Tagged Repair Techniques for Defect Tolerance in Hybrid nano/CMOS 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—**We propose two new repair techniques for hybrid nano/CMOS computing architecture with lookup table based Boolean logic. Our proposed techniques use tagging mechanism to provide high level of defect tolerance and we present theoretical equations to predict the repair capability including an estimate of the repair cost. The repair techniques are efficient in utilization of spare units and capable of targeting upto 20% defect rates, which is higher than recently reported repair techniques.**

I. Introduction

The gain in device density that can be achieved using nanoscale devices presents a compelling case for developing hybrid nano/CMOS computing architecture [1], [2], [3], [4]. In a hybrid nano/CMOS architecture, unreliable but highly dense nano devices are used to provide data storage and computation while CMOS devices are utilized for interfacing and for highly critical circuit operations. It is acknowledged that due to high defect rates associated with nanotechnology, it is unlikely to compete with CMOS for general purpose computing in the near future and hence defect tolerance is necessary [5], [6], [7], [8], [9]. While defect tolerance has been addressed to some level in CMOS generation, it has gained importance recently with the emergence of novel computational paradigms that involve highly dense nano/CMOS architectures.

To achieve acceptable levels of manufacturing yield for nano/CMOS architecture efficient repair techniques need to be implemented [10]. There is a large body of literature available for efficient CMOS memory repair techniques, recent examples include [11], [12]. It is our intention to develop repair techniques that use the available literature to provide defect tolerance in hybrid nano/CMOS architecture. This intention is motivated by the proven effectiveness of these repair techniques in practice. Further, incorporating these proven repair techniques will facilitate their greater acceptance by the electronics design industry. While exact manufacturing defect rate is not yet pinpointed, it is believed to exceed 10% [5]. Hence there is a need for more efficient repair techniques that target higher defect rates. Moreover, most of the earlier works in nano/CMOS design have targeted crossbar architectures [13], [14], [15]. 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. In this work, we target a hybrid nano/CMOS computational paradigm based on a LUT implementation of Boolean logic functions [16]. LUT implementation is an effective functional-coding approach that provides low-level protection of individual Boolean logic functions [17]. In [16], the authors have targeted a LUT based design paradigm that achieves 87% reliability at 10% defect rate for ISCAS'85 benchmark circuits. We show that our proposed repair techniques are capable of targeting higher defect rates (upto 14%) for ISCAS'85 benchmark circuits while achieving 100% reliability. To the best of our knowledge, there are no reported repair techniques that target such high defect rates in LUT based approach implemented using hybrid nano/CMOS architecture. It should be noted that reliability of nano/CMOS architecture using fault tolerance has been achieved through Error Correcting Codes [18], [19]. However, since our proposed techniques deal with defect tolerance, we have not covered fault tolerance techniques in this paper.

The paper is organized as follows: in section II, preliminaries and related work is outlined including a recent repair technique in section III. In section IV, we present the design and implementation of the proposed repair techniques. Simulation results of the proposed techniques are given in section V and the work is concluded in section VI.

Copyright (c) 2010 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org .

II. Preliminaries and Related Work

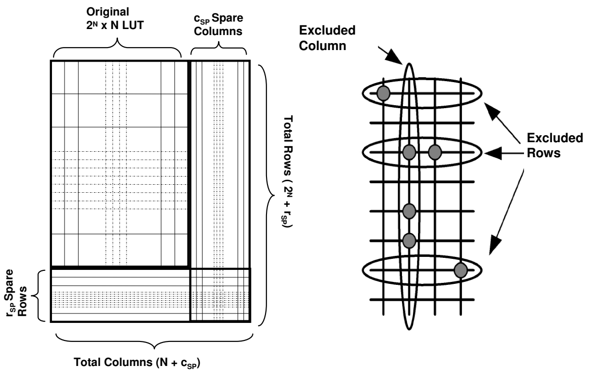
Three most prominent issues related to reliable nano/CMOS design are: defect tolerance technique, choice of architecture, and defect distribution model. Currently, most of the defect tolerance techniques in nano/CMOS have been derived from the memory repair techniques used in CMOS. The two most common techniques to address high defect rate in nanoscale system design have been reconfiguration and repair. Reconfiguration [10], [20] circumvents physical defects by first mapping defects on reconfigurable fabrics then synthesizing a feasible configuration to realize an application for each nanofabric instance. While reconfiguration is more popular in nano-crossbar designs for ASIC/FPGA architecture, repair is popular for nanoscale memory design [14]. Both techniques have shown great promise addressing defect tolerance so far. However, it is not meaningful to differentiate reconfiguration and repair techniques since both techniques utilize spare units to eliminate/minimize defective units and improve yield. Reconfiguration techniques use a graph theory approach to make use of least defective units while leaving out the defective ones. Repair techniques on the other hand use a hardware approach that includes a mapping circuit to keep track of defective units that are being replaced by the non-defective spare units. In terms of cost both techniques rely on redundant spare units and carry an additional overhead in terms of latency and power dissipation.

