Fault Tolerance Architecture for Reliable Hybrid CMOS/Nanodevice Memories

Nor Zaidi Haron Said Hamdioui
Delft University of Technology, Computer Engineering Laboratory
Mekelweg 4, 2628 CD Delft, The Netherlands
{N.Z.B.Haron, S.Hamdioui}@tudelft.nl

Abstract

Hybrid memories, structured from CMOS and non-CMOS devices, are potential candidates to replace existing memories due to their ultrascale integration. However, they are prone to high degree of non-permanent faults. In this paper a fault tolerance architecture for hybrid memories with higher reliability requirements is proposed. Non-permanent faults that can occur both in the data CMOS-encoder/decoder as well as in non-CMOS memory cell array are detected and corrected; hence significantly improving the reliability. The architecture is mainly based on a combination of two different fault tolerance schemes: (a) Redundant Residue Number System (RRNS) error correcting codes to allow multiple-bit error correction, and (b) scrubbing to enhance the error correction capacity.

1 Introduction

Hybrid CMOS/nanodevice memories, in short hybrid memories, are envisioned as one of the potential candidates to replace existing memories in the market. They are structured by complementing non-CMOS nanodevices (e.g., nanowires, single electron junctions, molecules) to nanoscale CMOS transistors. The former devices create arrays of storage, whereas the latter devices serve as the supporting periphery like encoder/decoder, input/output interface, global interconnects, etc. Several hybrid memory circuits have been proposed such as CMOL memory [1], nanowire-based memory [2], and molecular memory [3]. These novel circuits offer the potential for ultrascale integration in an order of 10^{12} device per centimeter square [1].

At the same time, however, the fabrication techniques and the characteristics of the nanoscale devices pose salient reliability challenges in such ultrascale circuits [4]. Nanodevices are prone to defects that induced by the inherent variability, for instance, of self-assembled fabrication. It is projected that these circuits are subject to extremely high defect rates as high as 10^{-1} compared to 10^{-9} in CMOS [4]. Additionally, the peripheral circuits fabricated from nanoscaled CMOS are more susceptible to variability (e.g., process, voltage, etc.) and external influences (e.g., noise, temperature fluctuation, and cosmic radiation) resulting in operational nonpermanent (i.e., intermittent and transient) faults [6].

The aim of this work is to develop a fault tolerance architecture for hybrid memories with higher reliability requirements; e.g., used in critical application (healthcare, security, aerospace, etc). The objective is to mitigate non-permanent faults both in the memory cell array as well as in data encoder/decoder. To realize the objective, appropriate fault tolerance schemes are chosen and combined while considering the impact on speed and area overhead.

The remainder of this paper describes briefly the concept of proposed architecture, including the combined fault tolerance schemes and the purpose of each one of them within the architecture.

2 The Proposed Fault Tolerance Architecture

As mentioned earlier, the aim of this work is to develop a fault tolerance architecture targeting faults both in the memory cell array and data encoder/decoder. Figure 1 shows the block diagram of the proposed architecture. Data coming into the memory is encoded into RRNS codeword. This codeword is then stored in the memory cell array. In fault-free case, during a read operation the codeword is fit into the decoder, which is then decoded and delivered as read data. Note that in this case, the decoder does not perform any correcting action.

It is envisioned that in hybrid memories, not only memory cell array but also data encoder/decoder can be impacted by non-permanent faults. For example during a writing operation, the encoder may encode the data into a wrong codeword, which will be then stored in the memory cell array. In this faulty case, the errors will be detected and corrected by the RRNS decoder during the read operation, provided that the errors are still within the RRNS code correction capability. Two critical parameters of the encoder/decoder, namely latency and area overhead, will be investigated in detail in order to develop an optimal solution.

