IJMTST | Volume: 2 | Issue: 06 | June 2016 | ISSN: 2455-3778

Design Novel CMOL Architecture in Terabit-scale Nano Electronic Memories Technology

T.Muthamizhselvan[1] | R.Anandan[2]

[1]PG Scholar, Department of Electronics and Communication Engineering, Gojan School of Business and technology, Chennai, TamilNadu, India.
[2]Assistant Professor, Department of Electronics and Communication Engineering, Gojan School of Business and technology, Chennai, TamilNadu, India.

ABSTRACT

We have calculated the minimum chip area overhead and hence the bit density reduction. In this approach, the number of faults in each line of a memory block (matrix) is counted, and the lines having the largest number of defects are replaced with the spare lines. This method achieved by memory array reconfiguration (bad bit exclusion), combined with error correction code techniques, in prospective terabit-scale hybrid semiconductor/nanodevice memories, as a function of the nanodevice fabrication yield and the micro-to-nano pitch ratio. The results show that by using the best (but hardly practicable) reconfiguration and block size optimization, hybrid memories with a pitch ratio of 10 may overcome purely semiconductor memories in useful bit density if the fraction of bad nanodevices is below ~15%, while in order to get an order-of-magnitude advantage in density, the number of bad devices has to be decreased to ~2%. For the simple 'Repair Most' technique of bad bit exclusion, complemented with the Hamming-code error correction, these numbers are close to 2% and 0.1%, respectively. When applied to purely semiconductor memories, the same technique allows us to reduce the chip area 'swelling' to just 40% at as many as 0.1% of bad devices. We have also estimated the power and speed of the hybrid memories and have found that, at a reasonable choice of nanodevice resistance, both the additional power and speed loss due to the nanodevice subsystem may be negligible.

KEYWORDS: *ECC, Memories, CMOS, FPGA, RTL, MCS, Semiconductor, Nanodevice.*

I. INTRODUCTION

The recent spectacular advances in molecular electronics, in particular the demonstration of single-molecule single electron transistors by several groups, offer the hope for a practical introduction of hybrid semiconductor/nanodevice circuits, first of all for terabit-scale memory applications. In such memories, nanodevices (e.g., single molecules) would be used as single-bit memory cells, while the semiconductor transistor subsystem would perform all the peripheral (input/output, coding/decoding, line driving, and sense amplification) functions that require relatively smaller number of devices (scaling as $N1/2$, where N is the memory size in bits).

The first experimental steps toward the implementation of the hybrid memories have already been made. The main architectural challenge faced by the hybrid memories is the anticipated substantial fraction of 'bad' nanodevices, limited by both the integration technique (e.g., the chemically-directed molecular self-assembly, and the vulnerability of Nanoscale devices to random charged defects. The main approach to addressing this problem in semiconductor memory technology is reconfiguration, the replacement of memory array lines (rows or columns) containing bad cells by spare lines.

The effectiveness of the replacement depends on how good its algorithm. The Exhaustive Search approach (trying all possible combinations) finds the best repair solution, though it is not practicable because of the exponentially large execution time. A more acceptable choice is the 'Repair Most'

method that allows a simple hardware implementation and an execution time scaling linearly with the number of bits.

In semiconductor memories the ECC is reserved mainly for the suppression of soft rather than the hard errors, for insuring the memory fault tolerance rather than defect tolerance. However, for the prospective hybrid memories the defect tolerance is expected to be a much more acute problem, and ECC may need to be involved at the initial repair process as well.

The technical analysis of the opportunity, carried out in the pioneering work has been limited to the exclusion of just a few spare lines. Figure 1 shows the assumed general structure of the hybrid memory. It is essentially a matrix of L memory blocks, while each block is in turn a rectangular array of $(n + a) \times (m + b)$ memory cells. Here a and b present redundant resources that are being used for bad bit replacement at the initial test and repair stage, so that the final number of used memory cells is $L \times n \times m$. The good cells, addressed at each particular time step, form a row with one cell per block.

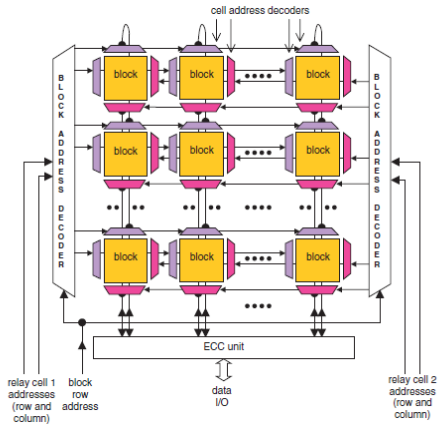


Figure: 1.1 Top structure of the analyzed hybrid memory

At each instance, block address decoders allow one to send the cell row and column addresses to a single row of blocks. The cell addresses are then processed by decoders of each block.

