Cost-Efficient Fault-Tolerant Decoder for Hybrid Nanoelectronic Memories

Nor Zaidi Haron[1,2] Said Hamdioui[1]
[1]Computer Engineering Laboratory, Delft University of Technology, The Netherlands
[2]Faculty of Electronic and Computer Engineering, Universiti Teknikal Malaysia Melaka, Malaysia
{N.Z.B.Haron[1,2], S.Hamdioui[1]}@tudelft.nl, zaidi@utem.edu.my[1,2]

***Abstract*— Existing work on fault tolerance in hybrid nanoelectronic memories (hybrid memories) assumes that faults only occur in the memory array and the encoder, not in the decoder. However, as the decoder is structured using scaled CMOS devices, it is also becoming vulnerable to faults. This paper presents a cost-efficient fault-tolerant decoder for hybrid memories that are impacted by a high degree of non-permanent clustered faults. Fault-tolerant capability is achieved by combining partial hardware redundancy scheme and on-line masking scheme based on Muller C-gates. In addition, the cost-efficient implementation of the decoder is realized by modifying the decoding sequence and implementing it based on time redundancy. Experimental results show that the proposed decoder is able to provide better reliability of the overall hybrid memory system, yet requires smaller area as compared to conventional decoder. For example, when assuming the fault ratio between decoder and memory array is 1:10 and at 10% fault rate, the proposed decoder ensures 1% higher reliability of the overall hybrid memory system. Moreover, the proposed decoder realizes 18.4% smaller area overhead for 64-bit word hybrid memory.**

I. Introduction

Recently, tremendous efforts have been made in exploring new computing paradigm to enhance the performance of memory chips. Referred to as *hybrid nanoelectronic memories* (hereafter is referred to as *hybrid memories*), this emerging paradigm shifts the functionality of data storage units to non-CMOS devices, yet still relying on CMOS devices to provide periphery tasks. Several research groups have proposed their hybrid memories of which the memory array is structured based on crossbar architecture [1]–[8]. The most prominent advantage of hybrid memories is the potential to provide 1Tbit/cm^2 chip area. On the other hand, hybrid memories are expected to suffer from massive numbers of *permanent* and *non-permanent (transient and intermittent)* faults leading to serious yield and reliability issues [1]–[3].

To tackle the reliability issues, several literatures propose to apply well-established fault tolerance schemes such as error correction codes (ECCs) [1], [2],[9]–[14], sparing [9], [12], reconfiguration [1], [2], [11]. However, these literatures assume that faults only occur in memory array and encoder but not in decoder. This assumption does not hold for hybrid memories because even at 130nm CMOS technology node, logic circuits have already exhibited almost similar transient faults rate to that of unprotected memories [15], [16]. Moreover, because of technology scaling, manufacturing process variabilities arise and might induces intermittent faults [17]. Therefore, it is inevitable to apply fault tolerance scheme to decoder as well to produce reliable hybrid memory system.

This paper presents a cost-efficient fault-tolerant decoder for hybrid memories that are impacted by a high degree of non-permanent clustered faults. The proposed decoder operates a modified *Redundant Residue Number System (RRNS)* code [14], which is implemented by combining *partial hardware redundancy* and *Muller C-gate* to achieve fault tolerance. Furthermore, to realize the cost-effectiveness, the decoding sequence of the modified RRNS code is modified and implemented based on time redundancy. Experimental results show that the proposed decoder is able to provide better reliability of the overall hybrid memory system, yet requires smaller area and shorter decoding latency (for long memory word) as compared to conventional decoder.

The rest of the paper is organized as follows. Section II proposes the cost-efficient fault-tolerant decoder. Section III presents the hardware implementation and experimental results. Section IV concludes this paper.

II. Cost-efficient Fault-Tolerant Decoder

This section describes the proposed cost-efficient fault-tolerant decoder. First, it reviews the an architecture of hybrid memories structured based on the modified RRNS code [14]. The architecture, referred to as D3R, assumes that no faults occur in the decoder. Second, it discusses the extension of D3R architecture by incorporating fault-tolerant decoder. Third, it presents a modified decoding procedure realizing a cost-efficient fault-tolerant decoder.

