Mitigating Defective CMOS to Non-CMOS Vias in CMOS/Molecular Memories

Nor Zaidi Haron[1,2], *Graduate Member IEEE* and Said Hamdioui[1], *Member IEEE*
1. Computer Engineering Laboratory, Delft University of Technology, The Netherlands
2. Faculty of Electronics and Computer Engineering, Universiti Teknikal Malaysia Melaka, Malaysia

***Abstract*— CMOS/Molecular (CMOL) memory is one of the emerging memory technologies that promises increased data storage, reduced power consumption and minimized fabrication complexity. The fabrication of these memories is based on the stacking of non-CMOS-based memory cell array on the top of CMOS-based peripheral circuits. Similarly to existing 3D technology, vertical vias are utilized to connect the two components. Because of their critical location and small in size, these CMOS to Non-CMOS Vias (CNVs) are prone to fabrication imperfection. A defective CNV may cause inaccessibility to the memory cell array, which in turn decreases the overall yield and/or reliability. This paper presents a modified CMOL architecture that mitigates faults due to defective CNVs. It is based on combining the Redundant Residue Number System (RRNS) error correction code (ECC) and interleaving. The number of banks interleaved in CMOL memories is determined by the ECC capability. Simulation results show that by setting an appropriate ECC capability with the associated number of banks, 95% to 100% mitigation of defective CNVs can be realized.**

I. Introduction

CMOS/molecular (CMOL) memories, proposed by Strukov and Likharev, are an emerging memory technology to replace existing solid-state memories as the main data storage [1]–[4]. Such memories are fabricated by combining two different device technologies, CMOS and non-CMOS devices, into one circuit. The use of non-CMOS devices as the memory cell array, which is stacked on top of the CMOS-based peripheral circuits, enables the fabrication of CMOL memories with up to 1 terabit/cm^2 data capacity [1], [2]. This stacked (3D) architecture is realized by the deployment of sharp-tip *CMOS to Non-CMOS Vias (CNVs)* that connect the peripheral circuits and the memory cell array. Existing field-emission arrays' silicon tips and metal are the potential candidates for the CNVs [3]. However, these tiny CNVs are prone to fabrication deficiency such as open, short, deform, misalignment, crack, etc [4]–[6]. Consequently, these defects might introduce permanent and intermittent faults, which in turn result in low yield and reliability of such memories.

Existing research has addressed defect and fault tolerance in CMOL memories with schemes like error correction codes (ECCs), sparing, and reconfiguration [1]–[4], [7]–[12]. However, using sparing requires the spare memory cells to be totally fault-free, which are hard to achieve in CMOL memories. Additionally, reconfiguration is a time-consuming process that needs testing circuitry and/or defect map to indicate the faulty memory cells. Furthermore, using well-known ECCs (e.g., Hamming, Bose-Chaudhuri-Hocquenghem (BCH), Reed Solomon (RS) or Redundant Residue Number System (RRNS)) alone cannot correct errors originated from CNVs. Therefore, other fault tolerance schemes to combine with suitable ECC is required to mitigate defective CNVs. Besides, the existing research mainly addresses faults occurred in the memory cell array but no one has addressed faults due to defective CNVs.

This paper presents a new CMOL memory architecture that aims to mitigate the faults due to defective CNVs. The architecture is based on a combination of two fault tolerance schemes, namely Redundant Residue Number System (RRNS) code and interleaving. In this architecture, the encoded input data (RRNS codeword) is interleaved before storing it in the multi-bank organization of CMOL memories. By doing this, the input data are dispersed to different memory banks. The defective CNVs that impact multiple interleaved stored data in one memory bank will only cause a low number of faults to the read data, which can be corrected by the decoder. The experiment simulations show that the proposed architecture achieves better mitigation performance of defective CNVs as compared to conventional architecture [1] at no extra cost.