They have the same external word and bit addresses in each block, though due to the internal re-routing during the initial reconfiguration process, the real physical location of the used cell may be different in each block. The objective of our work is a study of the bad component exclusion techniques extended to an arbitrary number of redundant rows and columns of the matrix blocks, applied both with and without the error correction.

We have calculated the chip real estate overhead necessary for the fabrication of spare rows, columns, and auxiliary circuits, as a function of the bad bit fraction q and the ratio R of critical dimensions of the semiconductor transistors and nanodevice components. The results for $R = 1$ are also presented, since we believe that they are important for the evaluation of scaling prospects of purely semiconductor memories.

In contrast to the memory top structure (figure 1.1), the block architecture is substantially different from the traditional 'uniform' (semiconductor) memories. At each elementary operation, the block decoders address two vertical and two horizontal lines implemented in the CMOS layers of the integrated circuit, thus selecting a pair of 'relay' CMOS cells. The difference between the CMOL approach and the earlier suggestions (which seem hardly feasible from the fabrication point of view) is in the interfacing between the nanowires and the underlying CMOS-level wires in CMOL it is provided by sharp-pointed pins that are distributed all over the circuit area.

II. Existing System

The objective of our work is a study of the bad component exclusion techniques extended to an arbitrary number of redundant rows and columns of the matrix blocks, applied both with and without the error correction. We have calculated the chip real estate overhead necessary for the fabrication of spare rows, columns, and auxiliary circuits, as a function of the bad bit fraction q and the ratio R of critical dimensions of the semiconductor transistors and nanodevice components. The results for $R = 1$ are also presented, since we believe that they are important for the evaluation of scaling prospects of purely semiconductor memories.

2.1 Cmos Memory Architecture

The block architecture (figure 4.1) is substantially different from the traditional 'uniform' (semiconductor) memories. At each elementary operation, the block decoders address two vertical and two horizontal lines implemented in the CMOS layers of the integrated circuit, thus selecting a pair of 'relay' CMOS cells.

2.2 Circuit Behavior

At every single contraption series ti, the early period of our two-stage pipelined filter performs the pursuing procedures for the input example X: compute the new rank of every single cell, insert X in a cell that encompasses the token, and bypass the token to the subsequent cell. This way that for all Pi, Ri, and Ti lists, their new benefits will be

computed and ambitious at this period so that they can be notified at the subsequent series ti+1. At the alike period, the subsequent pipeline period computes the median worth for the input example that enters the window at the preceding series ti−1. That is, for the output list Y, its new worth will be computed at this period so that it can additionally be notified at the subsequent series ti+1.

For every single cell ci, the benefits of its Pi, Ri, and Ti lists are all shown in the figure. Initially, at series t0, to make the early input example be stored in the early cell c1, the last cell c5 is projected to encompass the token (T5 = 1).The rank and example benefits (Pi and Ri) of every single cell, alongside the benefits of the two input/output lists X and Y , are all reset to be zero.

When the early example 12enters the window at series t1, the token has been advanced fromc5 to c1 (T1 = 1 and T5 = 0). The worth of P5 has additionally been notified to be 5 as the early zero of X (now stored in R5) is indulged as an adjacent example at the early series t0. To design for the subsequent series t2, the new worth of R1 will be ambitious as 12to store the input example as c1 encompasses the token. The new worth of P1 will be computed as 5 as example 12 (to be stored in R1) is larger than the example benefits of the supplementary four cells. Finally, the new benefits of T1 and T2 will be ambitious as 0 and1, suitably, to indicate that the token will be advanced from c1to c2. All the benefits of Pi, Ri, and Ti will next be notified at the subsequent series t2. After the window is fully inhabited alongside valid data at series t6, cell c1 holds the token once more (T1 = 1).The new worth of the median output Y for the subsequent series t7will be ambitious as the worth of R4 (47) as rank P4 is equal to 3, i.e., (5 + 1)/2. As the counseled design is a two-stage pipeline, after Y is notified to be 47 at series t7 for the input example 66, the benefits of all Pi, Ri, and Ti will additionally be notified at this series for the subsequent input example 52. It can be perceived from this example that after an input example is inserted into the window, the aged example in every single cell will not be moved. Instead, the rank of every single cell is recalculated so that the new median can be obtained in a cell whose rank is equal to (N + 1)/2.

