Repair Techniques for Hybrid Nano/CMOS Computational Architecture

Saket Srivastava, Aissa Melouki and Bashir M. Al-Hashimi
School of Electronics and Computer Science
University of Southampton, Southampton SO17 1BJ, UK
(ss3, am06r, bmah)@ecs.soton.ac.uk

***Abstract*—Presence of high defect rate in nanofabrics due to the inadequate fabrication processes has held back the development of emerging technology architecture. In this work, we propose two repair techniques to provide high level of defect tolerance in lookup table (LUT) based Boolean logic approach implemented in nano/CMOS. Further, we demonstrate that direct application of memory repair techniques is ineffective in dealing with high defect rate in hybrid nano/CMOS architecture. We show that the proposed techniques are capable of handling more than 20% defect rate in hybrid nano/CMOS architecture with efficient utilization of spare units.**

I. Introduction

Hybrid nano/CMOS architecture has shown promise in bridging the gap between CMOS and emerging technologies [1]. Tremendous gain in device density that can be achieved using nanoscale systems presents a compelling case for developing hybrid nano/CMOS computing architecture [2], [3], [4] where unreliable but highly dense nano/molecular systems are used to provide data storage and computation while CMOS components are utilized for interfacing and for highly critical circuit operations. It is acknowledged that due to high defect rate associated with nanotechnology, it is unlikely to compete with CMOS for general purpose computing in the near future and hence defect tolerance is necessary. Recently, error correcting techniques (ECC) have been proposed to improve the yield of hybrid nano/CMOS architecture in [5]. However, the reported ECC techniques are mainly used for the suppression of soft errors rather than physical defects i.e. maintaining the fault tolerance level rather than enhancing defect tolerance. Reconfiguration [6] is another technique that can circumvent physical defects by first mapping defects on reconfigurable fabrics then synthesizing a feasible configuration to realize an application for each nanofabric instance. While exact manufacturing defect rate is not yet pinpointed, it is believed to exceed 10% [7]. Repair techniques are very effective in memory design and this represents the starting point for this work. Traditional CMOS memory repair algorithms, when applied to hybrid nano/CMOS computational architecture, incur huge overhead in the decoder circuitry, which is implemented in CMOS. Moreover, as we will see later in this work, direct application of memory repair techniques is ineffective in dealing with high defect rate in nano/CMOS computational architecture. Most of the earlier work in nano/CMOS co-design have targeted memory [8] or crossbar architecture. To advance computational nanocircuits, new architectures must be pursued. One such promising architecture is the Look-Up Table (LUT) based Boolean logic approach considered in this paper.

LUT implementation targeted in this work, is an effective functional-coding approach that provides low-level protection of individual Boolean logic functions [9], [10]. We show that our proposed repair techniques are capable of targeting high defect rates using a tagging mechanism that results in low CMOS overhead. This technique is called *Tagged Repair* technique. We demonstrate the efficiency of the proposed repair techniques on hybrid nano/CMOS circuits synthesized into LUTs of different sizes. A further improvement in this technique, in terms of targeted defect rate, has been achieved by dividing the original LUT into smaller sub-LUTs (column-wise). This approach, called the *Modified Tagged Repair* technique, is capable of targeting higher defect rates (upto 20%), at a cost of extra CMOS tagging bits when compared to Tagged Repair technique. The novelty of this work lies in the development of highly efficient repair techniques in the context of emerging technology (nano/CMOS) architecture implemented as LUTs. An added advantage of the proposed techniques is reduced CMOS area overhead, as compared to traditional memory repair algorithms that require physical to logical address mapping [8]. To the best of our knowledge, there are no reported repair techniques that target such high defect rates in LUT based hybrid nano/CMOS architecture.