The repair capability of defect tolerance techniques can vary greatly for different architecture designs [8]. There have been a number of promising architecture proposed for the hybrid nano/CMOS design paradigm. One of the first architectures using hybrid nano/CMOS design paradigm was proposed in [21], using a double layered architecture. A FET based architecture using Carbon Nanotube and Silicon Nanowire interconnects was proposed by Dehon in [22], [23]. A field-programmable nanowire interconnect (FPNI) based architecture was recently proposed in [24]. In FPNI and CMOL [15] architectures, logic is performed using CMOS components, while nanowires are used for interconnects and routing. It appears from the literature that hybrid nano/CMOS design paradigm offers the most promise in achieving future computational needs [5]. In this work we have proposed repair techniques for a generalized hybrid nano/CMOS architecture that uses dense nanofabrics to perform LUT based computation and CMOS components to provide defect tolerance. For a general hybrid nano/CMOS architecture random defect distribution can be assumed, however, a more effective way to deal with large number of defects would be to identify and then target defect model inherent in a particular nanotechnology [7]. To demonstrate the significance of defect distribution we have also used the proposed repair techniques on clustered and row/column defect distribution in a nanofabric (in section V-C).

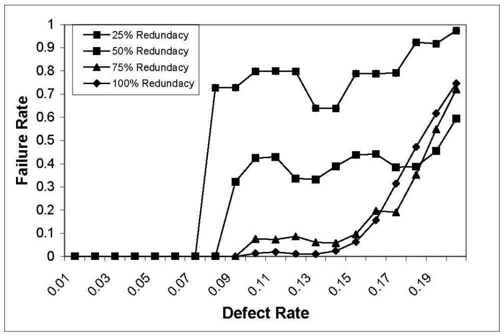
III. Repair Most Technique

To demonstrate the repair capability of current memory repair techniques for LUT based nano/CMOS architecture, we first implemented the Repair Most technique [14] that was recently proposed in the context of highly dense terabit scale nano/CMOS memory architecture. In Repair Most technique, an instance of a $2^N \times N$ LUT (here number of rows = 2^N and number of columns = N as shown in Fig. 1(a)) is created on a defective nanofabric by deleting rows and columns on which defective bits exist. An appropriate amount of redundancy need to be allocated in both dimensions to compensate for the removed/deleted rows and columns. Repair Most technique has two phases: In the first phase, if the number of defects per row (or column) exceeds the threshold r_{th} (or c_{th}), the row (column) will be excluded from the LUT and replaced with a spare row (column) as shown in Fig. 1(b). Here, r_{th} (or c_{th}) represents the minimum number of errors present in a row (or column) for it to be excluded. In the second phase, the remaining defective cells are repaired by replacing the defective columns (or rows) with the spare columns (rows).

We have simulated Repair Most technique for different $2^N \times N$ LUT implementations with 25%-100% redundancy in spare units. Fig. 1(b) shows the failure rate vs defect rate obtained for a $2^3 \times 3$ LUT at different values of redundancy



(a)



(b)

Fig. 1. Repair Most technique (a) Implementation for LUT based architecture and repair mechanism by setting value of $c_{th} = 2$ and $r_{th} = 0$. Plot of Failure rate Vs Defect rate using Repair Most technique for $2^3 \times 3$ LUT.

(% spares). The value of c_{th} and r_{th} were chosen as 2 and 0 respectively for illustration purposes. One of the limitations of Repair Most technique is a different value of c_{th} and r_{th} need to be chosen for a particular defect rate in order to achieve the most optimized repair capability. As can be seen, the Repair Most approach provides limited defect tolerance when defect rate $> 10\%$ even if we provide 100% redundancy. Hence this technique can target at most 10% defect rate for a $2^3 \times 3$ LUT implementation. For larger LUT sizes (upto $2^6 \times 6$) similar results were obtained, however, larger LUT implementations (such as $2^6 \times 6$) failed at a much smaller defect rate even with 100% redundancy. To estimate the CMOS area overhead of the Repair Most technique, the authors in [14] have proposed a special mapping table implemented in CMOS to map the physical addresses of the good nanodevices. The overall chip area for Repair Most technique for the implementation shown in Fig.2 is given by:

$$\begin{aligned} A_{total} &= A_{LUT\ decoders} + L \times [A_{LUT\ unit} + A_{LUT\ unit\ decoder} \\ &+ A_{mapping\ table} + A_{Wiring/Routing}] \end{aligned} \quad (1)$$

where L is the number of LUTs in the architecture. The area of this mapping table $A_{mapping\ table}$ (not shown in Fig.2) will depend on the size of LUT block being addressed. For more details on this technique, refer to [14].

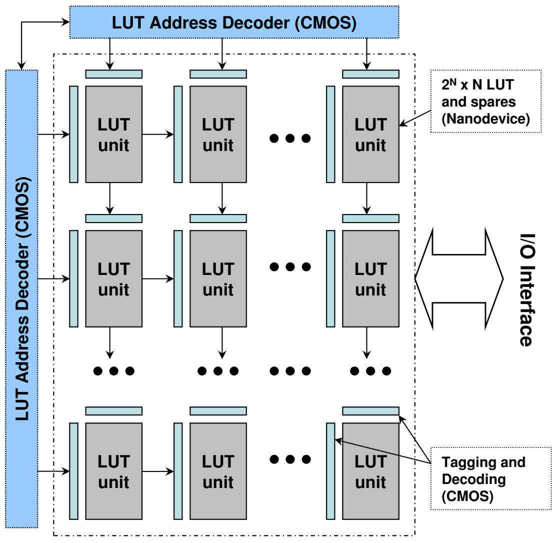

LUT Address Decoder (CMOS)
LUT Address Decoder (CMOS)
LUT unit
2N x N LUT and spares (Nanodevice)
I/O Interface
Tagging and Decoding (CMOS)

Fig. 2. Implementation of LUT based Boolean logic approach using hybrid nano/CMOS architecture