The rest of the paper is organized as follows. Section II describes the structure, operation and defect types of CMOL memories. Section III reviews the basic theory of RRNS code and interleaving. Section IV explains the proposed defect-tolerant architecture for CMOL memories. Section V presents the experimental evaluation and the hardware implementation of the proposed architecture including a comparison with the conventional architecture. Section VI gives the conclusion.

II. CMOL Memories and Defect Types

Figure 1 illustrates the structure of CMOL memories and the enlarged view of their crossbar-based non-CMOS memory cell array [1], [3]. The non-CMOS circuit consists of two crossbar planes of nanowires (or carbon nanotubes) and molecular non-CMOS devices (e.g., organic molecule, single electron junction and phase change material [1], [3]) at each crossbars' junction. While the nanowire crossbars build up a matrix-like local interconnect of memory cell array, the molecular non-CMOS devices function as single memory cells. Because the state-of-the-art non-CMOS devices are still in their infancy stage to provide logical operations like inversion and amplification, scaled CMOS devices are used instead. The CMOS-based peripheral circuits like encoder, decoder, etc. are connected to the non-CMOS-based memory cell array through vertical sharp-tip CNV. The short CNVs connect a set of CMOS lines to the bottom layer of nanowires, whereas the long CNVs connect another sets of CMOS lines to the top layer of nanowires.

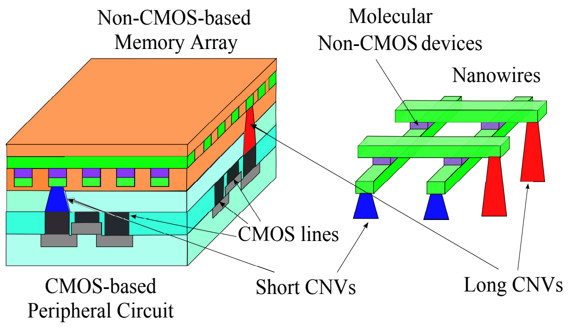

Fig. 1. The structure of CMOS/Molecular memories.

Since CMOL memories are extremely dense and the devices used to build up the circuits are incredibly tiny, they are prone to defects. The focus of this paper will be defects in CNVs; such defects may have different origins [1]–[5].

- The fabrication process variabilities (e.g., during polishing, bonding, via formation) of the CNVs might cause defects like open, short, deform, misalignment and crack. Such defects may cause permanent or intermittent faults are the consequences of these defects.
- High current density and high temperature give rise to electromigration to metal-based CNVs. Such defects may cause intermittent and eventually permanent faults are the results of this defect.

Figure 2 depicts the electrical equivalent circuit of the crossbar-based non-CMOS memory cell array shown in Fig. 1. These nanowire crossbars hold a group of molecular non-CMOS nanodevices (symbolized as resistive memory cells) representing a single CMOL memory word. The nanowires are connected to their corresponding CMOS lines through the CNVs. If one of the long CNVs is defective, then the corresponding memory cell cannot be accessed. This becomes worse if more than one long CNVs are defective where they may cause multiple faults that impact the memory word. The problem becomes even worse if the short CNV is defective; when this happens, the entire single memory word will be affected. Thus, the short CNVs are more critical than the long CNVs.

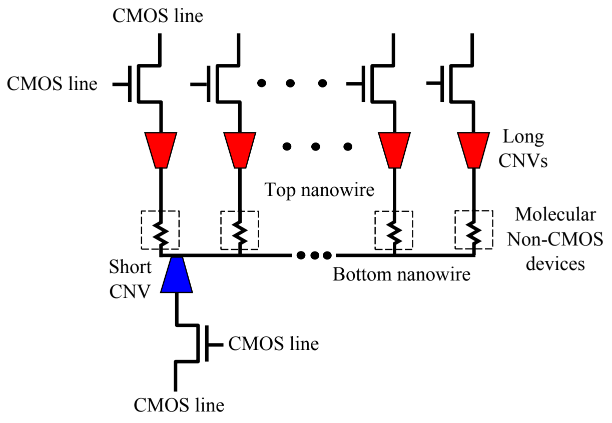

Fig. 2. Electrical equivalent circuit of one memory word in CMOL.

