# Implementing Proficient and Fault Secured System for Nano Applications

Udara.Yedukondalu, Chaparala Sree Lakshmi, Dr. A Jhansi Rani
Associate Professor, Assistant Professor, Professor of ECE

## Abstract

*The methods investigated are based upon majority decision decoding (which is a procedure for correcting a bit based upon how many failures it contributes to the syndrome). We discuss in detail an iterative method of decoding LDPC codes using a method. A new approach to design fault-secure encoder and decoder circuitry for memory designs is introduced. The key novel contribution of this project is identifying and defining a new class of error-correcting codes whose redundancy makes the design of fault-secure detectors (FSD) particularly simple. The parity-check Matrix of an FSD-ECC (fault secure detector - error correcting code) has a particular structure that the decoder circuit, generated from the parity-check Matrix, is Fault-Secure. LDPC codes 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.*

## 1. 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**.** A fault- tolerant nanoscale memory architecture which tolerates transient faults both in the storage unit and in the supporting logic (i.e., encoder, decoder, corrector and detector circuitries) is introduced.

**Transient faults**: When a node in the system loses its effective charge due to ionized particle hit or various sources of noises, it may cause the value of a node to be flipped in the circuit. However, the error does not permanently change the circuit, and it only generates a faulty bit value at the node that can last for one or few cycles. Feature-size scaling, faster clock cycles and lower power designs increase the transient fault rate. Feature-size scaling and voltage level reduction shrinks the amount of critical charges holding logical state on each node; this in turn makes each node more susceptible to transient faults, e.g., an ionized particle strike has Higher likelihood of being fatal as the critical charge is reduced in a node, which may cause a glitch or bit-flip.

## 2. Motivation The Impact of Providing Reliability for Supporting Logic:

It is significant to understand the impact of protecting the supporting logic on the system FIT (failure in time) rate. Figure 1 shows the FIT rate of the system decomposed into the contribution from the memory bank and the contribution from the supporting logic. The FIT in the supporting logic is without a fault-secure detector (i.e., any error in the supporting logic results an erroneous output, with worst-case analysis). Obviously the FIT of the whole system with no logic protection is the sum of the above two FITs, illustrated

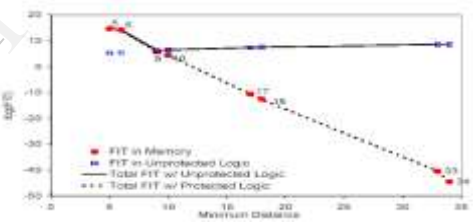


Figure 1: The impact of protecting logic on system reliability

With a solid line. For codes with minimum distance larger than 9, the FIT of the system with no logic protection is dominated by the FIT of the unprotected logic. Using codes with greater redundancy will decrease the FIT of memory bank; however, since the unprotected logic has a non-trivial FIT rate, increasing the code redundancy without protecting the logic does not decrease the FIT of the composite system.

## 3. Goal

A class of error-correcting codes (ECCs) that guarantees the existence of a simple fault-tolerant detector design should be identified. This class should satisfy 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. The parity-check Matrix of an FSD-ECC should have a particular structure that the decoder circuit, generated from the parity-check

Matrix, is Fault-Secure. The fault-secure detector should be designed, potential transient errors in the encoder are corrected using a corrector block and should provide a fully fault-tolerant memory system.

### 3.1 Low Density Parity Check codes
### Linear Block Codes

Since LDPC codes are a special case of linear block codes (LBC), in this section we will have an overview of this class of codes to set up a ground for discussing LDPC encoding and decoding. To encode, we need to map the information $i = [a_1, a_2, ....., a_K]$ into a codeword $c = [c_1, c_2, ....., c_K, c_{K+1}, ...., c_N]$ i.e. $c = f(i)$. Now the mapping can be a linear mapping. The canonical form of a linear transformation is

$$c = i * G$$

Where $G$ is a $K \times N$ matrix and all the code words {c} are distinct when the rank of G is K. The code rate of such a code is $K/N$ i.e. there are K information bits per N coded bits.

For a linear block code, the linear combination of any subset of code words is a codeword. We describe the encoding and decoding of LBC.

We first write the basis vectors (of size 1 X N) of G i.e., $[g_1, g_2, ....., g_k]$ of C as rows of matrix G ( $K \times N$ ).

