International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 17 Issue 2 – SEPTEMBER 2015.

EFFICIENT MAJORITY LOGIC FAULT DETECTION WITH EUCLIDEAN GEOMETRY LOW DENSITY PARITY CHECK CODES FOR MEMORY APPLICATION

[#1] K.Gowthami, [*2] M.Rami Reddy

[#1] *M.Tech,VLSI, Srinivasa Ramanujan Institute of Technology, Affiliated to JNTUA, AP, India.*
[*2] *Associate Professor, Dept of ECE, Srinivasa Ramanujan Institute of Technology, Affiliated to JNTUA, AP, India.*

***Abstract*— New bottom-up techniques can build silicon nano wires (dimension < 10 nm) that exhibit remarkable electronic properties, but with current assembly techniques yield very high defect and fault rates. Nano devices built using these nano wires have static errors that can be addressed at fabrication time by testing and reconfiguration, but soft errors are problematic, with arrival rates expected to vary over the lifetime of a part. In this paper, we propose using a special variant of low-density parity codes (LDPCs) Euclidean Geometry LDPC (EG-LDPC) codes to enable dynamic changes in level of fault tolerance. Apart from high error correcting ability and sparsity, a special property of EG-LDPC codes enables us to dynamically adjust the error correcting capacity for improved system performance (e.g., lower power consumption) during periods of expected low fault arrival rate. We present a system architecture for nano memory based on nano PLA building blocks using EG-LDPCs, and an analysis of its fault detection and correction capabilities.**

I. INTRODUCTION

The continued scaling of photolithographic fabrication techniques down to 32 nm and beyond face enormous technology and economic barriers. Self-assembled devices such as silicon nano wires or carbon nano tubes show promise to not only achieve aggressive dimensions, but to help address power and other gating issues in sys-tem architecture, while potentially helping contain rampant increases in fabrication capital costs. However, assembling high-quality large-scale nano electronic circuits (e.g., with Langmuir-Blodgett or related methods) has proven challenging. Among the major challenges are extremely high defect and fault rates in assembled devices. Apart from fabrication errors, nano scale devices are also more prone to soft errors than micro scale devices. Current-day micro scale devices (e.g., gates, PLAs, memories) constructed using top-down lithographic techniques have error rates of less than 1%. But computing and storage components built using nano scale elements (e.g., bistable and switchable organic molecules, carbon nano tubes, single-crystal semiconductor nano wires) may have an order of magnitude higher rates of faults (as high as 10%) . Static defects can be handled using testing and reconfiguration , though this presents increasing challenges as technology scales. Because of high soft-error rates, our envisioned nano memories would also need to employ online error correcting codes (ECC), as most modern memory subsystems already do.At the highest level, the work presented here aims to enable controlled reduction in fault tolerance during a part's lifetime. We work toward enabling the engineering of defect and fault tolerance, traded against system power and other key parameters, because soft error rates vary over the lifetime of complex electronic components. The bathtub curve" of failure rates classically begins with a very high rate of infant mortality, then smooths to a relatively low rate of failure for some time, and then slowly rises as the part nears the end of its life cycle. In addition, environmental stresses may change the expected fault rate (temperature, radiation levels). Classically, one must design error correcting codes (ECCs) to tolerate the maximum failure rate expected over the entire lifetime of a device, resulting in "wasteful" tolerance of "too many" faults during the bulk of the component's lifetime. But with dynamic coding, if we are able to diagnose and reconfigure out subcomponents after a time of high-fault-rate infant mortality, and we expect fault arrival rates to be greatly reduced for a significant period of time, we would like to turn off some of the power-hungry ECC circuits. Later, if fault rates were to climb again, we would wish to turn those ECC circuits back on.

At the nano scale, we desire an error correcting system that has (1) high error detecting and correcting ability, to tolerate relatively high soft error rates; (2) sparse modularity, so that the encoders and decoders can be synthesized using simple modular hardware; and (3) dynamic error-correcting capability, to be able to trade off between error correction and system performance in the presence of variable fault rates. In this paper, we propose

the use of low-density parity codes (LDPCs) for nano memories that satisfy all these properties. We propose the use of a variant of a particular type of LDPC code, Euclidean Geometry (EG) LDPC, where the H -matrix can be interpreted as an incidence matrix in a finite geometrical space spanned by m-tuples over GF (2^s). EG-LDPCs are decodable using a one-step majority decoder, thus making it unnecessary to have complex iterative decoders. EG-LDPC codes are sparse, and their encoding and parity check matrices have a regular modular structure, which enables the error correction rate to be changed dynamically. The sparseness (enabling low fan-in circuit implementation) and modularity make the dynamic EG-LDPC code easy to implement using nano scale hardware.