III. Background

This section reviews the fault tolerance schemes used in this work. First, it present the RRNS ECC, thereafter interleaving scheme.

A. Redundant Residue Number System Code

A Redundant Residue Number System (RRNS) (n,k) code is formed by a group of codewords. An RRNS codeword consists of a group of k-symbol dataword (called *non-redundant residues* x_i), and a group of $(n–k)$-symbol checkword (called *redundant residues* x_j) where $1{\leq}i{\leq}k$ and $k{+}1{\leq}j{\leq}n$ as shown in Fig. 3 [14]. The number of residues appended as the checkword determines the number of erroneous residues that can be corrected; this defines as error correction capability $t{=}\frac{(n-k)}{2}$. RRNS code is suitable to correct cluster faults.

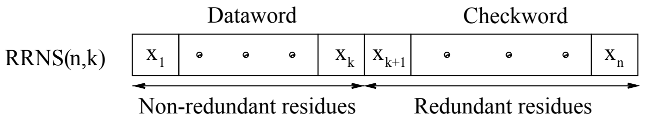

Fig. 3. The structure of RRNS codes.

RRNS encoding is a modulo operation of the input data to predefined moduli [14]. RRNS decoding is a single or two steps process: *detection* and *correction*. The detection process reads stored data (n residues) and compares to a predefined legitimate range. If the data is less than the legitimate range, then it is valid; hence, it is read out of the memory without any correction. Conversely, if the data is equal or more than the legitimate range due to faults, then the correction is eventually executed. The correction step performs an iterative operation similar to detection, yet using the remaining $(n-t)$ residues of the read data. Any corrected data within the legitimate range is regarded as the valid data and is read out of the memory.

B. Interleaving

Interleaving is a scheme to spread adjacent data in a codeword across different codewords [15]. This scheme effectively mitigate cluster fault that impact multiple adjacent data in an interleaved codeword. This is because they belong to different original codewords. By combining interleaving with other fault tolerance scheme (e.g., ECC), these erroneous bits can be corrected. E.g., four (4,2) RRNS codewords **C** are interleaved resulting into an interleaved stored data **C'** as follows. Note that, these codewords consist of n=4 and k=2; thus they can correct one erroneous residue.

$$\mathbf{C} = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{pmatrix} \qquad \mathbf{C'} = \begin{pmatrix} a_1 & b_2 & c_3 & d_4 \\ b_1 & c_2 & d_3 & a_4 \\ c_1 & d_2 & a_3 & b_4 \\ d_1 & a_2 & b_3 & c_4 \end{pmatrix}$$

With the interleaving scheme, up to four erroneous residues can be corrected if they occur in a single **C'** codeword while the other three **C'** codewords are error-free. E.g., if all residues of the first interleaved codeword $\mathbf{C'}_1$ (i.e., a_1, b_2, c_3, d_4) are erroneous, they are still correctable. This is because before decoding, the residues are de-interleaved into

C organization where the erroneous residues are re-organized into a diagonal direction from the top left (the highlighted cells). At that point, each **C** codeword has only one erroneous residue, which is correctable by the RRNS (4,2) decoder.

IV. Proposed CMOL Memories Architecture

This section describes the proposed architecture of CMOL memories based on the two schemes given in Section III. First, it explains the organization of conventional architecture followed by the proposed architecture.

A. Conventional Architecture

Figure 4 illustrates the architecture of the conventional CMOL memories storing RRNS codewords; this architecture is referred to as *conventional RRNS (C-RRNS)* [1]. For simplicity and coherency to the explanation given in Section III-B, only four memory banks that store four RRNS codewords are shown. Each codeword consists of two non-redundant residues (i.e., k=2) and two redundant residues (i.e., $(n-k)$=2), which render to a single residue correction capability (i.e., $t=\frac{4-2}{2}$=1). These four RRNS codewords, i.e., a_x, b_x, c_x, and d_x where $1 \le x \le 4$ are accessed simultaneously by asserting the appropriate address. They are written through the RRNS encoder and read through the RRNS decoder.