IV. Proposed Repair Techniques

We propose two new repair techniques that have been developed specifically for LUT based Boolean logic approach implemented in nano/CMOS. Fig.2 shows the implementation of the proposed LUT based hybrid nano/CMOS architecture. Defect tolerance techniques in hybrid nano/CMOS architectures can be implemented at different levels of abstraction. In [25], the authors have used an orthogonal approach as compared to our work (i.e. replacing defective LUTs with redundant non-defective LUTs). We have a twofold objective to implement wire level defect tolerance: First, wire level repair techniques are able to target highest defect rates since they provide maximum resolution for implementing repair. Secondly, since the defects occur in nanofabric, it is practical to incorporate different defect distribution models at wire level in order to propose most efficient repair scheme (as compared to defect tolerance implemented at higher level of abstraction). The general repair concept proposed in this work is derived from a memory repair technique that has been used in CMOS rather than nanodevice based systems [11] as outlined earlier in section I. The technique proposed in [11] is not applicable by itself to LUT based architecture proposed in this work because an individual LUT size is much smaller as compared to a highly dense memory architecture targeted in [11], hence we do not require replacement of blocks of memory unit which is the key difference between our proposed techniques and the memory repair techniques. Moreover, the original memory architecture if applied to LUT based nano/CMOS architecture will impose a significant CMOS area overhead due the the presence of a mapping table (Eq.1) which will nullify the gain in device density achieved by using nano components. Hence we propose the following mechanism to make the algorithm reported in [11] more suitable to LUT based nano/CMOS architecture. 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, is called the *Modified Tagged Repair* method which will be discussed in section IV-B. The overall area estimate for the implementation shown Fig. 2 for the two proposed techniques will be:

$$\begin{aligned} A_{total} &= A_{LUT\ \ decoders} + L \times [A_{LUT\ \ unit} \\ &+ A_{LUT\ \ unitdecoder/tagging} + A_{Wiring/Routing}] \end{aligned} \qquad (2)$$

Comparing Eq. 2 and Eq. 1 shows that the proposed techniques have an area advantage over the Repair Most implementation due to the absence of mapping table. The size of the mapping table can be significantly large depending on the block size of unit being repaired. The tagging mechanism for the proposed techniques is relatively simple and can be accommodated in the LUT unit decoder circuit. Since the primary motivation of this work is to propose repair techniques for a defect tolerant hybrid nano/CMOS computational architecture, we have not focussed on the fabrication issues related to LUT architecture and interconnect issues between nano and CMOS components [26]. For example, one way to implement CMOS tags can be to use nanoscale FET as a capacitive switching device [25], [27].

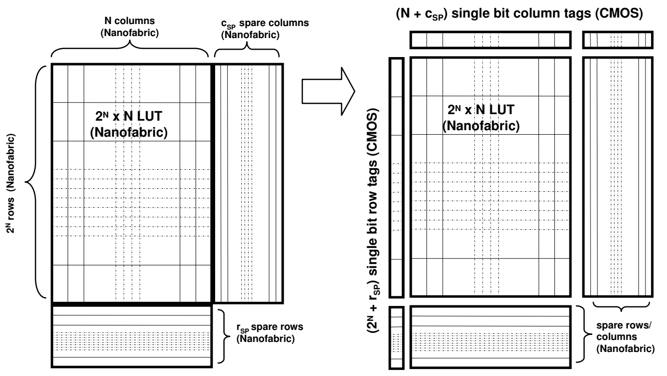

N columns (Nanofabric)
csp spare columns (Nanofabric)
2N x N LUT (Nanofabric)
2N rows (Nanofabric)
rsp spare rows (Nanofabric)
(N + csp) single bit column tags (CMOS)
(2N + rsp) single bit row tags (CMOS)
2N x N LUT (Nanofabric)
spare rows/ columns (Nanofabric)

Fig. 3. Tagged Repair Technique: Implementation for a $2^N \times N$ LUT using 1-bit CMOS tags