Information $i = [a_1, a_2, ....., a_K]$ is encoded uniquely as,

$$c = a.G = [a_1, a_2, ....., a_K].G, \quad a_i \in GF(2)$$

The dual space of a linear code C is denoted by $C^T$ , which is a vector space of dimension (N-K). A basis $\{h_0, h_1, \ldots, h_{N-K-1}\}$ for $C^T$ can be found and used to construct Parity.

### Check Matrix H:

$$H = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-K-1} \end{bmatrix} = \begin{bmatrix} h_{0,0} & h_{0,1} & \ldots & h_{0,N-1} \\ h_{1,0} & h_{1,1} & \ldots & h_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N-K-1,0} & h_{N-K-1,1} & \ldots & h_{N-K-1,N-1} \end{bmatrix}$$

The parity check theorem: A vector c is a codeword in C if and only if C $H^T$ = 0.

The parity check matrix for a code also offers convenient means for determining the minimum distance of the code.

The problem of recovering the data block from a codeword can be greatly simplified through the use of systematic encoding. If c is the code word and G is the generator matrix then generator matrix can obtained as explained below. The theorem can be proved by noting that the rows of a generator matrix are linearly independent and that the column rank of the matrix is equal to the row rank.

$$G = [P|I_K] = \left[\begin{array}{cccc|cccc} p_{0,0} & p_{0,1} & \ldots & p_{0,N-K-1} & 1 & 0 & \ldots & 0 \\ p_{1,0} & p_{1,1} & \ldots & p_{1,N-K-1} & 0 & 1 & \ldots & 0 \\ p_{2,0} & p_{2,1} & \ldots & p_{2,N-K-1} & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{K-1,0} & p_{K-1,1} & \ldots & p_{K-1,N-K-1} & 0 & 0 & \ldots & 1 \end{array}\right]$$

When a data block is encoded using a systematic generator matrix, the data block is embedded without modification in the last K coordinates of the resulting codeword.

$$\begin{aligned} c &= mG \\ &= \begin{bmatrix} m_0 & m_1 & \ldots & m_{K-1} \end{bmatrix} [P|I_K] \\ &= \left[\begin{array}{cccc|cccc} c_0 & c_1 & \ldots & c_{N-K-1} & m_0 & m_1 & \ldots & m_{K-1} \end{array}\right] \end{aligned}$$

After decoding, the last K symbols are removed from the selected codeword and passed along to the data sink. The performance of the Gaussian elimination operations on a generator matrix does not alter the codeword set for the associated code. Column reordering, on the other hand, may generate code words that are not in the original code. If a given application requires that a particular codeword set be used and thus does not allow for column reordering, it is always possible to use some set of the coordinates other than the last k for the message positions. This can slightly complicate certain encoder/decoder designs.

### System overview

An overview of the proposed fault secure encoder and decoder is shown in figure1, and is as described below.

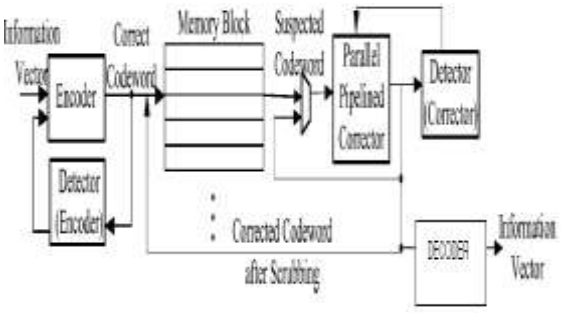


Figure 2: Block diagram of Fault Secure Encoder and Decoder.

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 code-word. The code-word is then stored in the memory. Later during operation, the stored code-word will be retrieved from the

memory unit. Since the code-word is susceptible to transient faults while it is stored in the memory, the retrieved code-word must be fed into the detector to detect any potential error and possibly to the corrector to recover any erroneous bits. In this design the corrector circuit has parallel structure and is implemented fully pipelined similar to the detector. All the memory words are pipelined through the corrector and then detector therefore; one corrected memory word is generated every cycle. The detector following the corrector would raise an error-detection flag only if a transient fault occurs in the corrector or detector circuitry. Due to the relative lower transient fault rate compared to the permanent defects and the relative small corrector and detector circuitry, this happens with low frequency. Therefore, the potential throughput loss of this system is low.

Data bits stay in memory for number of cycles and during this period each memory bit can be hit by transient fault with certain probability. Therefore, transient errors accumulate in the memory words over time. In order to avoid accumulation of too many errors in the memory words that surpasses the code correction capability, the system has to perform memory scrubbing. Memory scrubbing is periodically reading memory words from the memory, correcting any potential errors and writing them back into the memory

### 3.2 Design Structure:

In this section the design structure of the encoder, corrector, and detector units of the proposed fault secure encoder and decoder is provided.