A. D3R Architecture

Figure 1 depicts the architecture of D3R hybrid memory [14]. The top part of the figure shows the memory array structured from non-CMOS devices, while the bottom part is the peripheral circuitry structured from CMOS. This architecture operates D3R codeword, a modified version of RRNS, that comprises of two codeword parts: (i) the original the codeword (C=DW+CW) and (ii) the duplicate codeword (C'=DW'+CW'), as illustrated in the memory array. These D3R codeword parts are encoded based on *low-cost moduli* set of m_1=$2^{\frac{d}{2}}-1$, m_2=$2^{\frac{d}{2}+1}-1$ and m_3=$2^{\frac{d}{2}+1}$ where d is the width of the input data [19]. The D3R codeword length is b_{D3R}=$b + b'$=$2\times(\lceil log_2(m_1)\rceil+\lceil log_2(m_2)\rceil+\lceil log_3(m_3)\rceil)$. Because each D3R codeword part has a single checkword, it

978-3-9810801-7-9/DATE11/©2011 EDAA

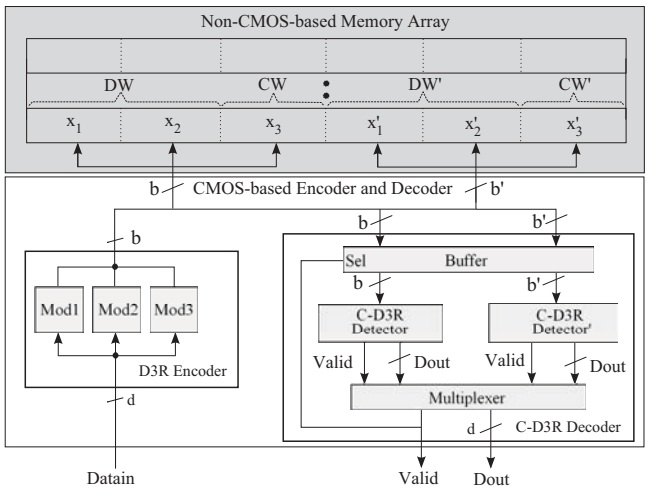

Fig. 1. D3R hybrid memory architecture.

TABLE I

RESIDUE SET FOR EACH ITERATION OF D3R CORRECTION PHASE.

Swapping Iteration	Residues	
	C	C'
1	x_1', x_2, x_3	x_1, x_2', x_3'
2	x_1, x_2', x_3	x_1', x_2, x_3'
3	x_1, x_2, x_3'	x_1', x_2', x_3

TABLE II

MULTIPLICATIVE INVERSES FOR CONVENTIONAL MRC.

Memory Word d	Moduli		Multiplicative Inverses
	$m_{(i-u)}$	m_i	$g_{(i-u)i}$
16	m_1=255	m_2=511	g_{12}=509
	m_1=255	m_3=512	g_{13}=255
	m_2=511	m_3=512	g_{23}=511
32	m_1=65535	m_2=131071	g_{12}=131069
	m_1=65535	m_3=131072	g_{13}=65535
	m_2=131071	m_3=131072	g_{23}=131071
64	m_1=4294967295	m_2=8589934591	g_{12}=8589934589
	m_1=4294967295	m_3=8589934592	g_{13}=4294967295
	m_2=8589934591	m_3=8589934592	g_{23}=8589934591

can detect a single erroneous residue. By duplicating them, D3R codeword ensures a valid output data if any part is error-free. Therefore, this code possesses error correction capability $t \leq 3$ residues, which is better than that of conventional RRNS code. Note that, conventional RRNS code that consists of six residues (two-residue dataword k, and four-residue checkword $(n–k)$) only posseses $t \leq \lfloor (n-k)/2 \rfloor = \lfloor (6-2)/2 \rfloor = 2$.