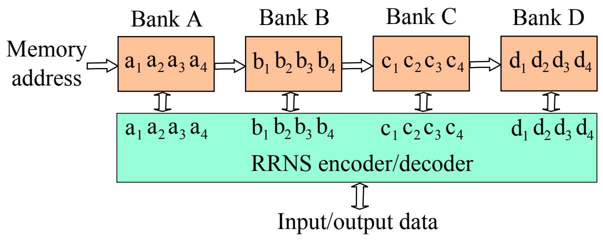

Fig. 4. Conventional CMOL memory architecture.

The shortcoming of this architecture is its low error correction capability for faults due to defective CNVs. E.g., assume that more than one long CNVs (see Fig. 2) of Bank A are defectives; this may results in more than one erroneous residues of codeword a_x. Since the the RRNS code in this example can only correct one residue, the correct codeword cannot be recovered.

B. Proposed Architecture

Figure 5 shows the proposed defect-tolerance architecture for CMOL memories storing interleaved RRNS codewords; this architecture is referred to as *Interleaved RRNS (I-RRNS)*. For readability, only the interleaved connections for the first codeword, i.e., a_x are shown. Interleaver and de-interleaver units are added after the encoder and before the decoder, respectively. This implies that the encoded data (RRNS codewords) are interleaved prior to be stored in the memory banks. Similarly, the RRNS codewords (stored data) are de-interleaved before decoding them. It is worth to note that the interleaving of codewords has to be in such a way that the stored residues in each bank fits within the word size of the bank. E.g., since $\|a_x\|$=$\|b_x\|$=$\|c_x\|$=$\|d_x\|$ where $\|a_x\|$ denotes the number of bits in codeword a_x, the size of $\{a_1, a_2, a_3, a_4\}$ (in Bank A of Fig. 4) is the same as that of $\{a_1, b_2, c_3, d_4\}$ (in Bank A of Fig. 5).

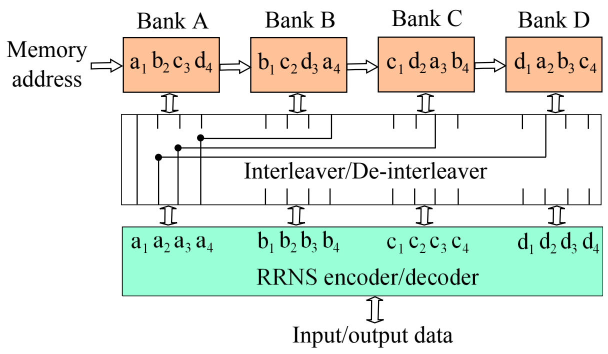

Fig. 5. Proposed CMOL memory architecture.

The proposed architecture has the advantage of being able to improve the error correction capability even for cluster faults, which are expected to be more dominant for CMOL memories [1], [13]. E.g., if one or more long CNVs in Bank A are defectives, then the proposed architecture will still be able to provide the correct read data. This is because each defective long CNV can impact at most one residue of a codeword, which is still within the error correction capability of used RRNS code. Even if a critical short CNV in Bank A is defective, the architecture will still recover the correct data. Again, this is because in the proposed architecture, only one residue is shared per memory bank.

In addition to cluster faults, the proposed architecture can also mitigate random faults as long as the defective CNVs do not impact two residues belonging to the same codeword. E.g. in Fig. 5, if the defective CNVs impact residues a_1 in Bank A and residue c_2 in Bank B, the read codeword a_x and c_x are still correctable. This is because each codeword has only one erroneous residue, which is within the correction capability of used RRNS.

V. Experimental Evaluation and Hardware Implementation

This section presents the important attributes in assessing defect/fault tolerance schemes, namely error correction performance, area and time overhead.