### 3.2.1 Encoder:

An n-bit code-word c, which encodes k-bit information vector i is generated by multiplying the k-bit information vector with k × n bit generator matrix G, i.e., c = i · G. Figure 2shows the generator matrix of (15, 7) EG-LDPC code. All the rows of the matrix are cyclic shifts of the first row. This cyclic code generation does not generate a systematic code and the information bits must be decoded from the encoded vector, which is not desirable for our fault-tolerant approach due to the further complication and delay that it adds to the operation.

The generator matrix of any cyclic code can be converted into systematic form (G = [I : X])

| | $C_0$ | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ | $C_8$ | $C_9$ | $C_{10}$ | $C_{11}$ | $C_{12}$ | $C_{13}$ | $C_{14}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i_0$ | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $i_1$ | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| $i_2$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| $i_3$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| $i_4$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| $i_5$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| $i_6$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

Figure 3:The generator matrix of EG-LDPC code of (15, 7) in cyclic format

| | $C_0$ | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ | $C_8$ | $C_9$ | $C_{10}$ | $C_{11}$ | $C_{12}$ | $C_{13}$ | $C_{14}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i_0$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| $i_1$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| $i_2$ | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| $i_3$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| $i_4$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| $i_5$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| $i_6$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

Figure 4: The generator matrix of EG-LDPC code of (15, 7)

Figure 4 shows the systematic generator matrix to generate (15, 7) EG-LDPC code. The encoded vector, which is generated by the inner product of the information vector and the generator matrix, consists of information bits followed by parity bits, where each parity bit is simply an inner product of information vector and a column of X, from G = [I : X].

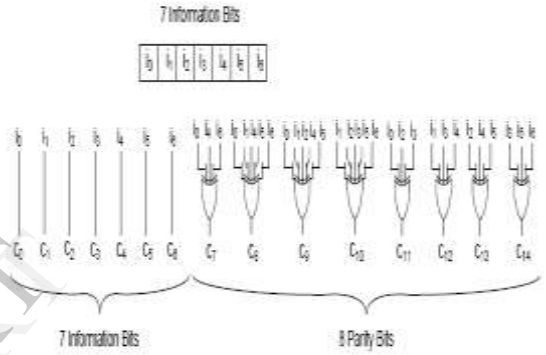


Figure5: The structure of an encoder circuit for (15, 7) EG-LDPC code.

Figure 5 shows the encoder circuit to compute the parity bits of the (15, 7) EG-LDPC code. In this figure i = (i0, ..., i6) is the information vector and will be copied to c0, ..., c6 bits of the encoded vector, c, and the rest of encoded vector, the parity bits, are linear sums (XOR) of the information bits. If the building block is two-input gates then the encoder circuitry takes 22 two input XOR gate. Since the systematic generator matrix of EG-LDPC and PG-LDPC codes does not have the standard row and column density, to compute the area of an encoder circuitry the corresponding systematic generator matrix has to be constructed. Once the systematic generator matrix is constructed the fan in size of the XOR gates can be determined by the column densities of the generator matrix.

### 3.2.2 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. c $H^T$ = S.

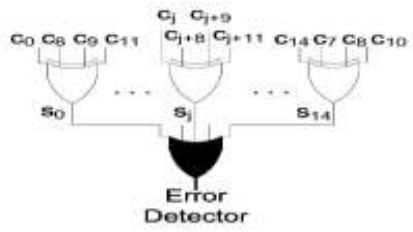


Figure 6: Fault-secure detector for (15, 7, 5) LDPC code.

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.6 shows the detector circuit for the (15, 7, 5) EG-LDPC code. Since the row weight of the parity-check matrix is, to generate one digit of the syndrome vector we need a –P Input XOR gate. 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.

### 3.2.3 Corrector:

**One-Step Majority-Logic Corrector**: One-step majority logic correction is the procedure that identifies the correct value of a each bit in the codeword directly from the received codeword; this is in contrast to the general message-passing error correction strategy, which may demand multiple iterations of error diagnosis and trial correction. Avoiding iteration makes the correction latency both small and deterministic. This technique can be implemented serially to provide a compact implementation or in parallel to minimize correction latency. This method consists of two parts: 1) generating a specific set of linear sums of the received vector bits and 2) finding the majority value of the computed linear sums. The majority value indicates the correctness of the code-bit under consideration; if the majority value is 1, the bit is inverted, otherwise it is kept unchanged.