Various nano devices have been proposed in the literature (e.g., nano wire-based PLA, quantum nano dots) to serve as the basic building block of memory and com-putation elements. In this paper, we propose a design using nano wire crossbar arrays built using flow techniques , which can serve as memory cores, programmable PLA planes, and programmable crossbar interconnects. We analyse the properties of EG-LDPC codes that make it suitable as ECC for nano memory, and suggest how the coder and decoder circuits can be efficiently fabricated using nano PLA memory units and gates. We also consider that the encoder and checker circuits themselves can suffer faults, since they may be fabricated out of fault-prone nano scale components. We provide an analysis of fault detection and correction capacities of nano memories with EG-LDPC codes, and design an overall system architecture based on nano PLA building blocks.

II. OVERALL MODEL

Figure 1 shows the overall system architecture of the nano memory with EG-LDPC ECC. During a write operation, the incoming word to be stored in memory is encoded by the encoder and the code word is stored in memory. During a read operation, a code word is retrieved from the memory and checked by the checker unit. Finally, the majority logic unit decodes the syndrome and does the error correction. The

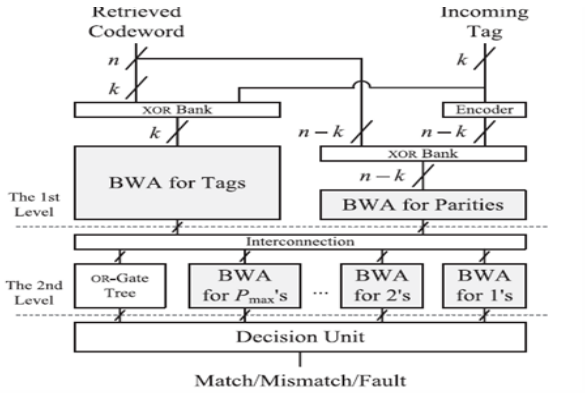

Fig. 1. Overall architecture of ECC .

controller unit controls the error detection and correction capability of the ECC unit. We now describe these different components and their implementation using nanoPLA components.

Encoder and Checker: As explained in Section 1, each submatrix H_i in H can be expressed in the form P_i H_1, where P_i is a permutation matrix. Note that since LDPC is a linear code and H_1 is in the standard form, the generator matrix G can also be expressed in a block-modular form as H , such that $G^T = (G^T_1\ G^T_2\ .\ .\ .\ G^T_\gamma)$ and each sub matrix $G_i = P_i\ G_1$. The modular design of the encoder circuit, the hardware implementation of G – a tree of nano PLA XOR gates implement the encoder block G_1, and a reconfigurable nano PLA crossbar implements the permutation block P_i . The same modular structure can be used for implementing the checker circuit.

Decoder and Controller: The majority-logic (ML) decoder requires two key hardware units: XOR and majority, which can both be implemented using nano PLA components. A controller circuit is used to generate the control signals for the selective enabling and disabling of the encoder, checker, and decoder components, to turn off unused hardware modules and thereby control the level of error correction in the dynamic code. Since we need only one controller for a complete memory bank, this module can be implemented using micro-level circuitry.

Dynamic Error Correction: Since the H (G) matrix is modular, the controller can selectively enable modules of the ECC circuit at different times of the operation, thereby changing γ and giving the required dynamic error correcting capability. This selective enabling and disabling of the decoder, checker, and encoder blocks by the controller can give significant power savings during normal operations of the ECC nano memory, particularly if failed components can be configured out of the system.

III. BACKGROUND

We outline concepts that will be useful in understanding our main design and architecture for ECC in nano memory based on EG-LDPC codes.

3.1 LDPC codes

LDPC codes have several advantages, which have made them popular in many communication applications: (1) low density of the encoding matrix, (2) easy iterative decoding, (3) generating large code words that can approach Shannon's limit of coding .

An LDPC code is defined as the null space of a parity check matrix H that has the following properties :

1. Each row has ρ number of 1's.

2. Each column has γ number of 1's.

3. The number of 1's that are common between any two columns (λ) is no greater than 1, i.e., λ = 0 or 1.

4. Both ρ and γ are small compared to the length of the code and the number of rows in H .

As both ρ and γ are very small compared to the code length and the number of rows in the matrix H , H has a low density of 1's. Hence H is said to be a low-density parity check matrix and the code defined by H is said to be a low-density parity check code.