A. Tagged Repair Technique

Fig. 3 shows the implementation of the Tagged Repair technique. For a LUT of size $2^N \times N$, we provide c_{sp} spare columns and r_{sp} spare rows. It can be seen that the overall nanodevice area of this technique is $c_{sp} \times r_{sp}$ less than Repair Most technique [14] shown in Fig. 1(a). Unlike the Repair Most technique, there is no need to specify the values of row threshold and column threshold. This technique uses a tagging method to tag rows and columns that are least defective. Initially the tags for the original 2^N rows and N columns in LUT are set to 1 and tags for the spare rows and spare columns (r_{sp} and c_{sp} respectively) are set to 0. The implementation algorithm for the Tagged Repair technique is:

```
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 aim of our proposed Tagged Repair technique is to identify a defect-free instance of a LUT of size $(2^N \times N)$ 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), which is given by the following equation:

$$P_{col}(L) = \sum_{k=L+1}^{r+L} \binom{r+L}{k} P^k (1-P)^{r+L-k} \tag{3}$$

where r and c are the number of rows (2^N) and columns (N) of the LUT respectively, P is the defect rate of the fabric and $L = r_{sp}$ for Tagged Repair technique. To successfully create an instance of the LUT on the fabric, columns should not only have less than L (r_{sp}) defective bits but also there should be at least r defect-free rows in at least c columns that are aligned. We have illustrated this with an example in Fig. 4 where the size of the LUT is 4×3. Although the number of defective bits in column 3 are less than L, it was excluded because its defect-free bits are not aligned with the defect-free bits in the other columns. Hence, the probability of a column being excluded is equal to the sum of probabilities of being defective and not defective but not aligned with the other non-defective columns.

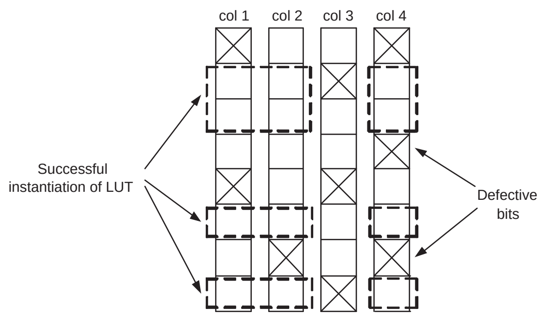


Fig. 4. A 4×3 LUT example using Tagged Repair technique: defect-free bits in columns should be aligned to ensure successful instantiation of LUTs.

To predict the probability of failure for a defective LUT, we first calculate the probability that two non-defective columns $col1$ with k defective bits $(0 \le k \le L)$ and $col2$ with k' defective bits $(0 \le k' \le L)$ are aligned:

$$P(col1, col2, L) = \binom{r+L}{k} P^k (1-P)^{r+L-k} \times \sum_{\substack{n=0 \\ k+k'-n \le L}}^{k'} \binom{k}{n} \binom{r+L-k}{k'-n} \times P^{k'} (1-P)^{r+L-k'} \tag{4}$$

where $k' \le k$. The probability that $col1$ and $col2$ are not aligned with each other is given by the following equation:

$$P'(col1, col2, L) = \binom{r+L}{k} P^k (1-P)^{r+L-k} \times \sum_{\substack{n=0 \\ k+k'-n > L}}^{k'} \binom{k}{n} \binom{r+L-k}{k'-n} \times P^{k'} (1-P)^{r+L-k'} \tag{5}$$

Using equations (3), (4) and (5), we can estimate the probability that n columns out of $(c + c_{sp})$ are not defective and aligned. This is given by the following equation:

$$P_{inst}(n, L) = \sum_{a=1}^{n} \Bigg[\Big(\prod_{\substack{b=1 \\ a \ne b}}^{n} \sum_{\substack{k=0 \\ k'=0}}^{L} P(a, b, L) \Big) \times \prod_{d=1}^{c+c_{sp}-a} \Big(\sum_{\substack{k=0 \\ k'=0}}^{L} P'(a, d, L) + P_{col}(L) \Big) \Bigg] \tag{6}$$

Hence, 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 $L = r_{sp}$:

$$P_{succ} = \sum_{x=c}^{c+c_{sp}} \binom{c+c_{sp}}{x} P_{inst}(x, r_{sp}) \tag{7}$$

and therefore, the overall failure rate is:

$$P_{failure} = 1 - P_{succ} \tag{8}$$

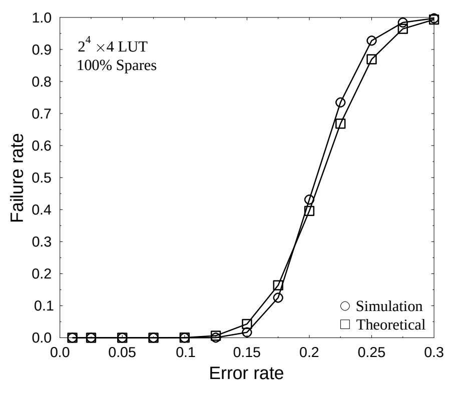


Fig. 5. Tagged Repair technique: Failure rate obtained both by theory and simulation for a $2^4 \times 4$ LUT and 100% redundancy.

Fig.5 illustrates the failure rate obtained both theoretically, based on Eq. 8 and by simulation. There is a strong correlation between the two plots, validating the derived theoretical equations. The slight mismatch in the two plots is due to the rounded off values of failure rate used in the simulated plot as compared to exact data values in the theoretical plot.

B. Modified Tagged Repair Technique

As shown in section V, Tagged Repair technique (Fig. 3) is capable of handling upto 17% defect rate in a hybrid nano/CMOS architecture implemented as LUTs. To address even higher defect rates, we propose another technique as presented in Fig. 6, which is called Modified Tagged Repair technique. As can be seen in Fig. 6, in Modified Tagged Repair technique instead of replacing entire columns, we have split the columns in two equal sections before applying tagging and replacement, to make more optimized use of the spare units as compared to the Tagged Repair technique.

