ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

Defect Secure Encoder and Decoder for very Small Memory Appliance

D.Venkatarami reddy[1], S. Chandravathi[2], M. Niharika[3], G. Divya Kumari[4]

Abstract: The project consists of an efficient VLSI implementation of fault secure encoder and decoder for memory appliances. A novel and efficient VLSI architecture is proposed and implemented for the fault secure memory. The VLSI architecture has been authored in Verilog code for fault secure encoder and decoder for memory and its synthesis was done with Xilinx XST. Xilinx ISE Foundation 9.1i has been used for performing mapping, placing and routing. For behavioral simulation and place and route simulation ISE simulator has been used. The Synthesis tool was configured to optimize for area and high effort considerations. The interest of the project work is an attempt to obtain fault secure memory architecture. This fault secure memory is used in computer systems mainly servers and in memory appliances and also used in military appliances.
A new approach is introduced to design fault-secure encoder and decoder circuitry for memory designs. The key novel contribution of this proposed system is identifying and defining a new class of Error-Correcting Codes (ECC) whose redundancy makes the design of Fault-Secure Detectors (FSD) particularly simple. This project present a fault-tolerant nano scale memory architecture which tolerates transient faults both in the storage unit and in the supporting logic (i.e., encoder and decoder (corrector) circuitry).The proposed system with high fault-tolerant capability is feasible when the following two fundamental properties are satisfied.

keywords: encoder,decodet, fault-tolerant detector, tolerentmemory,majoritylogiccorrector, synthesizing and optimizing

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 (i.e., encoder and decoder (corrector) circuitry.
Definition and 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.

Memory errors:

Memory errors are of two types, namely hard and soft.

- Hard errors are caused due to fabrication defects in the memory chip and cannot be corrected once they start appearing.
- Soft errors on the other hand are caused predominantly by electrical disturbances.

ERROR CONTROL CODING:

Error detection and correction or error controls 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.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

- 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 appliances.

1) 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(-correcting) 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.

Error-correcting codes are usually distinguished between convolution codes and block codes

- Convolution codes are processed on a bit-by-bit basis. They are particularly suitable for implementation in hardware, and the Viterbi decoder allows optimal decoding.
- Block codes are processed on a block-by-block basis. Early examples of block codes are repetition codes, Hamming codes and multidimensional parity-check codes. They were followed by a number of efficient codes, Reed-Solomon codes being the most notable due to their current widespread use. Turbo codes and low-density parity-check codes (LDPC) are relatively new constructions that can provide almost optimal efficiency.

Shannon's theorem is an important theorem in forward error correction, and describes the maximum information rate at which reliable communication is possible over a channel that has a certain error probability or signal-to-noise ratio (SNR). This strict upper limit is expressed in terms of the channel capacity. More specifically, the theorem says that there exist codes such that with increasing encoding length the probability of error on a discrete memory less channel can be made arbitrarily small, provided that the code rate is smaller than the channel capacity. The code rate is defined as the fraction *k/n* of *k* source symbols and *n* encoded symbols.

II.SYSTEM OVERVIEW

Linear Block Error Correcting Codes:

This section provides a brief introduction on linear block ECC's. Let I = (i0, i1, ..., i k−1) be k-bit information vector that will be encoded into n-bit codeword, C= (c0, c1, ..., c n−1). For linear codes the encoding operation essentially performs the following vector-matrix Multiplication.

$C = I \times G$

Where, G is a k × n generator matrix.

A code is a systematic code if any codeword consists of the original k-bit information vector followed by (n – k) parity-bits. With this definition, the generator matrix of a systematic code must have the following structure.

$G = [I\ X]$

Where, I is a k × k identity matrix and X is a k×(n−k) matrix that generates the parity-bits

The advantage of using systematic codes is that there is no need for a decoder circuitry to extract the information bits. The information bits are simply available in the first k bits of any encoded vector.