2.3 Rank Updating

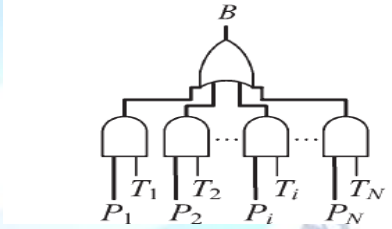
This section explains how to determine the new rank for each cell. Two types of cells will be separately discussed: a cell with the token and a cell without the token

Clk	Input Reg	Cell Registers															Output Reg
	X	T_1	T_2	T_3	T_4	T_5	R_1	R_2	R_3	R_4	R_5	P_1	P_2	P_3	P_4	P_5	Y
t_0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
t_1	12	1	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
t_2	59	0	1	0	0	0	12	0	0	0	0	5	0	0	0	4	0
t_3	35	0	0	1	0	0	12	59	0	0	0	4	5	0	0	3	0
t_4	47	0	0	0	1	0	12	59	35	0	0	3	5	4	0	2	0
t_5	66	0	0	0	0	1	12	59	35	47	0	2	5	3	4	1	12
t_6	52	1	0	0	0	0	12	59	35	47	66	1	4	2	3	5	35
t_7	38	0	1	0	0	0	52	59	35	47	66	3	4	1	2	5	47
t_8	18	0	0	1	0	0	52	38	35	47	66	4	2	1	3	5	52
t_9	26	0	0	0	1	0	52	38	18	47	66	4	2	1	3	5	47

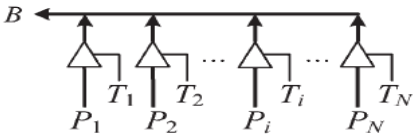
Table:1. Example illustrating the insertion of nine input samples into a window.

2.4 Ranksel and Mediansel Modules

The RankSel module is accountable for transferring the rank Pi of a cell ci to its output B if ci encompasses the token; i.e., after Ti = 1. Fig. 2(a) displays an easy implementation of this module employing AND/OR gates. It can additionally be requested by tri-state buffers in Fig. 2(b), whereas B is the output of a globe data bus that accumulates the output signals of all the tristate buffers. As there exists precisely one cell that encompasses the token at each time; i.e., there exists precisely one Ti signal whose worth is equal to1, the worth of B will always be valid.



(a)



(b)

Fig.3. Implementation of the RankSel module. (a) Using AND/OR gates. (b) Using tristate buffers.

The MedianSel module can additionally be requested in a comparable way. It transfers the worth of Ri to the output list Y if Riis the median; i.e., after Yi = 1. Though, if it is requested by tristate buffers, the median will be valid merely after there exists at least (N + 1)/2samples in the window; or else, it will stay in an elevated impedance state.

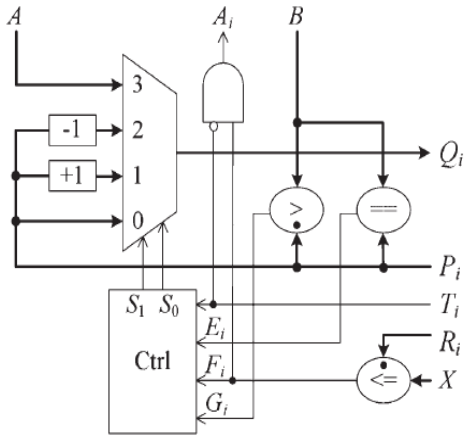
2.5 Rankgen and Rankcal Modules

For the RankGen module in a cell ci, its implementation is given in Fig. 4(a). signal Fi is the output of a comparator module"<=," that assesses the worth of Ri alongside that of the input example X so that Fi = 1if Riis less than or equal to X (Ri<=X), else Fi = 0 (Ri> X). Signal Ai is the output of a logic AND gate so that Ai = 1 if Ti = 0 and Fi = 1; i.e., if cell ci does not encompass the token and Riis less than or equal to X, else Ai = 0. The Ai signal of every single cell is related to the RankCalmodule, that computes the new rank of a cell that encompasses the token. New rank is computed as K + 1, whereas K is the number of cells, that does not encompass the token and whose example worth is less than or equal to X; i.e., K is equal to the number of logic 1's on the Ai signals of all the cells.