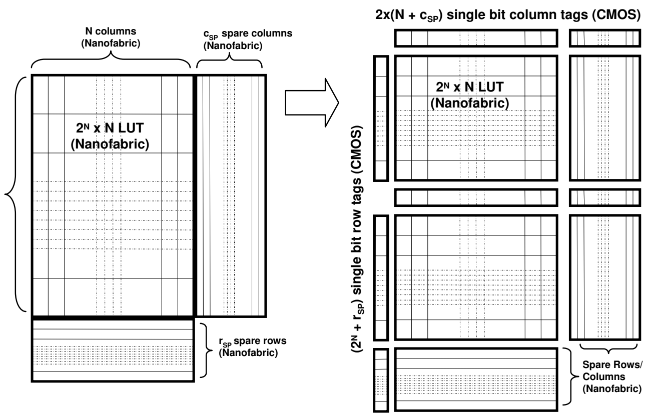

N columns (Nanofabric)
c_{SP} spare columns (Nanofabric)
2^N rows (Nanofabric)
2^N x N LUT (Nanofabric)
r_{SP} spare rows (Nanofabric)
2x(N + c_{SP}) single bit column tags (CMOS)
(2^N + r_{SP}) single bit row tags (CMOS)
2^N x N LUT (Nanofabric)
Spare Rows/ Columns (Nanofabric)

Fig. 6. Modified Tagged Repair Technique Implementation for a $2^N \times N$ LUT using 1-bit CMOS tags

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.(9) represents the total probability P_{succ} of instantiating a $2^N \times N$ LUT with r_{sp} spare rows and c_{sp} spare columns. Variable i in Eq.(9) represents 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:

$$\begin{aligned} P_{succ} &= \sum_{i=0}^{r_{sp}} \Bigg[\Big(\sum_{x=c}^{c+c_{sp}} \binom{c+c_{sp}}{x} P_{inst}(x,i) \Big) \\ &\quad \times \Big(\sum_{x'=c}^{c+c_{sp}} \binom{c+c_{sp}}{x'} P_{inst}(x', r_{sp}-i) \Big) \Bigg] \end{aligned} \quad (9)$$

The implementation algorithm used for the Modified Tagged Repair technique is similar to Tagged Repair technique but it has the following distinctive feature. In the algorithm for the 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 elements (2^N) in each column, hence it is easy to split each column halfway in size 2^{N-1}. A similar technique to split and tag rows cannot be used since the size of a row can be odd or even depending on the value of N and an odd value of N cannot be split in two equal integers. The downside of using this technique is that it will cause an increase in CMOS area overhead of the tagging circuitry, increasing the implementation cost (as seen later in section V-B. The number of column tags required will be double that of the Tagged Repair technique while the number of row tags remain the same.

V. Simulation Results

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 that each element of the LUT stores a 0 or 1 are equal. The LUTs are of sizes ranging from $2^3 \times 3$ to $2^6 \times 6$. We also experimented with larger circuits such as ISCAS'85 benchmarks by synthesizing them into smaller LUTs using synthesis tools such as Synplicity [28]. We estimate the Failure rate (or circuit failure probability), $P_{failure}$, resulting from randomly injecting m defects, by calculating the ratio of defective LUTs after repair to the total number of simulation iterations $I = 5000$. In our experimental simulations we have seen that a value of $I > 5000$ is suitable to obtain a convergence for $P_{failure}$. The 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. Further in section V-C, we investigate the effect of Gaussian and row/column defect distribution on the circuit failure probability ($P_{failure}$). All the simulations were carried out in C++ and the results were compared with Repair Most technique [14] to determine the gain in repair capability.

A. Repair Capability of the Proposed Techniques

Fig. 7 shows the plot of failure rate Vs defect rate using Tagged Repair technique for different LUT sizes with percentage redundancy varying between 25% to 100%. As can be seen in Fig. 7(a) at 25% redundancy, the Tagged Repair technique exhibits higher defect tolerance in the case of smaller sized LUTs (such as $2^3 \times 3$ LUT) than larger LUTs (such as $2^6 \times 6$ LUT). This is because for a particular defect rate, the likelihood of finding a non-defective row/column to replace a defective one decreases since the number of defects in the LUT increase as the LUT size increases. Since we have defined P_{succ} as the probability of obtaining a defect-free LUT instantiation, the presence of even a single defect (after the repair process) is considered as a failure. Hence synthesis of larger circuits into smaller LUTs will result in improved defect tolerance for the targeted nano/CMOS architecture. Similarly, we have simulated the failure rate Vs defect rate plots for 50%, 75% and 100% redundancy shown in Fig. 7(b)-(d). An example of $2^3 \times 3$ LUT using 100% spares is used to compare the repair capability of this technique with the Repair Most technique. As can be seen in Fig. 1(c) for 100% redundancy and Fig. 7(d) we can see that the Repair Most could only

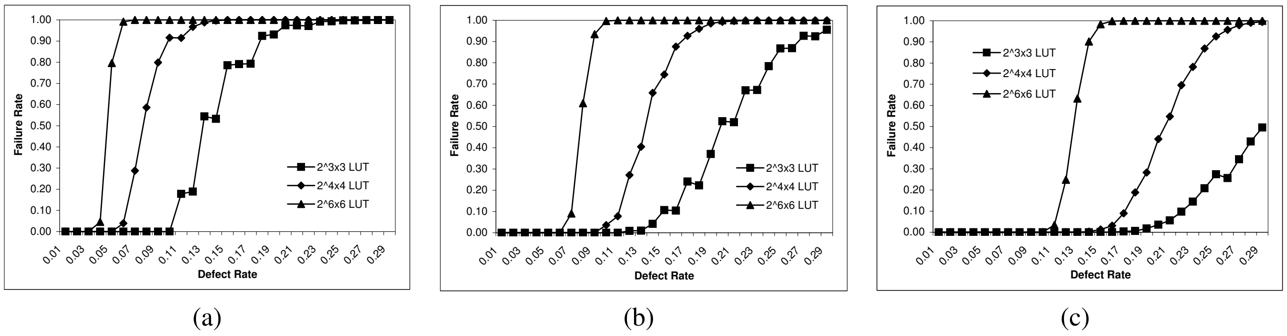