II. Proposed Repair Techniques

In this section we propose two repair techniques that have been developed specifically for LUT based Boolean logic approach implemented in nano/CMOS. The original algorithm used in CMOS memory repair incurs a significant CMOS area overhead when applied to LUT based approach which will nullify the device density gained by using nano components. The modified algorithm involves replacing rows and columns instead of blocks of defective units. We have also included a tagging mechanism to isolate defective rows and columns. Each row/column is associated with a CMOS tag that holds one bit of information. A '1' or '0' tag value specifies whether or not a row/column is selected in the final LUT after repair. We refer to this technique as *Tagged Repair* technique. A further improvement in this technique, in terms of targeted defect rate, has been achieved by dividing the original LUT into smaller sub-LUTs (column-wise). This approach, called the *Modified Tagged Repair* method, is capable of targeting

higher defect rates (upto 20%), at a cost of extra CMOS tagging bits when compared to Tagged Repair technique.

A. Tagged Repair Technique

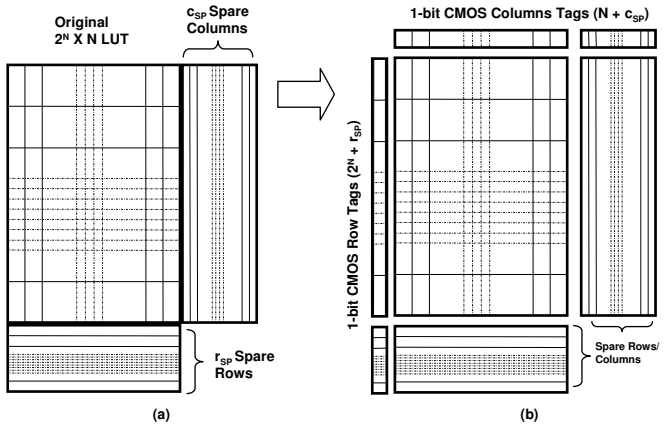

Fig. 1. Tagged Repair Technique: (a) Implementation for a $2^N \times N$ LUT with c_{sp} spare columns and r_{sp} spare rows (b) Use of tags to repair columns and rows

The aim of our proposed Tagged Repair technique is to identify a defect-free instance of a LUT of size (2^N (rows) $\times N$ (columns)) within a defective fabric given a certain amount of spare columns c_{sp} and spare rows r_{sp}. Hence, a theoretical estimation of the circuit failure rate of this technique reduces to the calculation of the probability of the non-existance of a subset of defect-free resources ($2^N \times N$) within the partially-usable fabric $\big((2^N + r_{sp}) \times (N + c_{sp})\big)$. We first calculate the probability $P_{(col,L)}$ of a column of size $(r + L)$ is defective (i.e. in which the total number of defective bits exceeds the number of spare rows L). The probability of successfully finding enough resources to create an instance of a given LUT using our Tagged Repair technique where the number of spare rows is $L = r_{sp}$:

$$P_{succ} = \sum_{x=c}^{c+c_{sp}} \binom{c+c_{sp}}{x} P_{inst}(x, r_{sp}) \tag{1}$$

where P_{inst} is the probability that n columns out of $(c + c_{sp})$ are not defective and aligned. $P_{inst} = f(P, c, c_{sp}, r, r_{sp})$ and P is the *defect rate*. Therefore, the overall *failure rate*, $P_{failure}$ is:

$$P_{failure} = 1 - P_{succ} \tag{2}$$

Fig. 1(b) shows the implementation of the Tagged Repair technique. This technique uses a tagging method to tag rows and columns that are least defective. Initially the tags for original 2^N rows and N columns in LUT are set to 1 and tags for spare rows and spare columns (r_{sp} and c_{sp} respectively) are set to 0. The implementation algorithm for this architecture is presented below:

1: Initialize LUT size, spare rows and spare columns
2: Initialize all LUT tags
   {S}can Column-wise
3: **for all** $i(< N)$ **do**
4: **for all** $j(< c_{sp})$ **do**
5: if totalDefects($c_{sp}(j)$) < totalDefects($column(i)$) ($Tag(c_{sp}(j)) = 1$, $Tag(column(i)) = 0$)
6: **end for**
7: **end for**
   {R}epeat scan Row-wise

After the repair process, the tags will hold '1' for the least defective rows and columns and '0' for the excluded ones. The proposed architecture is comparatively simpler as it does not require encoding/decoding circuitry that will also lead to additional area overhead in CMOS domain (as compared to other techniques such as [6], [11]).