The architecture consist of a D3R encoder and a D3R decoder. The D3R encoder consists of three modulo circuits to encode a d-bit input to the corresponding residues x_1, x_2 and x_3 simultaneously. The D3R decoder, referred to as *Conventional D3R (C-D3R)* decoder, comprises of a buffer, two *detectors* and a multiplexer. Note that the name Conventional D3R (C-D3R) is to distinguish this decoder with the fault-tolerant D3R proposed in this work. The two detectors operate based on *hardware redundancy* where they decode the D3R codeword parts (C and C') *simultaneously* (see Table I). All required parameters for D3R decoding is given in Table II. Literature [14] can be referred for more details.

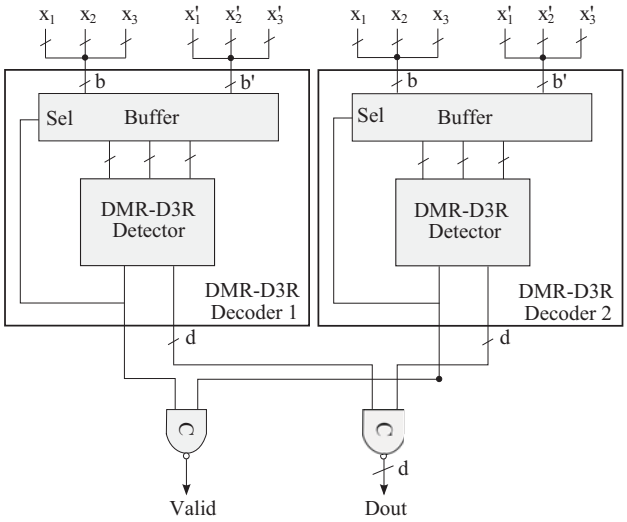

Fig. 2. Fault-tolerant DMR-D3R decoder.

B. Fault-Tolerant DMR-D3R Decoder

Figure 2 illustrates the block diagram of fault-tolerant decoder proposed in this work; it is referred to as *Double Modular Redundancy-D3R (DMR-D3R)*. As opposed to C-D3R decoder, DMR-D3R consists of *two* decoders: *DMR-D3R Decoder 1* and *DMR-D3R Decoder 2*. The output signals of both DMR-D3R decoders are voted using inverted C-gates. The C-gates are inverted to have the authentic output data because the original one produces opposite logic value to the input data when activated [18]. Because C-gates only change their output if both inputs are identical, these asynchronous logic gates are able to mask short period glitches. The possibility of two non-permanent faults to impact the same logic node in two different DMR-D3R decoders simultaneously is very low, i.e., $2\times(\frac{1}{d})^2$ where d is input bit length. For example, for 64-bit decoders, the possibility is $2\times(\frac{1}{64})^2 = 4.88\times10^{-4}$. This is where C-gate shows its superiority because even if there are many glitches, as long as they occur at different time and/or different logic nodes, the output is still unchanged.

Another difference of the proposed fault-tolerant DMR-D3R decoder is that it is structured based on time redundancy instead of hardware redundancy as in C-D3R. As shown in Fig. 2, each DMR-D3R decoder comprises of a single *detector* instead of two detectors in C-D3R [14]. The *DMR-D3R detector* decodes the D3R codeword parts *separately* (first C and then C'). Each time it reads a three-residue D3R codeword part (C or C'), converts it to binary data and compares it with the operating legitimate range 2^d-1 where d is input data length. If faults occur, the operation will be iterated until the valid decoded data is found; each iteration decodes the swapped residues given in Table I.

Because DMR-D3R operates based on time redundancy, it might incurs smaller area yet operates slower as compared to C-D3R that operates based on hardware redundancy. In order to achieve a cost-efficient implementation in terms of both area and time, a modification on the D3R decoding sequence has been made. This will be explained in the next subsection.

TABLE III
MULTIPLICATIVE INVERSES FOR MODIFIED MRC.