The density of H (r) is defined to be the ratio of the total number of 1's in H to the total number of entries in H — in this case r = ρ/n = γ/J , where J is the number of rows in H . This kind of LDPC code is said to be a (γ, ρ)-regular LDPC code. If the weights of all the columns or rows in H are not the same, then it is called an irregular LDPC code.

3.2 Decoding

Since EG-LDPC is a finite geometry cyclic code , it can be decoded effectively using one-step majority logic (ML) decoding. Here, we explain how that decoding is performed.

Let the n-bit faulty code word r be the sum of the original code word c and the error word e, where e is 1 at the locations at which the errors/faults have occurred, 0 otherwise:

$$r = c + e.$$

To find whether a particular error bit e_i is 1 or 0, the decoder first calculates parity equations, each of which calculates a check sum on some bits of r. A set of J parity equations is orthogonal on e_i if each of the J parity equations check e_i (i.e., e_i is included in the check sum of each parity equation), but no other error bit is checked by more than one parity equation. If a set of J parity equations is orthogonal on e_i, the ML decoding rule can be applied to decode e_i:e_i is decoded to 1 if the majority of the parity check sums of the J check sums is 1, otherwise e_i is 0.

Correct decoding of e_i using the ML decoding rule, applied on the outputs of the syndrome equations S_i, is guaranteed if the number of errors in e is less than γ/2. This can be repeated for every bit position i to estimate the error word e.

In this paper, we consider one-step ML decoding. However, the performance of the hard decision one-step ML decoding can be further improved using Gallager's iterative bit-flipping (BF) algorithm , which is able to correct errors when the number of errors exceeds γ/2. BF uses simple comparison, sum, and majority operations and is therefore easier to implement using nano-PLAs than the popular sum-product algorithms for decoding LDPC codes using belief-propagation, which requires real-number arithmetic and computation of logarithms.

IV. DYNAMIC LDPC IN NANOMEMORY

Dynamic ECC is a property of EG-LDPC codes that we will explore here in de-tail. We begin by outlining the motivation for having dynamic codes in nano architectures.

4.1 Motivation

Soft error rates vary over the lifetime of complex electronic components. The "bathtub curve" of failure rates classically begins with a very high rate of infant mortality, then smooths to a relatively low rate of failure for some period of time, and then slowly rises toward the end of the life cycle. In addition, environmental stresses may change the expected fault rate (e.g., temperature, radiation levels, nearby radio transmission). Classically, one must design error correcting codes (ECCs) to tolerate the maximum failure rate expected over the entire lifetime of a device, resulting in "wasteful" tolerance of "too many" faults during the bulk of the component's lifetime.

At the highest level, the work presented here aims to enable controlled reduction in fault tolerance during a part's lifetime. More precisely, we work toward enabling the engineering of defect and fault tolerance, traded against system power and other key parameters. For example, if we are able to diagnose and reconfigure out subcomponents after a time of high-fault-rate infant mortality, and we expect fault arrival rates to be greatly reduced for a significant period of time, we would like to turn off some of the power-hungry ECC circuits. Later, if fault rates were to climb again, we would wish to turn those ECC circuits back on.

In some cases, fault arrival rates are predictable. Some device families show a bathtub curve of fault arrival rates fairly smoothly . In some systems, fault rates follow a mission profile (e.g., altitude), and in others fault rates are strongly affected by easily observable environmental conditions (e.g., temperature). Also, if a system is able to detect the frequency and severity (number of bits) of errors in encoded words, and fault rates change slowly (with respect to clock cycle), then the system may be able to predict future fault rates to some degree. In these cases, we wish to enable the dynamic control of fault tolerance, which may allow one to shut down a subset of ECC circuits to save power.

V. PROPOSED APPROACH

For dynamic ECC, we will use a particular type of LDPC code – Gallager code – which is the original LDPC code

proposed by Gallager . The advantage of this code is that the decoding parity check matrix (H -matrix) of this code can be expressed in a special form:

$$H^T = (H_1^T H_2^T \ldots H_\gamma^T), \quad (1)$$

Where H_1 is a matrix in the standard systematic form [I : A] [4], and $H_2, \ldots, H_\gamma$ are permutations of H_1. A (γ, ρ)-regular Gallager code has a parity check matrix with column weight γ and row weight ρ, and can correct as many as $\gamma/2$ errors. The number of rows in the H -matrix is $J = k \times \gamma$, and the number of columns is $n = k \times \rho$.