Fig. 7. Plot of Failure rate Vs Defect rate using Tagged Repair technique for different LUT sizes with (a) 25% redundancy (b) 50% redundancy and (c) 100% redundancy in rows and columns.

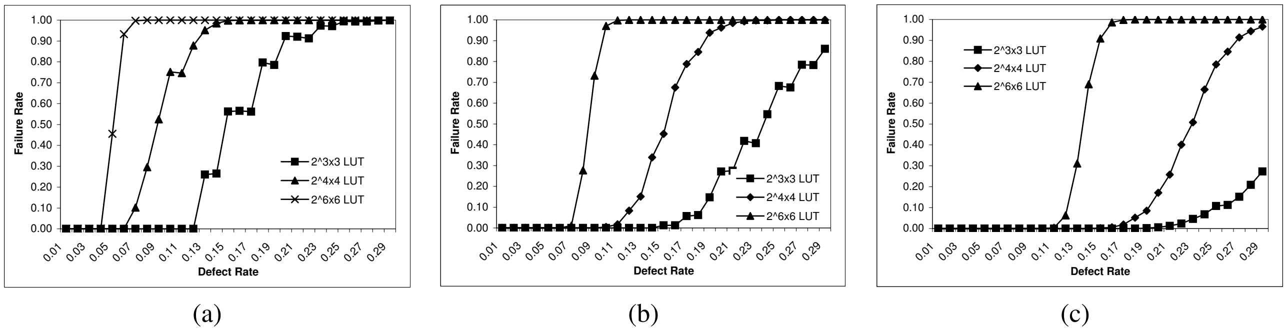


Fig. 8. Plot of Failure rate Vs Defect rate using Modified Tagged Repair technique for different LUT sizes with (a) 25% redundancy (b) 50% redundancy and (c) 100% redundancy in rows and columns.

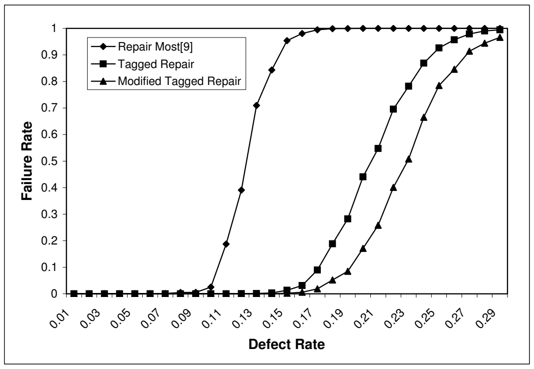


Fig. 9. Comparative study for the failure rate of $2^4 \times 4$ LUT with 100% redundancy

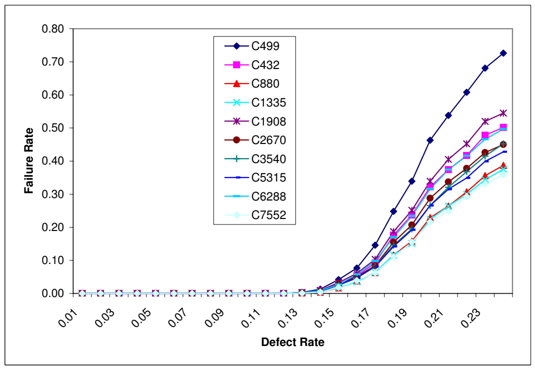


Fig. 10. Failure probability of synthesized ISCAS'85 benchmark circuits using Modified Tagged Repair technique.

handle defect rates $< 10\%$, whilst the Tagged Repair technique with 100% redundancy can target defects upto 17%.

Fig. 8 shows the plot of the Modified Tagged Repair technique for different LUT sizes with varying redundancy. Taking an example of a $2^3 \times 3$ LUT with 100% redundancy, we compare the results of Tagged Repair technique (Fig. 7(d)) with Modified Tagged Repair technique (Fig. 8(d)). It can be seen that while the Tagged Repair technique can achieve 0% failure rate at defect rates of upto 17%, the Modified Tagged Repair technique can target defect rate upto 20%. This improvement in repair capability $(1 - P_{failure})$ is due to the more optimized usage (by splitting the columns in two before applying repair) of the redundant spare units.

Fig. 9 compares the repair capability of the proposed techniques with the Repair Most technique [14] for LUTs of size $2^4 \times 4$ LUT with 100% redundancy ($c_{sp} = 4$ spare columns and $r_{sp} = 2^4$ spare rows). 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 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 almost fails completely.

To assess the repair capability of our proposed techniques for larger circuits, we performed an analysis on the ISCAS'85 benchmark circuits using the Modified Tagged Repair technique. The failure rates for synthesized ISCAS'85 circuits are shown in Fig. 10. The ISCAS'85 benchmark circuits were first synthesized into smaller LUTs sizes (between $2^2 \times 2$ to $2^6 \times 6$). However, as can be seen from Table I, majority

	$2^2 \times 2$ LUTs	$2^3 \times 3$ LUTs	$2^4 \times 4$ LUTs	$2^5 \times 5$ LUTs	$2^6 \times 6$ LUTs	Targeted Defect Rate
C499	0	0	2	2	8	13.0%
C432	3	1	2	1	5	13.0%
C880	1	4	2	4	5	14.0%
C1335	5	0	4	5	5	13.0%
C1908	2	2	2	4	10	13.0%
C2670	2	4	5	3	9	13.0%
C3540	6	2	10	13	22	13.0%
C5315	8	8	12	8	25	13.0%
C6288	20	4	14	9	48	13.0%
C7552	6	18	16	20	30	14.0%

TABLE I
TARGETED DEFECT RATE OF ISCAS'85 BENCHMARK CIRCUITS SYNTHESIZED INTO SMALLER $2^N \times N$ LUTS USING THE MODIFIED TAGGED REPLACEMENT TECHNIQUE.

Spares	LUT size	Repair Most [14]	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 II
COMPARATIVE REPAIR COST OF THE PROPOSED TECHNIQUES WITH THE REPAIR MOST TECHNIQUE IN TERMS OF TARGETED DEFECT RATE

of the synthesized circuits contain a higher proportion of $2^5 \times 5$ and $2^6 \times 6$ LUTs and hence the targeted defect rate of various benchmark circuits is around 13%-14% with little variation. This can be addressed by further synthesizing the circuits into LUTs of smaller sizes. Recently, a defect tolerance technique was reported in [16] for nano/CMOS architecture and it has been shown to achieve 87% success in mapping of synthesized ISCAS'85 benchmark circuits at 10% defect rate which is less than what is achieved with the proposed techniques (100% success for upto 14% defect rate). In [16], the authors used a mapping technique with a greedy algorithm on synthesized ISCAS'85 benchmark circuits implemented as LUTs. Since the size of synthesized LUTs in [16] is $N \times 1$, it only permits column replacement of defective columns, since row replacement will be quite expensive to achieve.

B. Estimation of Repair Cost