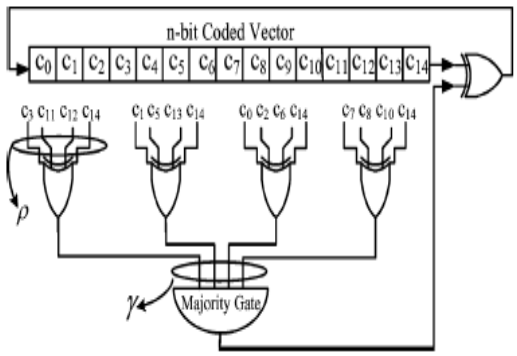


Figure 7:Serial one-step majority logic corrector structure

A linear sum of the received encoded vector bits can be formed by computing the inner product of the received vector and a row of a parity-check matrix. This sum is called Parity-Check sum. A set of parity-check sums is said to be orthogonal on a given code bit if each of the parity-check sums include the code bit but no other code bit is included in more than one of these parity-check sums.

## 4.Results

The main thing is to implementing fault tolerant system with existing whole large area is introduced I to “xc35100e” and vertex is sued for image acquisition. This chapter presents model-sim simulation and Xilinx synthesis results. Simulation waveforms are shown. It also gives summary of the work carried; this includes conclusions, performance analysis and scope for future work.

### 4.1 Top Level Fault Secure Encoder and Decoder:

The following figure gives the wave form that depicts the performance of Top level Fault Secure Encoder and Decoder.

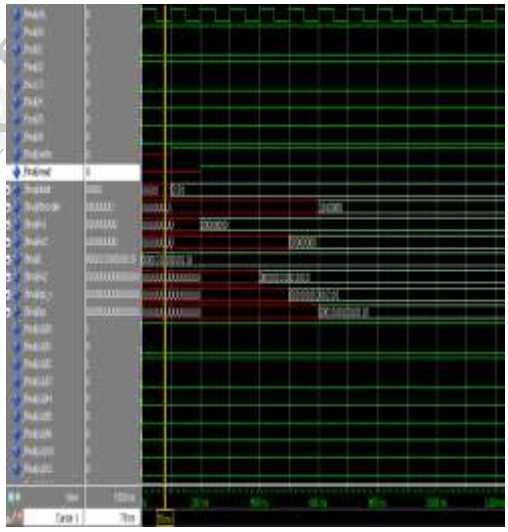

Figure 8:Simulation Result

## 5. Conclusion

In this report, a fully fault-tolerant memory system that is capable of tolerating errors not only in the memory but also in the supporting logic is designed. The LDPC codes are proved as part of a new subset of FSD-ECCs. Using these FSDs a fault-tolerant encoder and corrector is designed and synthesized in XC3S100E, finally results are shown.

## 6. References

[1] R. G. Gallager, “Low-Density Parity-Check Codes”. Cambridge, MA: MIT Press, 1963

[2] R. G. Gallager, “Low Density Parity Check Codes” , PhD dissertation, MIT, 1963 .

[3] R. J. McEliece, "The Theory of Information and Coding". Cambridge,U.K.: Cambridge University Press, 2002.
[4] W. E. Ryan, "An Introduction to LDPC Codes", in *CRC Handbook for Coding and Signal Processing for Recording Systems* (Ed. B. Vasic), CRC Press, 2004
[5] M. Karkooti, J.R. Cavallaro, "Semi-Parallel Reconfigurable Architectures for Real-Time LDPC Decoding", ITCC 2004.
[6] H. Zhong, T. Zhang, "Design of VLSI Implementation-Oriented LDPC Codes", *IEEE Semiannual Vehicular Technology Conference* (VTC), Oct. 2003.
[7] E. Yeo, B. Nikolic, and V. Anantharam, "Architectures and Implementations of Low-Density Parity Check Decoding Algorithms*", IEEE International Midwest Symposium on Circuits and Systems,* August. 2002.
[8] H. Naeimi and A. DeHon, "Fault secure encoder and decoder for memory applications," in Proc. *IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Sep. 2007, pp. 409–417.
[9] S.L. Howard, V.C. Gaudet, and C. Schlegal, "Soft-Bit Decoding of Regular Low-Density Parity Check Codes", *IEEE Transactions on Circuits and Systems.*