As already mentioned, the errors will be detected and corrected by the RRNS decoder during the read operation, provided that the errors are still within the RRNS correction capability. In case the number of errors exceed the capacity, the errors will be not corrected. To deal with such a situation, *scrubbing* is used to further enhance the 'correction capacity' of the architecture. The scrubbing targets faults within memory cell array. It is activated to check and correct the stored codeword in the memory array (e.g., during idle time of the memories); hence preventing the number of errors to exceed the RRNS decoder capability before the codeword is decoded. An appropriate scrubbing frequency/scheduling will be developed to minimize the impact on the memory performance and throughput.

Other features that will be investigated for the proposed architecture consists of the use of *N-tuple modular redundancy (NMR)* (and the voting process) in RRNS encoder and decoder. Using of NMR allow the errors in the replicated encoder/decoder to be discarded during the voting process, on the condition that the number of fault-free encoder/decoder are more than the impacted part and fault-free voter. The appropriate number of N is the main subject to be investigated as the area is linearly increased as the number of N become larger. This scheme can be further explored by considering the *reconfigurability* of such redundant NMRs to choose appropriate number of duplications.

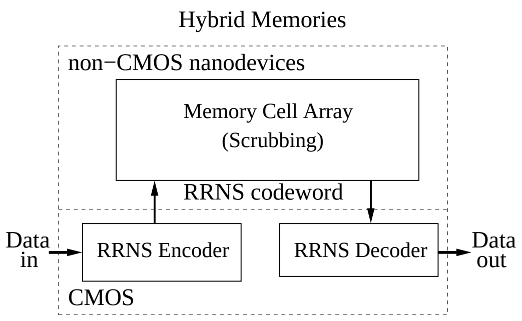

Figure 1. Proposed fault tolerance architecture.

In the rest of this paper a brief description for each of the schemes that will be investigated is given.

- **Redundant Residue Number System code**
  Error correction code based on Redundant Residue Number System (RRNS) is capable to perform multiple-bit error detecting and correcting in parallel, thus fast [5]. RRNS code has the same error correction capability to Reed Solomon (RS) code, which is half of the number of parity word appended to the data word.
- **Scrubbing**
  This technique periodically reads and corrects memory contents (if any error) to reduce the accumulated errors induced by non-permanent faults [6]. Scrubbing can be performed during the memory idle time or upon activated by special control signal akin to interrupt. At the time of scrubbing operation, memory cell array is isolated from input and output connection.
- **N-tuple modular redundancy (NMR)**
  This technique is a hardware redundancy technique implemented at circuit level [6]. The circuit of a RRNS moduli, which form encoder and decoder will be replicated N-times and the corresponding outputs will then be voted to determine the desired output. The NMR technique have been used not only at the N-copied of identical circuits but at voter.
- **Reconfiguration**
  This technique is a restructuring of hardware commonly utilized in programmable logic devices. The technique is suitable to be employed since RRNS codes are modular [5]. Therefore, the number of replicated encoder/decoder or the number of circuit that structure the RRNS encoder/decoder can be restructured on-the-fly depending on targeted metrics (e.g., level of reliability).

3 Conclusion

The proposed architecture is promising in delivering reliable hybrid memories as it mitigates faults both in non-CMOS based memory array and CMOS based encoder/decoder circuits. The validation of the concept is ongoing activity.

References

[1] D. B. Strukov et al., “CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices”, *Nanotechnology*, vol. 16, pp. 888–900, 2005.

[2] A. DeHon, “Nonphotolithographic nanoscale memory density prospects”, *IEEE Trans. on Nanotechnology*, vol. 4, no. 2, pp: 215–228, March 2005.

[3] Zettacore^{TM}. *ZettaCore^{TM}memory.* http://www.zettacore.com/

[4] J. R. Heath et al., “A defect-tolerance computer architecture: opportunities for nanotechnology”, *Science*, vol. 280, pp. 1716–1721, 1998.

[5] L. L. Yang et al., “Redundant Residue Number System Based Error Correction Codes”, *Proc. IEEE VTS 54th Vehicular Technology Conf., 2001*, vol. 3, pp. 1472–1476, 2001.

[6] R. Mastipuram, “Soft errors' impact on system reliability”, *Electronic Design Strategy, New Magazine*, pp. 69–74, September 2004.