Memory Word d	Moduli		Multiplicative Inverses
	m_i	$m_{(i-u)}$	$g_{i(i-u)}$
16	m_3=512	m_2=511	g_{32}=1
	m_3=512	m_1=255	g_{31}=128
	m_2=511	m_1=255	g_{21}=1
32	m_3=131072	m_2=131071	g_{32}=1
	m_3=131072	m_1=65535	g_{31}=32768
	m_2=131071	m_1=65535	g_{21}=1
64	m_3=8589934592	m_2=8589934591	g_{32}=1
	m_3=8589934592	m_1=4294967295	g_{31}=2147483648
	m_2=8589934591	m_1=4294967295	g_{21}=1

C. Modified Decoding Procedure

The key idea is *to reverse the decoding sequence in such a way that the decoder executes the residues of the codewords in the opposite order* as compared to that of the conventional. In conventional RRNS, the decoding first executes the most significant residue x_1 and ends with the least significant residues x_3. However in this paper, it starts from the least significant residues x_3, and ends with the most significant residue x_1. This modification results in smaller decoding parameters and in turns realizes simpler circuits (i.e., smaller area and and faster decoding) than that of C-D3R.

Table III gives the parameters required for D3R decoding including the modular multiplicative inverses $g_{i(i-u)}$ where i and u are integers. It is interesting to note that the DMR-D3R detector requires smaller modular multiplicative inverses as compared to that of conventional decoding given in Table II. The modified $g_{i(i-u)}$ comprise of g_{32}=g_{21}=1 regardless the input data length and g_{31}=$2^{\frac{d}{2}-1}$ where d is the input data length. Due to the nature of g_{31}, the DMR-D3R detector is realized with shifters instead of multipliers. This is because g_{31} has the form of $100...0_2$ (e.g., for 16-bit data is 2^7=128=10000000_2), which can use shift operation to accomplish a binary multiplication.

Figure 3 shows the functional units inside the proposed DMR-D3R detector. It consists of shifters instead of multipliers as in C-D3R. Also, x_3 is connected directly to the adder, while x_2 and x_1 are connected to their corresponding shifters.

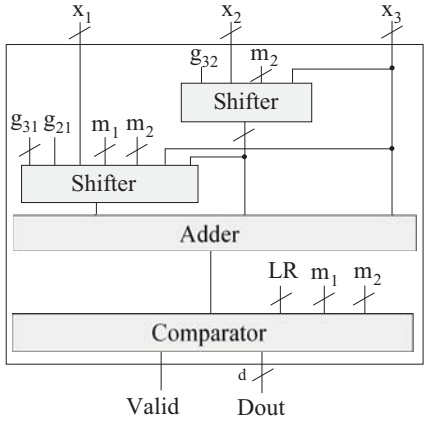

Fig. 3. Modified MRC-based DMR-D3R detector.

III. IMPLEMENTATION AND EXPERIMENTAL RESULTS

This section gives the implementation and experimental results of the proposed decoder. The focus will be on three aspects, namely area overhead, time latency and reliability.

A. Setup

In this work the two decoders mentioned in Section II-A and II-B were implemented. Note that C-D3R is based on the conventional MRC algorithm and implemented using hardware redundancy [14]. Contrarily, DMR-D3R decoder is based on the modified MRC algorithm and implemented using time redundancy (see Fig. 2 and 3). The implementation of the encoder and decoder were done using VHDL on Xilinx ISE and Synopsys Design Compiler tools.

Evaluation of the error correction capability was carried out using Matlab simulation. A series of adjacent bits, each with random length, that represent non-permanent clustered faults were injected to the memories at various fault rates from 1% up to 10%. At the same time, the faults were also injected to the decoder with the ratio of 1:10 as compared to the faults injected in the memory array. This ratio is set based on the soft error rates between SRAM bit and logic for 90nm [15].

B. Results

Figure 4 illustrates the area overhead for the DMR-D3R decoder as compared to C-D3R decoder. It shows that DMR-D3R occupies substantial less area than C-D3R. E.g. for 64-bit memory, the area for DMR-D3R is 18.4% smaller than that for C-D3R. The difference increases linearly as the memory word enlarges. Hence, it is expected that this benefit will be more noticeable for larger memory word size, which is the case for hybrid memories.

Figure 5 depicts the time latency for both decoders. It shows that for smaller memory word size, C-D3R operates slightly faster than DMR-D3R. However, as the memory word size increases, this difference becomes insignificant. Moreover, for memory word size bigger than 64-bit, DMR-D3R might operate faster than C-D3R. Taking the fact that hybrid memories will be designed with longer memory word size, it can be concluded that DMR-D3R suitable in terms of performance.