B. Modified Tagged Repair Technique

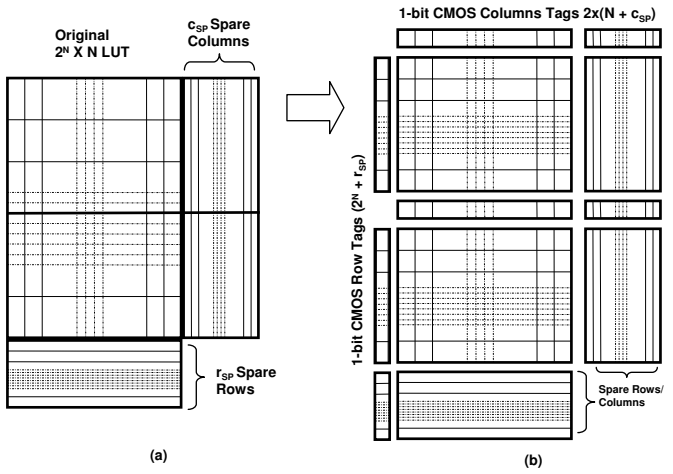

Fig. 2. Modified Tagged Repair Technique (a) Implementation for a $2^N \times N$ LUT with c_{sp} spare columns and r_{sp} spare rows (b) Use of tags to repair columns and rows

To address even higher defect rates, we investigate a modified technique as presented in Fig. 2 which is a modified implementation of the previous Tagged Repair technique shown in Fig. 1. In this technique instead of replacing entire columns, we have split the columns in two equal sections before applying tagging and replacement, to make more optimized usage of the spare units.

In the Modified Tagged Repair technique, a successful instantiation of a LUT on the fabric is achieved by successfully instantiating each half of the LUT ($2^{N-1} \times N$) on the fabric given the amount of spare columns c_{sp} for each half and the spare rows r_{sp} that is reserved for both of them. Eq.(3) represents the total probability P_{succ} that can be used to calculate $P_{failure}$ from Eq. 2. Variable i is the number of spare rows used by our technique to repair the defective rows in the first half, whereas the rest of spare rows $(r_{sp} - i)$ are used in the repair of the second half of the LUT. Hence P_{succ} can be computed as follows:

$$P_{succ} = \sum_{i=0}^{r_{sp}} \Bigg[\Big(\sum_{x=c}^{c+c_{sp}} \binom{c+c_{sp}}{x} P_{inst}(x, i) \Big) \times \Big(\sum_{x'=c}^{c+c_{sp}} \binom{c+c_{sp}}{x'} P_{inst}(x', r_{sp} - i) \Big) \Bigg] \tag{3}$$

The implementation algorithm used for this technique is similar to Tagged Repair technique however slightly more complex due to the split column implementation. In the algorithm for Modified Tagged repair technique, the column-wise scan needs to be done in two stages and the row-wise scan will be done in a single stage. The reason for this is as follows: since a $2^N \times N$ LUT will always have even number of rows (2^N), it is easy to use this approach in splitting the columns halfway each of size 2^{N-1}. A similar approach to split and tag columns cannot be used since the column length can be odd or even depending on value of N and an odd value of N cannot be split in two equal integers. The downside of this approach is that it makes the technique more complex since the number of column tags required will be double that of the Tagged Repair technique (section III).