A code is said to be cyclic code if for any codeword c, all the cyclic shifts of C is still a valid codeword. A code is cyclic if the rows of its parity-check matrix and generator matrix are the cyclic shifts of their first rows.

The checking or detecting operation is the following vector-matrix multiplication.

$S = C \times H^T$

Where, H is an (n−k)×n Parity-Check matrix.

The (n − k)-bit vector S is called syndrome vector.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

A syndrome vector is zero if C is a valid codeword and non-zero if C is an erroneous codeword.

Parity Checking:

One simple way to detect errors is

1. Count the number of ones in the binary message.
2. Append one more bit, called the parity bit, to the message
3. set the parity bit to either 0 or 1, so that the number of ones in the result is even. For example, if the original message contained 17 ones, the parity bit would be a one; if there had been 16 ones, the parity bit would be a zero.
4. Count the number of ones in the received message, including the parity bit. The result will always be even if no errors were encountered. (This approach also works if the parity bit is set to make the count come out odd, as long as the receiver checks for an odd count.)

This simple check does have two limitations it only detects errors, without being able to correct them; and it can't detect errors that invert an even number of bits.

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

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

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.

Similarly, let the next from least significant bit of p, p_1, be an even parity check of positions 2, 3, 6, 7, 10, 11, … (in binary, 10, 11, 110, 111, 1010, 1011, …), and put p_1 in one of these positions. Those positions have a 1 in their second from least significant binary position number. The receiver checks the parity of these positions (including the position of p_1). If the result is 1, an error occurred in one of those positions, and if the result is 0, either no error occurred or an error occurred in some other position. Continuing, the third from least significant check bit, p_2, is made an even parity check on those positions that have a 1 in their third from least significant position number, namely positions 4, 5, 6, 7, 12, 13, 14, 15, 20, …, and p_2 is put in one of those positions.

Putting the check bits in power-of-two positions (1, 2, 4, 8, …) has the advantage that they are independent. That is, the sender can compute p_0 independently of p_1, p_2, … and, more generally, it can compute each check bit independently of the others.As an example, let us develop a single error correcting code for Solving (1) for m=3 gives with equality holding. This means that all 2^m possible values of the m check bits are used, so it is particularly efficient. A code with this property is called a perfect code.

This code is called the (7, 4) Hamming code, which signifies that the code length is 7 and the number of information bits is 4. The positions of the check bits p_i and the information bits u_i are shown below.

P0	P1	U3	P2	U2	U1	U0

7

Reed-Solomon codes

Reed-Solomon codes were developed in 1960 by Irving S. Reed and Gustave Solomon, who were then members of MIT Lincoln Laboratory. Their seminal article was entitled "Polynomial Codes over Certain Finite Fields." (Reed & Solomon 1960) When the article was written, an efficient decoding algorithm was not known. A solution for the latter was found in 1969 by Elwyn Berlekamp 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 appliance 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

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

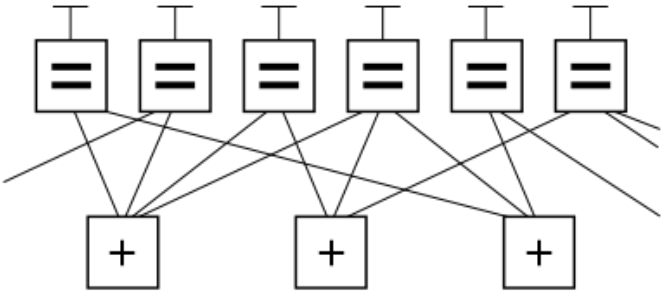
n = the block length in symbols
k = the uncoded message length in symbols
(n-k) = the parity check symbols (check bytes)
t = the number of correctable symbol errors
(n-k)=2t (for n-k even)
(n-k)-1 = 2t (for n-k odd)
Therefore, an RS code may be described as an (n,k) code for any RS code where, $n \le 2^m - 1$, and $n - k \ge 2t$.
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

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

The number of correctable symbol errors (t), and block length (n) is set by the user.
RS codes are widely implemented in digital storage devices and digital communication standards, though they are being slowly replaced by more modern low-density parity-check (LDPC) codes or turbo codes. For example, RS codes are used in the digital video broadcasting (DVB) standard DVB-S, but LDPC codes are used in its successor DVB-S2.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.
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, modulo 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

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

propagation used in practice lead to good approximations. In contrast, belief propagation on thebinary erasure channel is particularly simple where it consists of iterative constraint satisfaction.