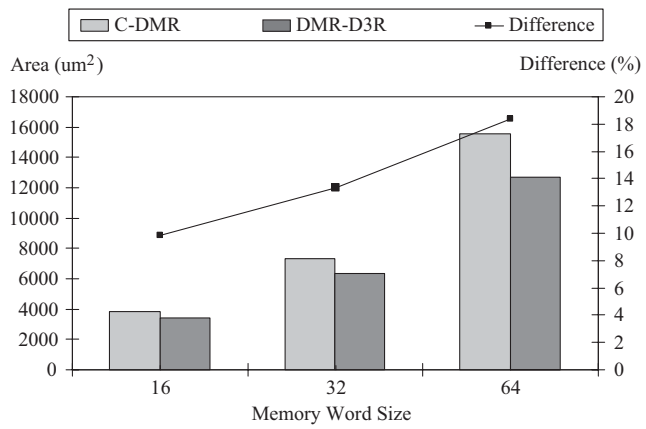

Fig. 4. Area overhead of the implemented decoders.

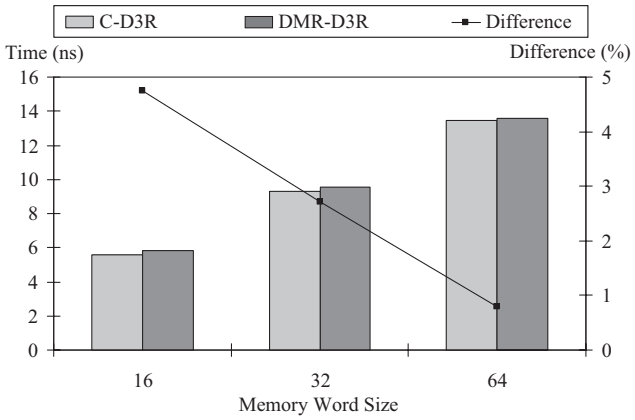

Fig. 5. Time latency of the implemented decoders.

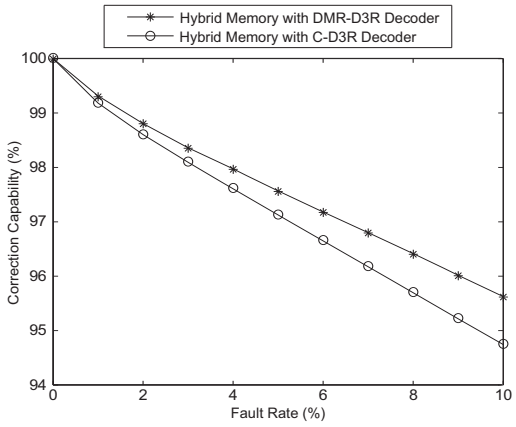

Fig. 6. Correction capability of hybrid memories with both decoders.

Figure 6 shows the correction capability for 64-bit memory decoded using the two decoders. It shows that DMR-D3R decoder is able to correct better than C-D3R decoder irrespective of the fault rate. The difference becomes larger at higher fault rate, e.g., 10× greater at 10% fault rate as compared to 1% fault rate. This result proves that the DMR-D3R decoder improves the reliability of the hybrid memory system especially at higher fault rates, which is the case for hybrid memories.

To the best knowledge of the authors, no published work has addressed this problem for hybrid memories except [13]. Besides of targeting random faults using *bit-oriented* ECC, the decoder proposed by [13] is implemented using non-CMOS devices. Contrarily, the decoder proposed in this work operates *symbol-oriented* ECC and is implemented using CMOS devices. Nevertheless, the results from [13] will be used for comparison.

The proposed DMR-D3R decoder can correct up to 50% of clustered faults for 32-bit memory, while this is only 14% for the decoder proposed in [13]. Moreover, the correction capability of the former remains constant as the memory word size increase, whereas the capability decreases in case of [13]. However, it is worth noting that the required memory array for DMR-D3R is 1.5× larger than that of [13].