As shown in section IV-A and V-A, the proposed Tagged Repair technique uses considerably less redundancy to tolerate even higher defect rates compared to Repair Most technique. This reduction is explained as follows: the overall nanodevice area advantage of Tagged Repair technique as compared to Repair Most for a $2^N \times N$ LUT with c_{sp} spare columns and r_{sp} spare rows will be $c_{sp} \times r_{sp}$. With 100% redundancy ($c_{sp} = N$ and $r_{sp} = 2^N$), this will result in a 25% advantage in nanodevice area. To calculate the nanodevice area advantage of the proposed techniques quantitatively, the total nanodevice area of $2^4 \times 4$ LUT implementation with 100% spares for the proposed techniques (Fig. 3 and 6) will be $3 \times 2^4 \times 4 = 192$ units/LUT. However, due to the implementation architecture of Repair Most technique (Fig. 1), we can see that the total area (including original LUT and the spare units) of the Repair Most implementation will be $4 \times 2^4 \times 4 = 256$ units/LUT, which is 25% more as compared to the proposed techniques. Table II shows the comparative repair cost of the proposed techniques and 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 10% defect with only 25% spares, while Repair Most is not able to target 10% defect rate with even 100% spares. The targeted defect rate values given in this table have been rounded off to the nearest 1.0%. Similarly the amount of spare units have been rounded off to the nearest whole number based on percentage of spares.

As can be seen from Fig 3(b), the CMOS area overhead for the for a $2^N(rows) \times N(columns)$ LUT with spare rows (r_{sp}) and spare columns (c_{sp}) using Tagged Repair technique is $(2^N + r_{sp})$ single bit row tags and $(N + c_{sp})$ single bit column tags. When compared to the Tagged Repair technique, the Modified Tagged Repair technique will require an extra $(N + c_{sp})$ single bit CMOS column tags (making it a total of $2 \times (N + c_{sp})$ column tags). Here the multiplication factor 2 accounts for the splitting of columns into equal halves to obtain better repair capability. Considering a single bit SRAM cell requires 6 transistors [29], the overall CMOS area overhead in terms of transistor count can be calculated accordingly. The number of row tags will be equal to the Tagged Repair technique. For example, in case of a $2^4 \times 4$ LUT with 100% redundancy, the CMOS area overhead for Tagged Repair technique will be a total of $2 \times (2^4 + 4) = 40$ single bit tags. However, for the Modified Tagged Repair technique, the CMOS area overhead will be $2 \times (2^4 + 4 + 4) = 48$ tags for a single LUT which will result in $8 \times 6 = 48$ extra transistors/LUT as compared to Tagged Repair technique.

C. Effect of Defect Distribution Model

Fig. 11 shows the effect of defect distribution on the repair capability of the proposed repair techniques. As can seen, with Gaussian and row/column defect distribution, the Modified Tagged Repair technique can target defect rates upto 22% and 27% respectively. The row/column defects in this simulation are assumed to be concentrated along the columns. The mean and standard deviation values for the Gaussian defect distribution plot are 0 and 2.0 respectively. Another interesting observation from the plot shown in Fig. 11 is the rate of change of Failure rate. While the Failure rate of for random defects shows a smooth curve, there is an abrupt increase in failure rate for the Gaussian defect distribution. This abrupt change in failure rate (for $> 22\%$ defect rate) can be attributed to a large number of defects that lie outside the cluster. The failure rate for row/column defect distribution increases much more slowly.

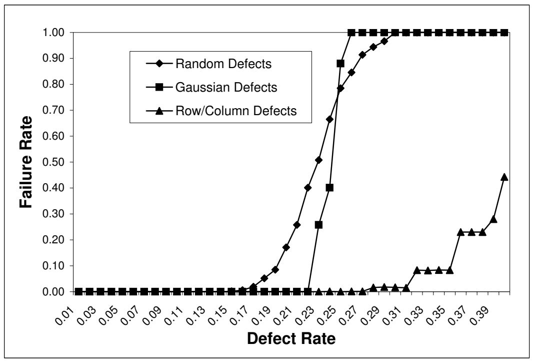


Fig. 11. Effect of defect distribution on defect tolerance for a $2^4 \times 4$ LUT with 100% redundancy using the Modified Tagged Repair technique

VI. Conclusion

Two new and effective Tagged Repair techniques for hybrid nano/CMOS architecture implemented as LUTs have been proposed. The techniques achieve higher defect tolerance than recently reported repair techniques with upto 20% defect rate for a $2^3 \times 3$ LUT implementation and upto 14% defect rate for synthesized ISCAS'85 benchmark circuits.

VII. Acknowledgement

We acknowledge the EPSRC (UK) for funding this project in part under grant EP/E035965/1 and the Algerian Ministry of Higher Education and Scientific Research.

References

[1] M. Ziegler and M. Stan, "CMOS/nano co-design for crossbar-based molecular electronic systems," *Nanotechnology, IEEE Transactions on*, vol. 2, pp. 217–230, Dec. 2003.
[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] 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.
[4] 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.
[5] M. Stan, P. Franzon, S. Goldstein, J. Lach, and M. Ziegler, "Molecular electronics: from devices and interconnect to circuits and architecture," *Proceedings of the IEEE*, vol. 91, pp. 1940–1957, Nov 2003.
