSD-ECC.

III. IMPLEMENTATION

FAULT-TOLERENT MEMORY SYSTEM

We outline our memory system design that can tolerate errors in any part of the system, including the storage unit and encoder and corrector circuits using the fault-secure detector. For a particular ECC used for memory protection, let E be the maximum number of error bits that the code can correct and D be the maximum number of error bits that it can detect, and in one error combination that strikes the system, let e_e, e_m, and e_c be the number of errors in encoder, a memory word, and corrector, and let e_{de} and e_{dc} be the number of errors in the two separate detectors monitoring the encoder and corrector units. In conventional designs, the system would guarantee error correction as long as $e_m \leq E$ and $e_e = e_c = 0$. In contrast, here we guarantee that the system can correct any error combination as long as $e_m \leq E$, $e_c + e_{dc} \leq D$, nd $e_m + e_c + e_{dc} \leq D$. This design is feasible when the following two fundamental properties are satisfied

1) Any single error in the encoder or corrector circuitry can at most corrupt a single codeword bit

(i.e., no single error can propagate to multiple codeword bits);
2) There is a fault secure detector that can detect any combination of errors in the received codeword along with errors in the detector circuit. This fault-secure detector can verify the correctness of the encoder and corrector operation.

The first property is easily satisfied by preventing logic sharing between the circuits producing each codeword bit or information bit in the encoder and the corrector respectively. We define the requirements for a code to satisfy the second property.

An overview of our proposed reliable memory system is shown in Fig. 3.1 and is described in the following. The information bits are fed into the encoder to encode the information vector, and the fault secure detector of the encoder verifies the validity of the encoded vector. If the detector detects any error, the encoding operation must be redone to generate the correct codeword. The codeword is then stored in the memory. During memory access operation, the stored code words will be accessed from the memory unit. Code words are susceptible to transient faults while they are stored in the memory; therefore a corrector unit is designed to correct potential errors in the retrieved code words. In our esign (see Fig. 3.1) all the memory words pass through the corrector and any potential error in the memory words will be corrected. Similar to the encoder unit, a fault-secure detector monitors the operation of the corrector unit. All the units shown in Fig. 3.1 are implemented in fault-prone the only component which must be implemented in reliable circuitry are two OR gates that accumulate the syndrome bits for the detectors.

Fault Secure Detector

The core of the detector operation is to generate the syndrome vector, which is basically implementing the following vector-matrix multiplication on the received encoded vector c and parity-check matrix H

$$s = c.H^T$$

Therefore each bit of the syndrome vector is the product of c with one row of the parity-check matrix. This product is a linear binary sum over digits of c where the corresponding digit in the matrix row is 1. This binary sum is implemented with an XOR gate. Fig. 3.4 shows the detector circuit for the (15, 7, 5) EG-LDPC code. Since the row weight of the parity-check matrix is ρ, to generate one digit of the syndrome vector we need a ρ-input XOR gate, or $(\rho-1)$ 2-input XOR gates. For the whole detector, it takes $n(\rho-1)$ 2-input XOR gates. Table 3.1 illustrates this quantity for some of the smaller EG-LDPC codes. Note that implementing each syndrome bit with a separate XOR gate satisfies the assumption of Theorem I of no logic sharing in detector circuit implementation. An error is detected if any of the syndrome bits has a nonzero value. The final error detection signal is implemented by an OR function of all the syndrome bits. The output of this -input OR gate is the error detector signal.

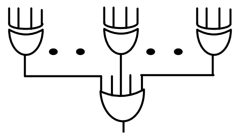


FIG: Fault-secure detector for (15, 7, 5) EG-LDPC code.

Parallel Corrector

For high error rates [e.g., when tolerating permanent defects in memory words as well], the corrector is used more frequently and its latency can impact the system performance. Therefore we can implement a parallel one-step majority corrector which is essentially n copies of the single one-step majority-logic corrector. Fig. 3.1 shows a system integration using the parallel corrector. All the memory words are pipelined through the parallel corrector. This way the corrected memory words are generated every cycle. The detector in the parallel case monitors the operation of the corrector, if the output of the corrector is erroneous; the detector signals the corrector to repeat the operation. Note that faults detected in a nominally corrected memory word arise solely from faults in the detector and corrector circuitry and not from faults in the memory word. Since detector and corrector circuitry are relatively small compared to the memory system, the failure rate of these units is relatively low.

Assuming our building blocks are two-input gates, γ number of ρ-input parity-check sums will require $\gamma \times (\rho-1)$ two-input XOR gates. The size of the majority gate is defined by the sorting network implementation. Table 3.1 shows the overall area of a serial one-step majority-logic corrector in the number of two-input gates for the codes under consideration. The parallel implementation consists of exactly n copies of the serial one-step majority- logic corrector.