III. Experimental Results

In this section we first evaluate the performance of the two proposed repair techniques (Tagged Repair and Modified Tagged Repair). Simulations were performed on randomly-generated symmetric LUTs where the probability of 0 and 1 are equal. The LUTs are of sizes ranging from $2^3 \times 3$ to $2^6 \times 6$. Larger circuits (such as ISCAS'85 benchmarks) with even higher number of inputs and outputs can be synthesized to smaller LUTs using synthesis tools such as Synplicity [12]. The circuit failure probability $P_{failure}$, resulting from randomly injecting m defects, is obtained by calculating the ratio of defective LUTs after repair to the total number of simulation iterations $I = 5000$. *Targeted defect rate* for a particular repair technique is the maximum defect rate for which 0% failure rate can be achieved. *Redundancy (or Spares)* is the percentage of extra rows/columns that are allocated for repair. All the simulations were carried out in C/C++. To demonstrate the limitation of current memory repair techniques in the context of LUT based hybrid nano/CMOS architecture, we simulated Repair Most technique [8], that was proposed recently in the context of hybrid nano/CMOS memory design. We compared the results of Repair Most technique with our proposed techniques to show the gain in repair efficiency. For implementation details of Repair Most technique, please refer to [8]. The simulations for Repair Most technique have been performed by fixing the values of column threshold (c_{th}) at 3 and row threshold (r_{th}) at 1.

A. Simulation of the Proposed Techniques

Fig. 3 shows the plot of Failure rate Vs Defect rate using Tagged Repair technique for different LUT sizes. As can be seen, the Tagged Repair technique exhibits significantly higher defect tolerance (upto 17%) in the case of smaller sized LUTs (such as $2^3 \times 3$ LUT) than larger LUTs (such as $2^6 \times 6$ LUT). Hence synthesis of larger circuits into smaller LUTs will result in improved defect tolerance for the targeted nano/CMOS architecture. The results shown assume 100% redundancy (i.e. $c_{sp} = N$ and $r_{sp} = 2^N$) with 0% DCCs in the LUTs.

As can be seen from the results shown in Fig. 4 the modified technique further improves the defect tolerance of the given LUT architecture when compared to the Tagged Repair technique. Taking an example of a $2^3 \times 3$ LUT, we compare the results of Fig. 3 with Fig. 4. It can be seen that while the Tagged Repair technique can achieve 0% failure rate at defect rates of upto 17%, the modified technique can target defect rate upto 20%. This improvement in efficiency is due to the more optimized usage (by splitting the columns in two before applying repair) of the redundant spare units. Similar improvement in repair efficiency can also be seen for other LUT sizes.

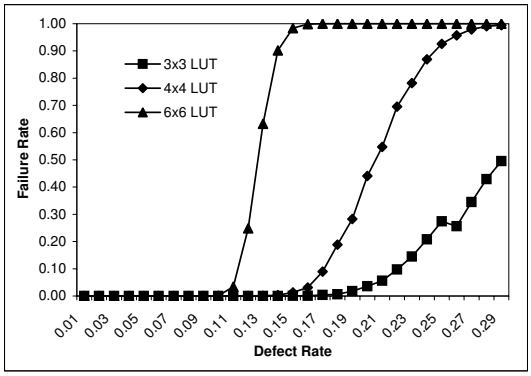

Fig. 3. Plot of Failure rate Vs Defect rate using Tagged Repair technique for different LUT sizes with 100% redundancy in rows and columns.

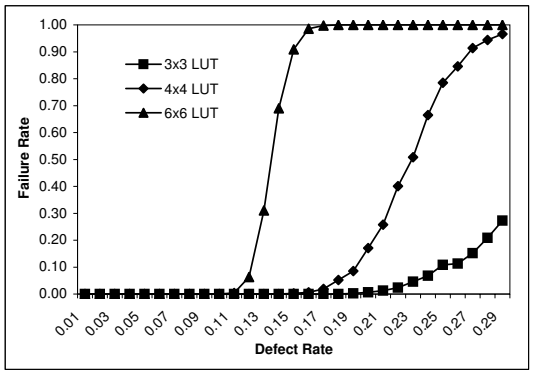

Fig. 4. Plot of Failure rate Vs Defect rate using Modified Tagged Repair technique for different LUT sizes with 100% redundancy in rows and columns.