[6] M. Jacorne, C. He, G. de Veciana, and S. Bijansky, "Defect tolerant probabilistic design paradigm for nanotechnologies," *DAC '04. Proceedings. 41st*, pp. 596–601, 2004.
[7] M. Tahoori, "Defects, yield, and design in sublithographic nanoelectronics," in *20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2005*, pp. 3–11, Oct. 2005.
[8] A. DeHon and H. Naeimi, "Seven strategies for tolerating highly defective fabrication," *Design and Test of Computers, IEEE*, vol. 22, pp. 306–315, July-Aug. 2005.
[9] M. B. Tahoori, "Application-independent defect-tolerant crossbar nanoarchitectures," in *ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, (New York, NY, USA), pp. 730–734, ACM, 2006.
[10] M. Mishra and S. Goldstein, "Defect tolerance at the end of the roadmap," *In ITC*, vol. 1, pp. 1201–1210, 30-Oct. 2, 2003.
[11] S.Lu and C. Hsu, "Fault Tolerance Techniques for High Capacity RAM," *IEEE Transactions on Reliability*, vol. 55, pp. 293–304, June 2006.
[12] D. Chang, J. Li, and Y. Huang, "A Built-In Redundancy-Analysis Scheme for Ramdom Access Memories with Two-Level Redundancy," *Journal of Electron Test*, vol. 24, no. 1, pp. 181–192, 2008.
[13] W. Zhang, N. K. Jha, and L. Shang, "NATURE: a hybrid nanotube/CMOS dynamically reconfigurable architecture," in *DAC '06: Proceedings of the 43rd annual conference on Design automation*, (New York, NY, USA), pp. 711–716, ACM, 2006.
[14] D. B. Strukov and K. K. Likharev, "Prospects for terabit-scale nanoelectronic memories," *Nanotech.*, vol. 16, no. 1, pp. 137–148, 2005.
[15] D. Strukov and K. Likharev, "CMOL FPGA: a reconfigurable architecture for digital circuits with two-terminal nanodevices," *Nanotechnology*, vol. 16, no. 6, pp. 888–900, 2005.
[16] 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.
[17] 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.
[18] 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.
[19] D. Strukov and K. Likharev, "Defect-tolerant architectures for nanoelectronic crossbar memories," *Journal of Nanoscience and Nanotechnology*, vol. 7, pp. 151–167, Jan 2007.
[20] S. Goldstein and M. Budiu, "NanoFabrics: spatial computing using molecular electronics," *IEEE ISCA*, pp. 178–189, 2001.
[21] 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.
[22] A. DeHon, "Array-based architecture for FET-based, nanoscale electronics," *Nanotechnology, IEEE Transactions on*, vol. 2, pp. 23–32, Mar 2003.
[23] A. DeHon, "Nanowire-Based Programmable Architectures," *ACM Journal of Emerging Technologies in Computing Systems*, vol. 1, pp. 109–162, July 2005.
[24] G. S. Snider and R. S. Williams, "Nano/CMOS architectures using a field-programmable nanowire interconnect," *Nanotechnology*, vol. 18, no. 3, p. 035204 (11pp), 2007.
[25] S. Li and T. Zhang, "Exploratory study on circuit and architecture design of very high density diode-switch phase change memories," in *ISQED '09: Proceedings of the 2009 10th International Symposium on Quality of Electronic Design*, (Washington, DC, USA), pp. 424–429, IEEE Computer Society, 2009.
[26] S. Li and T. Zhang, "Hybrid resistor/fet-logic demultiplexer architecture design for hybrid cmos/nanodevice circuits," in *Computer Design, 2007. ICCD 2007. 25th International Conference on*, pp. 574–579, Oct. 2007.
[27] K. Chakraborty and P. Mazumdar, "Fault-Tolerance and Reliability Techniques for High-Density Random-Access Memories," *Prentice Hall*, . 2002.
[28] "http://www.synplicity.com/,"
[29] A. Bellaouar and M. Elmasry, "Low-Power Digital VLSI Design: Circuits and Systems," *Springer Publication*, . 1995.

Saket Srivastava received the Ph.D. degree in electrical engineering from the University of South Florida, Tampa, USA, in 2008.

He worked as a postdoc researcher at the shool of electronics and computer science, University of Southampton, Southampton, U.K.

Aissa Melouki received the M.Eng. degree in computer systems engineering from the University of Warwick, Coventry, U.K., in 2006. He is currently working towards the Ph.D. degree in electronics at the University of Southampton, Southampton, U.K.

His current research interests include developing fault and defect tolerance techniques for nanometre CMOS and nanoscale devices.

Bashir M. Al-Hashimi (F'09) received the Ph.D. degree in electronic system design from York University, York, U.K., in 1989.

He is a Professor of computer engineering at the University of Southampton, Southampton, U.K.