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.

III.THE WORKING PRINCIPLE

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 chapter, 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.

2.3 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 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

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

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

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

$$\geq d - e \quad \text{When} \quad e > 0$$

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.

Efficiency of EG-LDPC

It is important to compare the rate of the EG-LDPC code with other codes to understand if the interesting properties of low-density and FSD-ECC come at the expense of lower code rates. We compare the code rates of the EG-LDPC codes that we use here with an achievable code rate upper bound (Gilbert- Varshamov bound) and a lower bound (Hamming bound). Table I shows the upper and lower bounds on the code overhead, for each of the used EG-LDPC. The EG-LDPC codes are no larger than the achievable Gilbert bound for the same k and d value and they are not much larger than the Hamming bounds. Consequently, we see that we achieve the FSD property without sacrificing code compactness.

IV.IMPLEMENTATION OF SYSTEM

FAULT-TOLERENT MEMORY SYSTEM OVERVIEW:

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 $e_m \leq E$, $e_c + e_{dc} \leq D$, and $e_m + e_c + e_{dc} \leq D$. 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 design (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

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

be implemented in reliable circuitry are two OR gates that accumulate the syndrome bits for the detectors

FPGA Design flow

The various steps involved in the design flow are as follows

1) Design entry.
2) Functional simulation.
3) Synthesizing and optimizing (translation) the design.
4) Placing and routing the design
5) Timing simulation of the design after post PAR.
6) Static timing analysis.
7) Configuring the device by bit generation.

Design entry:

The first step in implementing the design is to create the **HDL** code based on design criteria. To support these instantiations we need to include UNISIM library and compile all design libraries before performing the functional simulation. The constraints (timing and area constraints) can also be included during the design entry. Xilinx accepts the constraints in the form of user constraint (UCF) file.

Functional Simulation:

This step deals with the verification of the functionality of the written source code. ISE provides its own ISE simulator and also allows for the integration with other tools such as Modelsim. This project uses **ISE** simulator for the functional verification by selecting the option during project creation. Functional simulation determines if the logic in the design is correct before implementing it in a device. Functional simulation can take place at the earliest stages of the design flow. Because timing information for the implemented design is not available at this stage, the simulator tests the logic in the design using unit delays.

Synthesizing and Optimizing

In this stage behavioral information in the HDL file is translated into a structural net list, and the design is optimized for a Xilinx device. To perform synthesis this project uses Xilinx XST tool. From the original design, a net list is created, then synthesized and translated into a **native generic object** (NGO) file. This file is fed into the Xilinx software program called **NGD Build**, which produces a logical **native generic database** (NGD) file.

Design implementation

In this stage, The **MAP** program maps a logical design to a Xilinx FPGA. The input to MAP is an NGD file, which is generated using the NGD Build program. The NGD file contains a logical description of the design that includes both the hierarchical components used to develop the design and the lower level Xilinx primitives. The NGD file also contains any number of NMC (macro library) files, each of which contains the definition of a physical macro. MAP first performs a logical DRC (Design Rule Check) on the design in the NGD file. MAP then maps the design logic to the components (logic cells, I/O cells, and other components) in the target Xilinx FPGA.

The output from MAP is an **NCD (Native Circuit Description**) file, and **PCF (Physical constraint file).**

- NCD (Native Circuit Description) file—a physical description of the design in terms of the components in the target Xilinx device.
- PCF (Physical Constraints File)—an ASCII text file that contains constraints specified during design entry expressed in terms of physical elements. The physical constraints in the PCF are expressed in Xilinx's constraint language.