In dynamic ECC, one would typically want the ECC circuit for the nano memory to have higher error correcting capacity during the initial and final stages, since faults decrease during the normal operation of the circuit as compared

For example, for an EG-Gallager code of length n = 64 constructed based on the Euclidean Geometry EG(2, 2^3), the different error correction capacities for varying values of k and γ are shown in Table 1. As the ECC requirement decreases from 4 to 1, k changes from 37 to 49 — this means that we can pack more bits into the same memory code word, since n remains the same. Such a code can be used very effectively in our dynamic coding scheme.

The selective enabling and disabling of decoder blocks by the controller can give some power savings during normal operations. However, significant power savings are possible only if parts of the memory can be switched off, too. For that, we propose a memory repacking scheme in our memory bank architecture, details of which are described next.

A desirable property in some nano memory applications is modularity of hard-ware design of encoder and decoder. This makes the hardware design quite simple, since one can optimize the design of a single module and use multiple such modules to create both the encoder and the decoder circuits.

We will show how a particular class of EG-LDPC code, type-II, has this desirable modularity property. Type-II EG-LDPC codes are quasi-cyclic, i.e., each row of their H -matrix can be obtained by shifting previous rows by a particular fixed number of positions c (if c = 1, the code is cyclic codes are a special case of quasi-cyclic codes). Now, any quasi-cyclic matrix can be equivalently put in a circulant form, where the H - matrix comprises a set of circulants . Note that a circulant is a square matrix where each row is a cyclic shift of the row above it (with wraparound of the top row), and each column is a cyclic

VI. CONCLUSIONS AND FUTURE WORK

We want to focus on several interesting extensions to this work. If there are line-level faults in the address lines or row/column lines, then it is useful to have block-level error correcting codes like Reed-Solomon instead of LDPC. If the whole memory block becomes unusable because of a fault, then it is useful to have distributed error correction in different memory blocks instead of a centralized error-correcting scheme. To that effect, we intend to explore striped RAID-type architecture for nano memory banks . In the nano domain, asymmetric and unidirectional faults might occur – we plan to investigate such error models, study the prevalence of these types of errors through fault simulations in models of nano computing, and handle such faults using corresponding error-correcting codes, e.g., asymmetric codes, unidirectional codes . Finally, we want to implement an actual prototype of the proposed architecture on the nano-PLA substrate and perform experiments with actual memory traces from different domains , injecting faults following different fault models.

REFERENCES

[1] D. A. Anderson and G. Metze. Design of totally self-checking checker for m-out-of-n codes. IEEE Trans. on Computers, C-22, March 1973.

[2] K. Andrews, S. Dolinar, and J. Thorpe. Encoders for block-circulant ldpc codes. In ISIT, 2005.

[3] V. Beiu, S. Aunet, J. Nyathi, R. R. Rydberg-III, and A. Djupdal. The van-ishing majority gate trading power and speed for reliability. In Proceedings of NanoArch, 2005.

[4] R. E. Blahut. Theory and Practice of Error Control Codes. Addison-Wesley, 1983.

[5] M. Butts, A. DeHon, and S. C. Goldstein. Molecular electronics: Devices, systems and tools for gigagate, gigabit chips. In Proceedings of the Inter-national Conference on Computer-Aided Design, pages 433–440, November 2002.

[6] S.-C. Chae and Y.-O. Park. Low complexity encoding of regular low density parity check codes. In Proceedings of VTC, 2003.

[7] A. Dehon. Nanowires-based programmable architectures. ACM Journal on Emerging Technologies in Computing Systems, 1(2):109–162, July 2005.

[8] A. Dehon and H.Naeimi. Seven strategies for tolerating highly defective fabrication. In IEEE Design and Test of Computers, volume 22, pages 306–315, July-August 2005.

[9] A. DeHon and K. Likharev. Hybrid cmos/nanoelectronic digital circuits: devices, architectures, and design automation. In ACM/IEEE International Conference on Computer-Aided Design (ICCAD), pages 375–382, 2005.

[10] D. Edenfeld, A.B. Kahng, M. Rodgers, and Y. Zorian. 2003 technology roadmap for semiconductors. Computer, 37(1):47–56, Jan 2004.