International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 2, ISSN 2249 –071X

Secure Transmission for Nano-Memories using Fault Secure Encoder and Decoder

Harikiran Nallagopula
M. Tech. (VLSID),
Nimra College of Engineering & Technology,
Vijayawada, A.P., India

Tirmalarao
Asst. Professor, Dept. of ECE,
Nimra College of Engineering & Technology,
Vijayawada, A.P., India

***Abstract* - Traditionally, memory cells were the only circuitry susceptible to transient faults. The supporting circuitries around the memory were assumed to be fault-free. 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 be protected. 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. 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. Here introducing 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 and further quantify the importance of protecting encoder and decoder circuitry against transient errors.**

Key Words – Secure Transmission, Nano-Memories, Encoder, Corrector, Décor.

I. Introduction

Nanotechnology provides smaller, faster, and lower energy devices, which allow more powerful and compact circuitry; however, these benefits come with a cost—the nanoscale devices may be less reliable. Thermal- and shot-noise estimations [5, 6] alone suggest that the transient fault rate of an individual nanoscale device (e.g., transistor or nanowire) 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 nanoscale systems, we must protect both combinational logic and memory against transient faults. In the present work we introduce a fault-tolerant nanoscale memory architecture which tolerates transient faults both in the storage unit and in the supporting logic (i.e., encoder and decoder (corrector) circuitry).

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.

Our proposed system with high fault-tolerant capability is feasible when the following two fundamental properties are satisfied: 1) Any single error in the encoder or corrector circuitry can only corrupt a single codeword digit (i.e., cannot propagate to multiple codeword digits). 2) There is a Fault Secure detector (FSD) circuit which can detect any limited combination of errors in the received codeword or the detector circuit itself.

II. System Overview

An overview of proposed reliable memory system is shown in Fig. 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.

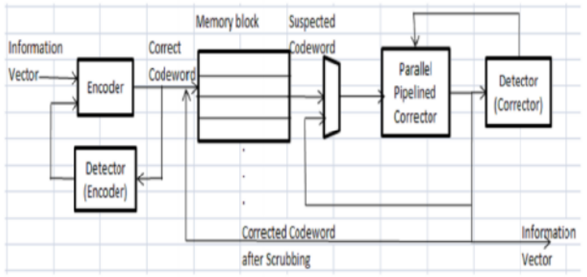

Fig. 1. Overview of proposed system

During memory access operation, the stored codewords will be accessed from the memory unit. Codewords 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 codewords. In our design (see Fig. 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. 1 are implemented in fault-prone, nanoscale circuitry; the only component which must be implemented in reliable circuitry are two OR gates that accumulate the syndrome bits for the detectors (shown in Fig. 2).

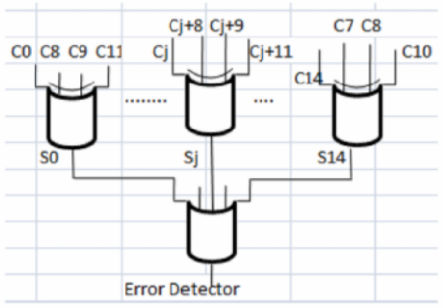

Fig. 2. Fault-secure detector for (15, 7, 5) EG-LDPC code.

Data bits stay in memory for a number of cycles and, during this period, each memory bit can be upset by a 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 any memory word that surpasses the code correction capability, the system must perform memory scrubbing. Memory scrubbing is the process of periodically reading memory words from the memory, correcting any potential errors, and writing them back into the memory (e.g., [6]).

III. Methodology

In this proposed system, 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, decoder (corrector), and detector circuitries).Particularly; we identify a class of Error-Correcting Codes (ECC's) that guarantees the existence of a simple fault-tolerant detector design. This class satisfies a new restricted definition for ECC's 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, 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.

We use the fault-secure detection unit to design a fault-tolerant encoder and corrector by monitoring their outputs. If a detector detects an error in either of these units, that unit must repeat the operation to generate the correct output vector. Using this retry technique, we can correct potential transient errors in the encoder and corrector outputs and provide a fully fault-tolerant memory system. The novel contributions of this proposed system include the following:

- A mathematical definition of ECC's which have simple FSD which do not requiring the addition of further redundancies in order to achieve the fault-secure property
- Identification and proof that an existing LDPC code (EG-LDPC) has the FSD property.

3.1 Memory System:

In this section, we present the details of the encoder, corrector, and detector units of our proposed fault-tolerant memory system.

3.1.1. Encoder:

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.

3.1.2. Corrector:

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 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.