After the creation of Native Circuit Description (NCD) file with the MAP program, place and route that design file using PAR. PAR accepts a mapped NCD file as input, places and routes the design, and outputs an NCD file to be used by the bit stream generator (BitGen).

The **PAR placer** executes multiple phases of the placer. PAR writes the NCD after all the placer phases are complete. During placement, PAR places components into sites based on factors such as constraints specified in the PCF file, the length of connections, and the available routing resources.

After placing the design, PAR executes multiple phases of the router. The router performs a converging procedure for a solution that routes the design to completion and meets timing constraints. Once the design is fully routed, PAR writes an NCD file, which can be analyzed against timing. PAR writes a new NCD as the routing improves throughout the router phases.

Synthesize options

The following properties apply to the Synthesize properties .using the Xilinx® Synthesis Technology (XST) synthesis tool.

Optimization Goal

Specifies the global optimization goal for area or speed.

Select an option from the drop-down list.

- Speed Optimizes the design for speed by reducing the levels of logic.
- Area Optimizes the design for area by reducing the total amount of logic used for design implementation.

By default, this property is set to Speed.

Optimization Effort

Specifies the synthesis optimization effort level.

Select an option from the drop-down list.

- Normal Optimizes the design using minimization and algebraic factoring algorithms.
- High Performs additional optimizations that are tuned to the selected device architecture. "High" takes more CPU time than "Normal" because multiple optimization algorithms are tried to get the best result for the target architecture.

By default, this property is set to Normal.

This project aims at Timing performance and was selected HIGH effort level.

Power Reduction

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

When set to Yes (checkbox is checked), XST optimizes the design to consume as little power as possible.
By default, this property is set to No (checkbox is blank).

Use Synthesis Constraints File

Specifies whether or not to use the constraints file entered in the previous property. By default, this constraints file is used (property checkbox is checked).

V.EXPERIMENTAL RESULTS

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.

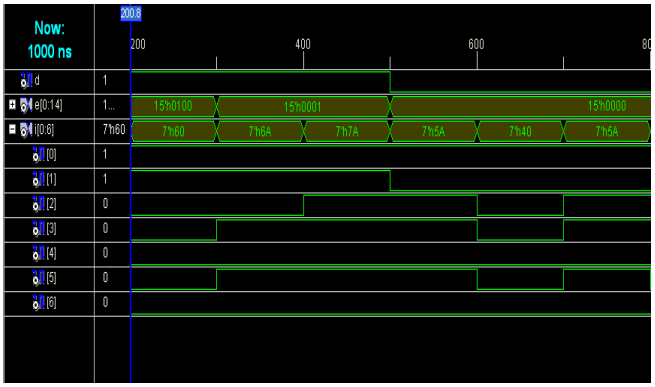


Figure 5.1.Behavioral simulation waveform for the fault secure encoder

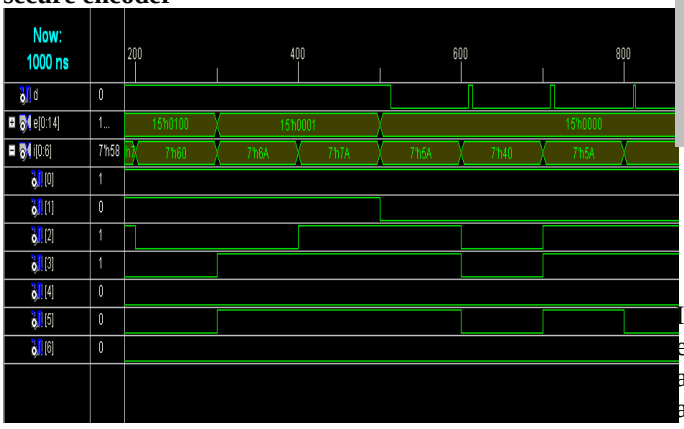
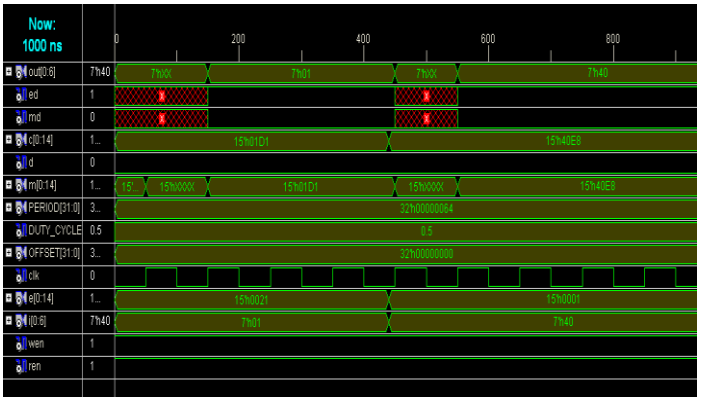