All designs including the encoder and decoder of the RRNS code, interleaver, de-interleaver, memories and fault injection were described using MATLAB script. The RRNS code is based on the moduli set $\{2^{\frac{d}{2}+1}-1, 2^{\frac{d}{2}+1}, 2^{\frac{d}{2}+1}+1, 2^{\frac{d}{2}+2}+1\}$ where d=64 [11], [12]. In the experiment, defective CNVs were assumed to be completely open. Hence, the cells associated with the defective CNVs were assumed to be faulty. Faults were randomly injected; and the fault rate can be up to 10% of the memory. The experiment is performed for two cases:

- Case 1: Defective short CNVs leading to the bottom layer of nanowires.
- Case 2: Defective long CNVs leading to the top layer of nanowires.

Figure 6 shows the simulation results of the error correction performance for C-RRNS and I-RRNS architectures both for the two cases mentioned above. Overall, I-RRNS outperforms C-RRNS for both defective CNV cases. Specifically for the defective short CNVs (i.e., Case 1), I-RRNS ensures 100%

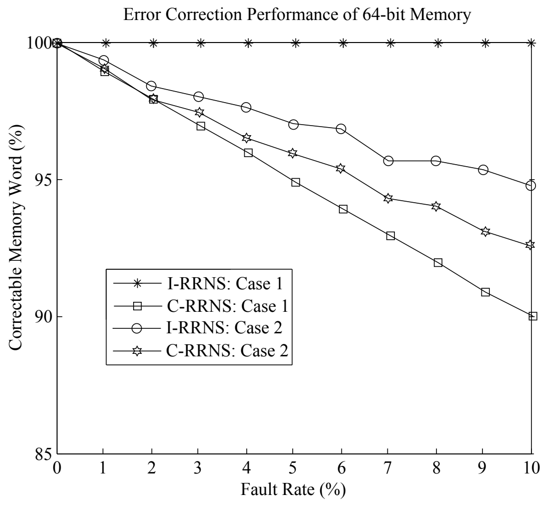

Fig. 6. Simulation results of C-RRNS and I-RRNS on 64-bit memory.

error correction performance for all fault rates. However, it is not the case for C-RRNS; the correction performance decreases linearly with the fault rate approaching 90% at 10% fault rate. For defective long CNVs (i.e., Case 2), I-RRNS is still better than C-RRNS. Interesting enough, the difference between the two architecture becomes more visible as the fault rate increases; e.g., I-RRNS performs about 1% better than C-RRNS at 5% fault rate, while it is about 2.5% at 10% fault rate.

I-RRNS is able to provide better error correction performance as compared to C-RRNS for both cases because:

- Case 1: Defective short CNVs may impact all four residues of I-RRNS codewords, however, the residues belong to different memory words. During decoding the erroneous residue can be corrected by the I-RRNS decoder. Contrarily, the erroneous four residues that form the C-RRNS codeword belong to the same memeory word; thus, they always beyond the error correction capability of the C-RRNS decoder.
- Case 2: Defective long CNVs may impact corrupted single, double, triple or even all four residues that form I-RRNS and C-RRNS codewords. As in the first case, I-RRNS decoder can correct these erroneous residues, but it is not the case for C-RRNS decoder.

To estimate the area and speed of the encoder and decoder of both C-RRNS and I-RRNS, the circuits were designed and synthesized at 90nm CMOS technology using Xilinx and Synopsys design tools. The synthesized circuits is further estimated based on 32nm CMOS technology [7]. Similar results were obtained for both C-RRNS and I-RRNS as shown in Table I. The table shows that the total area overhead of the encoder and decoder is << 1% of 1cm^2 CMOL memories [4]. In addition, their speed realized by both encoder and decoder is by far below the typical access time of CMOL memories, which can be 30ns [4]. It must be noted that, the interconenction delay might incurred in I-RRNS architecture. However, the delay is insignificant and is ignored in this paper.

TABLE I
SYNTHESIZED AND ESTIMATED OVERHEAD FOR 64-BIT ENCODER/DECODER FOR BOTH C-RRNS AND I-RRNS.