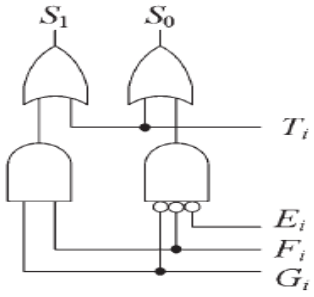
The RankCal module can be requested by a multi-input adder that adds all the Ai signals and next increments the sum by 1. If cell ci encompasses the token, the output A of the RankCal module will be its new rank at the subsequent series .On the contrary, if cell ci does not encompass the token, its new rankwill be ambitious by the supplementary signals. Signal Gi is the output of a comparator module ">," that assesses the worth of rank Pi alongside that of signal B so that Gi= 1 if Pi is larger than B, else Gi= 0. As the worth of B is the rank Pj of one more cell cj that encompasses the token, the meaning of Gi can additionally be delineated as Gi= 1 if Pi is larger than Pj(Pi >Pj),else Gi= 0(Pi <= Pj). signal Ei is the output of a compactor module "==," that additionally assesses the worth of Pi alongside that of B so that Ei= 1 if Pi is equal to B, else Ei= 0. Likewise, the meaning of Ei can be delineated as Ei= 1 if Pi is equal to Pj(Pi = Pj), else Ei= 0. Joining these two signals Ei and Gi, for the three relations amid Pi and Pj(Pi >Pj,Pi= Pj, and Pi <Pj), the corresponding worth of EiGi will be 01, 10, and 00, respectively.

2.6 CTRL Module

For a cell ci, since there are four possible sources to update its rank, a 4-to-1 multiplexer is used to select one of these sources for signal Qi in Fig. 3(a). Then, the worth of rank Pi will be notified by the worth of Qi at every single cycle. The multiplexer is manipulated by two selection signals S1 and S0; and these two signals, that are generated by the Ctrl module, are ambitious by four signals Ti, Ei, Fi, and Gi. If cell ci encompasses the token (Ti = 1), its new rank is obtained from the output A of the RankCal module; i.e., the worth of S1S0 ought to be 11when Ti = 1. If cell ci does not encompass the token (Ti = 0), there are five cases for this. In Case 1 after Pi >Pj(EiGi= 01) and Ri<= X (Fi = 1),therankof ci will be decremented by 1; i.e., the worth of S1S0should be 10 after EiFiGi= 011. Comparably in Case 2, after Pi <Pj(EiGi= 00) and Ri> X (Fi = 0),therankof ci will be incremented by 1; i.e., the worth of S1S0 ought to be 01when EiFiGi= 000. Finally, after Pi <Pj(EiGi= 00) and Ri<= X (Fi = 1) in Case 3, Pi >Pj(EiGi= 01) and Ri>X (Fi = 0) in Case 4, and Pi = Pj(EiGi= 10) in Case 5, the rank of ci will be retained unchanged; i.e., after EiFiGi= 010,001, or 1 − 0, the worth of S1S0 ought to be 00. Fig.4(b) depictsa easy implementation of the Ctrl module.



(a)



(b)

Fig.4. (a) Implementation of the Rank Gen module. (b) Implementation of the Ctrl module.

III. PROPOSED SYSTEM

3.1 Adaptive Median Filter

The Adaptive Median Filter is projected to remove the setbacks confronted alongside average median filter. The frank difference amid the two filters is that, in the Adaptive Median Filter, the size of window encircling every single pixel is variable. This variation depends on the median of the pixels in the present window. If the medians of pixel worth are an impulse, next the size of the window is expanded. Otherwise, more processing is completed on portion of the image inside the present window specifications. 'Processing' the image basically entails the following: The center pixel window is assessed to confirm whether it is an impulse or not. If it is an impulse, next new worth of that pixel in

filtered image will be the median worth of the pixels in that window. If, though, the center pixel is not an impulse, next the worth of the center pixel is retained in the filtered image. Thus, unless the pixel being believed is an impulse, the gray-scale worth of the pixel in the filtered image is the alike as that of the input image. Thus, the Adaptive Median Filter solves the dual intention of removing the impulse noise from the image and cutting distortion in the image. Adaptive Median Filtering can grasp the filtering procedure of a picture corrupted alongside impulse noise of probability larger than 0.2. This filter additionally smoothens out supplementary kinds of noise, therefore, providing a far larger output image than the average median filter.