IV. Conclusion

This paper has presented a cost-efficient fault-tolerant decoder for hybrid nanoelectronic memories. The main objective of this work is to improve the reliability of the entire hybrid memory system by protecting the memory array as well as the decoder. In order to achieve this objective, the advantages of concurrent correction capability of ECC, simplistic feature of hardware redundancy and rendezvous property of C-gate are combined. Furthermore, by modifying the decoding sequence of the RRNS-variant code and implementing it based on time redundancy, the cost-efficient implementation of the fault-tolerant decoder is realized. The proposed DMR-D3R decoder offers smaller area overhead, faster operation and better overall reliability when compared to the conventional C-D3R decoder for large memory word size.

References

[1] D. B. Strukov and K. K. Likharev, "Prospects for Terabit-Scale Nanoelectronic Memories", *J. Nanoscience and Nanotechnology*, vol. 16, no. 1, pp. 137–148, 2005.

[2] D. B. Strukov and K. K. Likharev, "Defect-Tolerant Architectures for Nanoelectronics Crossbar Memories", *J. Nanoscience and Nanotechnology*, vol. 7, no. 1, pp. 151–167, 2007.

[3] A. DeHon et al, "Nonphotolithographic Nanoscale Memory Density Prospects", *IEEE Trans. on Nanotechnology*, vol. 4, no. 2, pp. 215–228, 2005.

[4] M. A. Reed et al, "Molecular Random Access Memory Cell", *Applied Physics Lett.*, vol. 78, no. 23, pp. 3735–3737, 2001.

[5] L. B. Kish and P. M. Ajayan, "TerraByte Flash Memory with Carbon Nanotubes", *Applied Physics Letters*, vol. 86, no. 9, pp. 1–2, 2005.

[6] R. J. Luyken and F. Hofmann, "Concept for Hybrid CMOS-Molecular Non-volatile Memories", *J. Nanoscience and Nanotechnology*, vol. 14, no. 2, pp. 273–276, 2003.

[7] R. Waiser and M. Aono, "Nanoionics-based resistive switching memories", *Nature Materials*, vol. 6, pp. 833–840, Nov. 2007.

[8] C. Kügeler et al, "High Density 3D Memory Architecture Based on the Resistive Switching Effect", *J. Nanoscience and Nanotechnology*, vol. 14, no. 2, pp. 273–276, 2003.

[9] 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.

[10] S. Biswas et al, "A Pageable, Defect-Tolerant Nanoscale Memory System", in *Proc. of IEEE Int'l Symp. on Nanoscale Architecture*, pp. 85–92, 2007.

[11] 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.

[12] N.Z. Haron et al, "Redundant Residue Number System Code for Fault-Tolerant Hybrid Memories", accepted for *ACM Journal of Emerging Technologies in Computer Systems (JETC)*, Oct 2010.

[13] 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.

[14] N. Z. Haron and S. Hamdioui, "A High Performance Cluster-Fault Tolerance Scheme for Hybrid Nanoelectronic Memories", in *Proc. of IEEE Int'l Symp. on Defect and Fault Tolerance of VLSI Systems*, pp. 144–151, 2010.

[15] R. C. Baumann, "Radiation-Induced Soft Errors in Advanced Semiconductor Technologies" *IEEE Trans. on Device and Materials Reliability* vol. 5, no. 3, pp. 305–316, 2005.

[16] D. G. Mavis and P. H. Eaton, "Soft Error Rate Mitigation Techniques for Modern Microcircuits", in *Proc. of 40th Int'l Reliability Physics Symp.*, pp. 216–225, 2002.

[17] C. Constantinescu, "Trends and Challenges in VLSI Circuit Reliability", *IEEE Micro*, vol. 23, no. 4, pp. 14–19, 2003.

[18] D. E. Muller and W. S. Bartky, "A Theory of Asynchronous Circuits", in *Int'l Symp. Theory of Switching*, pp. 204-243, 1959.

[19] N. Szabo and R. Tanaka, *Residue Arithmetic and its Application to Computer Technology*. New York: McGraw-Hill, 1967.