CMOS Technology	**Circuit**	**Area (μm^2)**	**Speed(ns)**
90nm (Synthesized)	Encoder Decoder	3172.38 20527.32	1.61 4.93
32nm (Estimated)	Encoder Decoder	198.27 1282.96	0.16 0.49

VI. CONCLUSION

This paper has presented a modified CMOS/Molecular (CMOL) memory architecture that mitigate faults due to defective CMOS to Non-CMOS Vias (CNVs). The architecture is based on combining the interleaving organization of CMOL's memory banks and to the error correction capability of Redundant Residue Number System. Depending on the appropriate setting of these two parameters, 95% to 100% mitigation of defective CNVs can be acheived. Furthermore, the proposed architecture incurs the same area and time overhead as compared to the conventional one. Future work is to investigate the feasibility of using redundant CNVs in CMOL memories.

REFERENCES

[1] D. B. Strukov and K. K. Likharev, "Prospects for terabit-scale nanoelectronic memories", *Nanotechnology*, vol. 16, no. 1, pp. 137–148, 2005.

[2] J. E. Green, C. J. Wook, A. Boukai, Y. Bunimovich, E. J. Halperin, et al., "A 160-kilobit molecular electronic memory patterned at $10^1$1 bits per square centimeter", *Nature*, vol. 445, pp. 414–417, 2007.

[3] K. K. Likharev, "Hybrid CMOS/Nanoelectronic Circuits: Opportunities and Challenges", *J. of Nanoelectronics and Optoelectronics*, vol. 3, no. 3, pp. 203–230, 2008.

[4] D. B. Strukov and K. K. Likharev, "Defect-Tolerant Architectures for nanoelectronic crossbar memories", *Nanotechnology*, vol. 7, no. 1, pp. 151–167, 2007.

[5] D. Tu, M. Liu, S. Haruehanroengra and W. Wang, "3D CMOL Based on CMOS/Nanomaterial Hybrid Technology", *Proc. of the 7th IEEE International Conference on Nanotechnology*, pp. 879–882, 2007.

[6] G. S. Snider and R. S. Williams, "Nano/CMOS architecture using a field-programmable nanowire interconnect", *Nanotechnology*, vol. 18, no. 3, pp. 1–11, 2007.

[7] F. Sun and T. Zhang, "Defect and Transient Fault-Tolerant System Design for Hybrid CMOS/Nanodevice Digital Memories", *IEEE Trans. on Nanotechnology*, vol. 6, no.3, pp. 341–351, 2007.

[8] C. M. Jeffery and R. J. O. Figueiredo, "Hierarchical Fault Tolerance for Nanoscale Memories", *IEEE Trans. on Nanotechnology*, vol. 5, no. 4, pp. 407–414, 2006.

[9] H. Naeimi and A. DeHon, "Fault Secure Encoder and Decoder for NanoMemory Applications", *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 4, pp. 473–486, 2009.

[10] S. Biswas, T. S. Metodi, F. T. Chong and R. Kastner, "A Pageable, Defect-Tolerant Nanoscale Memory System", *in Proc. of IEEE Int'l Symposium on Nanoscale Architecture*, pp. 85–92, 2007.

[11] N. Z. Haron and S. Hamdioui, "Using RRNS Codes for Cluster Faults Tolerance in Hybrid Memories", in *Proc. of IEEE Int'l Symposium on Defect and Fault Tolerance of VLSI Systems*,pp. 85–93, 2009.

[12] N. Z. Haron and S. Hamdioui, "Residue-based Code for Reliable Hybrid Memories", in *Proc. of IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 27–32, 2009.

[13] M. Mishra and S. C. Goldstein, "Defect tolerance at the end of the Roadmap", in *Proc. of International Test Conference*, vol. 1, pp. 1201–1211, 2003.

[14] F. Barsi and P. Maestrini, "Error correcting properties of redundant residue number systems", *IEEE Transactions of Computers*, vol. 2, no. 2, pp. 915–923, 1973.

[15] A.J. van de Goor. *Testing Semiconductor Memories, Theory and Practice*. COMTEX Publishing. Gouda. 1998.