Fig. 5 compares the efficiency of the proposed techniques with the Repair Most technique [8] for LUTs of size $2^4 \times 4$ LUT with 100% redundancy. As can be seen, the Modified Tagged Repair technique targets the highest defect rate followed by the Tagged Repair and Repair Most respectively. For example, when the defect rate is 15%, the Modified Tagged

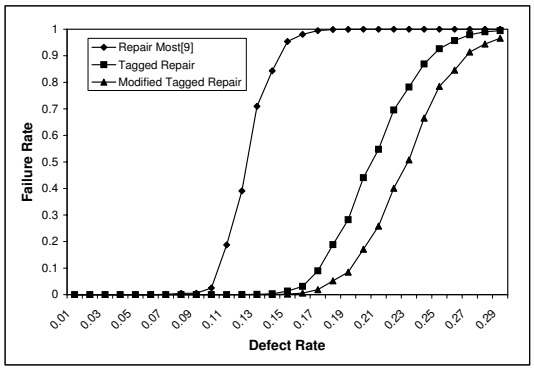

Fig. 5. Comparative study of the proposed repair techniques with the Repair Most technique for the failure rate of $2^4 \times 4$ LUT with 100% redundancy

Spares	LUT size	Repair Most [8]	Tagged Repair (proposed)	Mod. Tagged Repair (proposed)
	3x3	7.0%	10.0%	12.0%
25%	4x4	4.0%	5.0%	6.0%
	6x6	2.0%	3.0%	4.0%
	3x3	8.0%	11.0%	14.0%
50%	4x4	6.0%	9.0%	9.0%
	6x6	2.0%	6.0%	6.0%
	3x3	9.0%	17.0%	20.0%
100%	4x4	8.0%	14.0%	15.0%
	6x6	2.0%	10.0%	11.0%

TABLE I
COMPARATIVE REPAIR COST OF THE PROPOSED TECHNIQUES AND THE REPAIR MOST TECHNIQUE IN TERMS OF TARGETED DEFECT RATE

repair technique gives a failure rate of 0%, and the original Tagged Repair technique gives a failure rate of 2%, whereas, the Repair Most technique gives a failure rate of 90%.

B. Cost of Repair

A key advantage of our proposed techniques is that they use considerably less redundancy (percentage of spare units) to tolerate even higher defect rates compared to Repair Most technique. The overall nanodevice area of the proposed techniques (Fig. 1 and 2) for a $2^N \times N$ LUT with c_{sp} spare columns and r_{sp} spare rows will be $(2^N + r_{sp}) \times (N + c_{sp}) - (r_{sp} \times c_{sp})$. However, due to the implementation architecture of Repair Most technique, the total area (including original LUT and the spare units) of the Repair Most implementation will be $(2^N + r_{sp}) \times (N + c_{sp})$ units/LUT. As as example, the total nanodevice area in case of a $2^3 \times 3$ LUT implementation with 100% spares for the proposed techniques will be $3 \times 2^3 \times 3 = 72$ units/LUT. Whereas for the Repair Most technique, the total nanodevice area will be $4 \times 2^3 \times 3 = 72$ units/LUT, which is 25% higher. Table I shows the comparative repair cost of the two proposed techniques with the Repair Most technique in terms of targeted defect rate. It can be seen that in case of a $2^3 \times 3$ LUT, Modified Tagged Repair can target upto 12% defect with only 25% spares, Repair Most is not able to target 10% defect rate even with 100% spares. The targeted defect rate values given in this table have been rounded off to the nearest 1%. Similarly the number of spare units has been rounded off to the nearest whole number based on percentage of spares.