IV. RESULTS AND ANALYSIS

Simulation Results

The behavioral simulation and post rout simulations waveforms for the fault secure encoder is shown in figure 5.1 and figure 5. 2. In the figure 5.1,the input is information vector and output is the detector output d which detects the errors in the encoder. First information vector is given to encoder it gives encoded vector as an output which is n-bit length. This encoded vector is given as input to the detector. Any error is present in encoded vector the detector output is '1'. If it is '0' encoded codeword is correct.

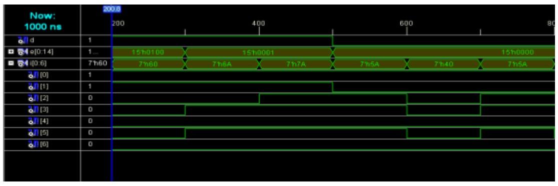


Figure: Behavioral simulation waveform for the fault secure encoder

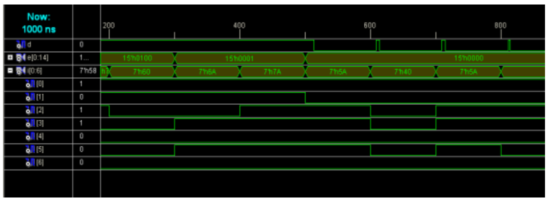


Figure: Post route simulation waveform for the fault secure encoder

The behavioral simulation and post route simulation waveforms for the fault secure memory system is shown in figure 5. 3 and figure 5. 4. In fig 3 inputs are I (information vector), clk, wen(write enable), ren(read enable), and e (error vector) to introduce an error. In this the encoded word is given to the memory for this if 'wen' is '1'(high) data is write into memory in a perticular address, here address line is the information vector. If 'ren' is high data is read and given as an output of memory. The memory output is combination of coded vector and error vector. This memory output is given as an input to the corrector which corrects the coded word. This corrected coded word is given to the detector to check whether coded word is correct or not.At the corrector side detector sinal is 'md'.

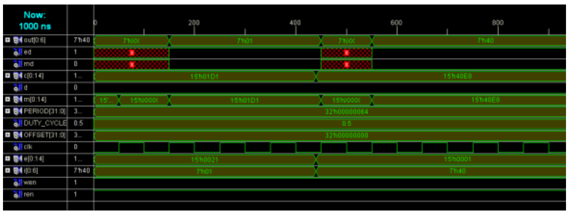


Figure: Behavioral simulation waveform for the fault secure memory

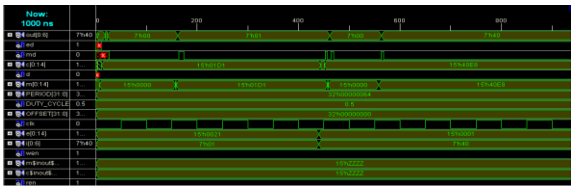


Figure:.Post route simulation waveform for the fault secure memory system

V. CONCLUSION

FPGA implementations of fault secure encoder and decoder for memory applications. Using this architecture tolerates transient faults both in the storage unit and in the supporting logic (i.e., encoder, decoder (corrector), and detector circuitries). The main advantage of the proposed architecture is using this detect-and-repeat technique we can correct potential transient errors in the encoder or corrector output and provide fault-tolerant memory system with fault-tolerant supporting circuitry. And also takes less area compared to other ECC techniques and in this architecture there is no need of decoder because we use systematic generated matrix.

We presented a fully fault-tolerant memory system that is capable of tolerating errors not only in the memory bits but also in the supporting logic including the ECC encoder and corrector. We used Euclidean Geometry codes**.**

We proved that these codes are part of a new subset of ECCs that have FSDs. Using these FSDs we design a fault-tolerant encoder and corrector, where the fault-secure detector monitors their operation. We also presented a unified approach to tolerate permanent defects and transient faults. This unified approach reduces the area overhead. Without this technique to tolerate errors in the ECC logic, we would required reliable (and consequently lithographic scale) encoders and decoders. Accounting for all the above area overhead factors, all the codes considered here achieve memory density of 20 to 100 GB/nm^2, for large enough memory (≥ 0.1 GB). Fault secure encoder and decoder for memory applications is to protect the memory and supporting logic from soft errors. The proposed architecture tolerates transient faults both in the storage unit and in the supporting logic. Scope for further work is instead of memory we use nano memory which provides smaller, faster, and lower energy devices which allow more powerful and compact circuitry.