3.1.3. Memory block:

Data bits stay in memory for a number of cycles and, during this period, each memory bit can be upset by a 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 any memory word that surpasses the code correction capability, the system must perform memory *scrubbing*. Memory scrubbing is the process of periodically reading memory words from the memory, correcting any potential errors, and writing them back into the memory. To perform the periodic scrubbing operation, the normal memory access operation is stopped and the memory performs the scrub operation.

IV. Error Detection And Correction Process

4.1. Encoder:

An n -bit codeword C, which encodes a k-bit information vector I is generated by multiplying the k-bit information vector with a k x n bit generator matrix G. The code rate is defined as the fraction *k/n* of *k* source symbols and *n* encoded symbols.

i.e. $C = I\,.G$

EG-LDPC codes are not systematic 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. However, these codes are cyclic codes. A code is a systematic code if any codeword consists of the original k-bit information vector followed by (n – k) 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. 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

EG-LDPC has the following parameters for any positive integer t > 2. where *t* is the number of errors that the code can correct.

• 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 x n.
• Row weight of the parity-check matrix, $p = 2^t$.
• Column weight of the parity-check matrix, $y = 2^t$.

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 (n, k) linear code. Since the matrix is (n x 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*.

4.2 Memory Access with NanoScale Interface:

The nano-memory architecture introduced in [1] has a lithographic scale interface. For our fault-tolerant system the supporting logic is implemented at the sub-lithographic scale; we replace the lithographic scale interface with a sublithographic one. The main part of the interface is a demultiplexer designed with gate-able nanowires [2], shown in Figure 4(a). Each of the output nanowires (vertical wires) is gate-able by r/n nanowires of the memory rows, where r is the number of memory rows. For example, in Figure 4(a), the four topmost rows gate the first output nanowire. The deterministic lithographic wires selecting the memory rows are programmed to select exactly one of the r/n controlling nanowires of each output nanowire.

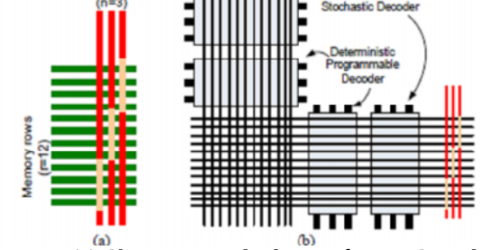

Fig. 4: (a) Shows a simple demux for n = 3, and r = 12.
(b) Shows integration of this demux with nano-memory.

This way at most one nanowire is driven high and can actually gate the controllable region. With this design, on each read, the demultiplexer selects n rows and presents them to the output nanowires; these outputs are then routed to the encoder, corrector and detectors. Figure 4(b) shows the integrated demultiplexer with the nano-memory system. With a few additional transformations, the demultiplexer can be statistically assembled similar to the restoration array used by the nanoPLA [3, 4].

V. Conclusion

In this paper we have presented a fault-tolerant nanotechnology memory system that tolerates faults in the encoder, corrector and detector circuitry as well as the memory. We use Euclidean Geometry codes with a fault-secure detector to design this memory system. 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. Euclidean Geometry 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.

References

[1] A. DeHon. Deterministic Addressing of Nanoscale Devices Assembled at Sublithographic Pitches. IEEE Transactions on Nanotechnology, 4(6):681–687, 2005.
[2] Y. Cui, X. Duan, J. Hu, and C. M. Lieber. Doping and Electrical Transport in Silicon Nanowires. Journal of Physical Chemistry B, 104(22):5213–5216, June 8 2000.
[3] A. DeHon. Law of Large Numbers System Design. In S. K. Shukla and R. I. Bahar, editors, Nano, Quantum and Molecular Computing: Implications to High Level Design and Validation, chapter 7, pages 213–241. Kluwer Academic Publishers, Boston, 2004.
[4] A. DeHon. Nanowire-Based Programmable Architectures. ACM Journal on Emerging Technologies in Computing Systems, 1(2):109–162, 2005.
[5] J. Kim and L. Kish. Error Rate In Current-Controlled Logic Processors With Shot Noise. Fluctuation and Noise Letters, 4(1):83–86, 2004.
[6] M. Forshaw, R. Stadler, D. Crawley, and K. Nikoli´c. A Short Review of Nanoelectronic Architectures. Nanotechnology, 15:S220–S223, 2004.

Author's Profile

Harikiran Nallagopula
M. Tech. (VLSID),
Nimra College of Engineering & Technology,
Vijayawada, A.P., India

Tirmalarao
Asst. Professor, Dept. of ECE,
Nimra College of Engineering & Technology,
Vijayawada, A.P., India