The CMOS area overhead for the Tagged Repair technique is $(2^N + r_{sp})$ single bit row tags and $(N + c_{sp})$ single bit column tags. As an example, assuming 100% redundancy, the CMOS area overhead for a $2^3 \times 3$ LUT using Tagged Repair technique will be a total of $2 \times (2^3 + 3) = 22$ single bit tags. However, the Modified Tagged Repair technique will require an extra $(N + c_{sp})$ single bit column tags (making it a total of $2 \times (N + c_{sp})$ column tags). Hence the CMOS area overhead for a $2^3 \times 3$ LUT, with 100% redundancy, will be $2 \times (2^3 + 3 + 3) = 28$ tags for a single $2^3 \times 3$ LUT. Considering a single bit SRAM cell requires 6 transistors [13], this will result in $(28 - 22) \times 6 = 36$ extra CMOS transistors/LUT as compared to Tagged Repair technique. The overall CMOS area overhead in terms of transistor count for other LUTs can be calculated accordingly. The number of row tags for the Modified Tagged Repair technique will be equal to the Tagged Repair technique.

IV. CONCLUSION

In this work we proposed two new repair techniques for emerging technology (nano/CMOS) architecture implemented as LUTs. We have presented theoretical equations and implementation algorithm for our proposed repair techniques. Simulation results have shown that our proposed techniques are capable of handling upto 20% defect rates in LUT based hybrid nano/CMOS architecture. We have also shown that our repair efficiency of our proposed techniques is better if circuits are synthesized in smaller sized LUTs. To the best of our knowledge, the proposed methods are the only techniques capable of targeting such high defect rates in LUT implementation in hybrid nano/CMOS architecture. We have also presented a detailed study of repair cost in terms of percentage redundancy and an estimate of CMOS area overhead for the proposed techniques.

V. ACKNOWLEDGEMENT

The authors would like to acknowledge the EPSRC (UK) for funding this project in part under grant EP/E035965/1 as well as the Algerian Ministry of Higher Education and Scientific Research.

REFERENCES

[1] M. M. Ziegler and M. R. Stan, "A Case for CMOS/nano co-design," in *ICCAD '02: Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, (New York, NY, USA), pp. 348–352, ACM, 2002.

[2] C. Jeffery, A. Basagalar, and R. Figueiredo, "Dynamic sparing and error correction techniques for fault tolerance in nanoscale memory structures," *Nanotechnology, 2004. 4th IEEE Conference on*, pp. 168–170, Aug. 2004.

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

[4] A. DeHon, S. Goldstein, P. Kuekes, and P. Lincoln, "Nonphotolithographic nanoscale memory density prospects," *Nanotechnology, IEEE Transactions on*, vol. 4, pp. 215–228, March 2005.

[5] A. Singh, H. Zeineddine, A. Aziz, S. Vishwanath, and M. Orshansky, "A heterogeneous CMOS-CNT architecture utilizing novel coding of boolean functions," *NANOARCH 07*, pp. 15–20, Oct. 2007.

[6] M. Mishra and S. Goldstein, "Defect tolerance at the end of the roadmap," *In ITC*, vol. 1, pp. 1201–1210, 30-Oct. 2, 2003.

[7] M. Stan et.al., "Molecular Electronics: From Devices and Interconnect to Circuits and Architecture," *Proceedings of the IEEE*, vol. 91, pp. 1940–, Nov. 2003.

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

[9] S. Paul, R. S. Chakraborty, and S. Bhunia, "Defect-aware configurable computing in nanoscale crossbar for improved yield," *IEEE International On-Line Testing Symposium*, vol. 0, pp. 29–36, 2007.

[10] N. R. Shanbhag, S. Mitra, G. de Veciana, M. Orshansky, R. Marculescu, J. Roychowdhury, D. Jones, and J. M. Rabaey, "The search for alternative computational paradigms," *IEEE Design and Test of Computers*, vol. 25, no. 4, pp. 334–343, 2008.

[11] S. Goldstein and M. Budiu, "NanoFabrics: spatial computing using molecular electronics," *IEEE ISCA*, pp. 178–189, 2001.

[12] "http://www.synplicity.com/,"

[13] A. Bellaouar and M. Elmasry, "Low-Power Digital VLSI Design: Circuits and Systems," *Springer Publication*, . 1995.