REFERENCES

[1]. Shu Lin and Daniel J. Costello. Error Control Coding. Prentice Hall, second edition, 2004.

[2] R. G. Gallager, "Low-density parity-check codes", IRE Trans. Information Theory, vol. IT-8, no. 1, pp. 21–28, January 1962.

[3] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes", Electronics Letters, vol. 32, no. 18, pp.1645–1646, March 1997.

[4] R. J. McEliece, The Theory of Information and Coding. Cambridge, U.K. Cambridge University Press, 2002.

[5]. M. Sipser and D. Spielman, "Expander codes," IEEE Trans. Inf. Theory, vol. 42, no. 6, pp. 1710–1722, Nov. 1996.
[6]. D. E. Knuth, The Art of Computer Programming, 2nd ed. Reading, MA Addison Wesley, 2000.
[7]. Allen D. Holliday, Hamming Error-Correction Codes, February 17, 1994 (revised June 15, 2002; March 1, 2004).
[8]. 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.
[9]. H. Naeimi and A. DeHon, "Fault-tolerant nano-memory with fault secure encoder and decoder," presented at the Int. Conf. Nano-Netw., Catania, Sicily, Italy, Sep. 2007.
[10] S. J. Piestrak, A. Dandache, and F. Monteiro, "Designing fault-secure parallel encoders for systematic linear error correcting codes," IEEE Trans. Reliab., vol. 52, no. 4, pp. 492–500, Jul. 2003.
[11]. G. C. Cardarilli et al. Concurrent error detection in reed-solomon encoders and decoders. IEEE Trans. VLSI, 15842–826, 2007.
[12]. Hamming, Richard W., "Error Detecting and Error Correcting Codes," The Bell System Technical Journal 26, 2 (April 1950), 147–160.
[13]. Hill, Raymond. A First Course in Coding Theory. Clarendon Press,1986.
[14]. W. W. Peterson and E. J. Weldon, Jr., Error-Correcting Codes, 2nd Ed. Cambridge, MA M.I.T. Press, 1972.
[15] L. Edwards, "Low cost alternative to Hamming codes corrects memory errors," Cornput. Des., pp. 132-148, July 1981.
[16]. Y. Kou, S. Lin and M. Fossorier, Low density parity check codes based on _nite geometries a rediscovery and more," IEEE Trans. Inform. Theory, vol. 47, pp. 2711-2736, Nov. 2001.
[17]. Bhargava, V., "Forward Error Correction Schemes for Digital Communications," IEEE Communications Magazine, Vol. 21, No. 1, January 1983, pp 11-19. A non-mathematical overview of coding theory.
[18]. Geisel, W.A., "Tutorial on Reed-Solomon Error Correction Coding", NASA Technical Memorandum No. 102162, August, 1990.

Author's Profile

Mr. Y. Sharath, Post Graduated in Embedded systems (**M.Tech**) From **JNTU**, Hyderabad in 2011 and Graduated in Electronics and Communications Engineering **(B.Tech)** from JNTUH, in 2009. He is working as Assistant Professor in Department of Electronics and Communications Engineering in **Mallareddy Engineering College for women**, R.R Dist, AP, India. He has 3+ years of Teaching Experience. His Research Interests in Embedded systems and VLSI.

Ms. S.Srivani, Post Graduated in Embedded Systems (**M.Tech**) From **JNTU**, Hyderabad in 2009 and Graduated in Electronics &Communication Engineering (B.Tech) from JNTUH, in 2005. She is working as Assistant Professor in Department of Electronics &Communication Engineering in **Mallareddy Engineering College for Women**, R.R Dist, AP, India. She has 7+ years of Teaching Experience. Her Research Interests Include Network Security, Wireless Communication and Encryption & Decryption.

Mrs. O.M.Chandrika , Post Graduated in VLSI System Design (**M.Tech**) From **JNTU**, Hyderabad in 2012 and Graduated in Electronics & Communication Engineering (B.Tech) from JNTUH, in 2009. She is working as Assistant Professor in Department of Electronics & Communication Engineering in **Gouthami Institute of Technology & Management For Women**, Proddutur , Kadapa dist, ap, India. She has 2+ years of Teaching Experience. Her Research Interests Include Network Security, Wireless Communication and Encryption & Decryption and VLSI.

Mr. B.Gopi Krishna, Graduated in Electronics and Communication Engineering (**M.Tech**) From **JNTU**, Kakinada in 2013 and Graduated Post in Electronics and Communication Engineering (B.Tech) from JNTUH, in 2007. He has 4+ years of Teaching Experience. His Research Interests Include Embedded systems, Micro Controllers & Telecommunication and networking.