Figure 5. 2.Post route simulation waveform for the fault secure encoder

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'.



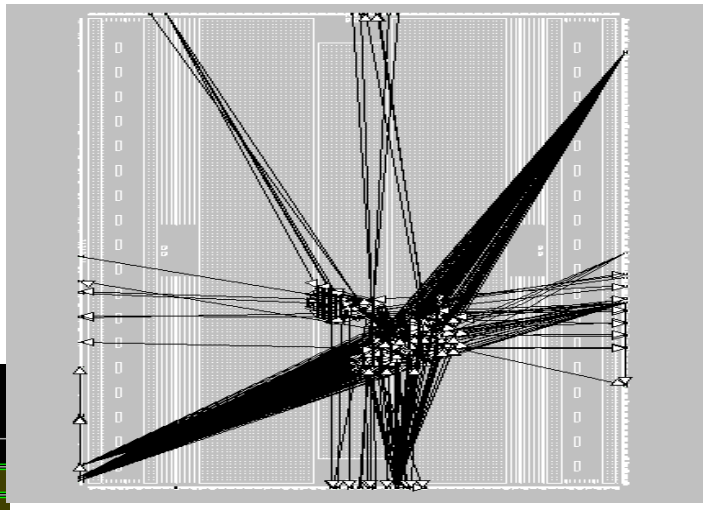


Figure 5.3 Floor plan of an fault secure encoder and decoder for memory

VI.CONCLUSION

In this project FPGA implementations of fault secure encoder and decoder for memory appliances. 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.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

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 ($\geq$ 0.1 GB). Fault secure encoder and decoder for memory appliances 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.

BIOGRAPHY

Mr.D.Venkatarami reddy,presently working in madhira institute of technology and sciences,He received the master of technology degree in Dr.paulraj engineering college-jntuh,he received the bachelor of technology degree in S.A engineering college-Anna university

Mrs.SOMIREDDY CHANDRAVATHI,, Post Graduated In He Received The Master Of Technology Degree SRI VENKATESHWARA ENGINEERING COLLEGE in Suryapet,JNTUH. received the bachelor of technology degree in SRI VENKATESWARA ENGINEERING COLLEGE,she Is Presently working in MADHIRA INSTITUTE OF TECHNOLOGY AND SCIENCES, KODAD, A.P, India

Mrs.MANDADI NIHARIKA,Graduated ANURAG ENGINEERING COLLEGE, Andhra Pradesh, India , And Is M.Tech SRI VENKATESHWARA ENGINEERING COLLEGE in Suryapet,A.P,India.She Is Presently working in NAGARJUNA INSTITUTE OF TECHNOLOGY AND SCIENCES, Miryalaguda ,A.P,India.

Mrs.GANTALA DIVYA KUMARI, Graduated SRI VENKATESWARA ENGINEERING COLLEGE,and M.tech: BOMMA INSTITUTE OF TECHNOLOGY AND SCIENCE in KHAMMAM,she Presently working in PULIPATI PRASAD INSTITUTE OF TECHNOLOGY & SCIENCE, KHAMMAM