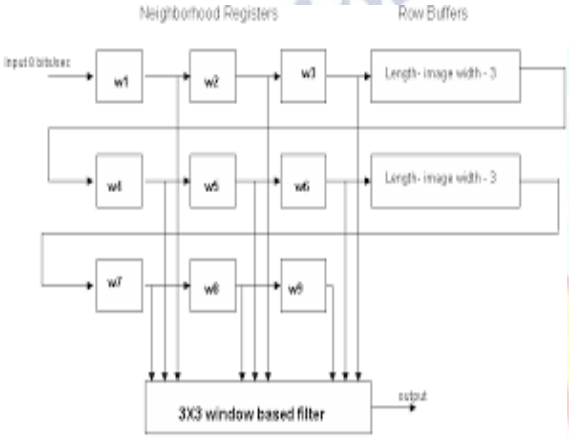


Fig.5. Implementation of a 3*3 filter window

3.2 Parallel Sorting strategy

To make fair analogy of parallel sorting strategy opposing wave sorter strategy in words of the finished number of needed steps sort an array, it is vital to ponder the steps utilized to elucidate data from recollection and the steps needed to store sorted data back to memory. The counseled way is established on alike construction of lists array utilized in the wave sorter strategy. With this kind of array, data can be stored in the array by dispatching a datum to the early list and afterward, after subsequent datum is dispatched to the early list, the worth on the early array is advanced to subsequent register. Thus, for every single datum dispatched to the array to be stored, benefits in lists advanced to their corresponding adjacent registers. This procedure needs to n steps. The alike number of steps is needed to seize data out from the array. This way permits store a new set of data in the array as the preceding set is being dispatched back into the memory. As remarked in serving 2, suffix sorting could imply extra than one sorting iterations. If k sorts are needed, next parallel sorting needs ((n+n/2) * k + n) to sort an array of n data.

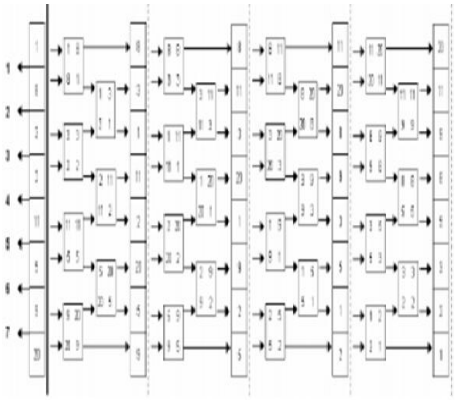


Fig.6. Parallel sorting.

IV. Results and Conclusion

To summarize, our results show that a simple combination of array reconfiguration (bad line exclusion) and Hamming-code error-correction techniques may make hybrid CMOS/nanodevice memories tolerant to a substantial fraction of bad nanodevices. However, the fabrication technology still has to reduce this fraction to below ~0.1% before terabit-scale memory chips, exceeding purely semiconductor memories in bit density by an order of magnitude, become possible. Possibly, this threshold may be moved up by almost an order of magnitude by the development of novel reconfiguration algorithms that would combine the simplicity and speed of the Repair Most exclusion with a higher repair quality (comparable to that of the Exhaustive Search). Under this approach, candidate solutions are modified by employing many small changes and occasionally large jumps. As a result, CS can substantially improve the relationship between exploration and exploitation, still enhancing its search capabilities. Despite such characteristics, the CS method still fails to provide multiple solutions in a single execution.

References

[1] Park H et al 2000 Nanomechanical oscillations in a single-C60 transistor Nature 407 57–60.
[2] Shorokhov V V, Johansson P and Soldatov E S 2002 Correlated electron tunnelling in the single-molecule nanosystem J. Appl. Phys. 91 3049–53.
[3] Zhitenev N B, Meng H and Bao Z 2002 Conductance of small molecular junctions Phys.Rev.Lett. 88 226801.
[4] Park J et al 2002 Coulomb blockade and the Kondo effect in single-atomtransistors Nature 417722–5.
[5] Kubatkin S et al 2003 Single-electron transistor of a single organic molecule with access to several redox states Nature 425 698–701.
[6] Kuekes P et al 2000 Molecular wire crossbar memory US Patent Specification #6 128 214.

[7] Likharev K 2003 Electronics below 10 nm Nano and Giga Challenges in Microelectronics ed J Greer et al (Amsterdam: Elsevier) pp 27–68.
[8] Stan M et al 2003 Molecular electronics: from devices and interconnect to circuits and architecture Proc. IEEE 91 1940–57.
[9] Chen Y et al 2003 Nanoscale molecular-switch crossbar circuits Nanotechnology 14 462–8.
[10] Zhong Z et al 2003 Nanowire crossbar arrays as address decoders for integrated nanosystems Science 302 1377–9.
[11] Li C et al 2004Multilevel memory based on molecular devices Appl. Phys. Lett. 84 1949–51.
[12] Fendler J H 2001 Chemical self-assembly for electronics applications Chem. Mater.13 3196–10.
[13] Tour J 2003 Molecular Electronics (Singapore: World Scientific).
[14] Prince B 1991 Semiconductor Memories 2nd edn (Chichester: Wiley).
[15] Chakraborty K and Mazumder P 2002 Fault-Tolerance and Reliability Techniques for High Density Random-Access Memories (Upper Saddle River, NJ: Prentice-Hall).