Online Multiple Error Detection in Crossbar Nano-architectures

Navid Farazmand[1] and Mehdi B. Tahoori[1,2]
[1]*Department of Electrical and Computer Engineering*
Northeastern University, Boston, MA

[2]*Faculty of Informatik (ITEC)*
Karlsruhe Institute of Technology, Germany
Emails: {nfarazma,mtahoori}@ece.neu.edu

***Abstract*— Crossbar nano-architectures based on self-assembled nano-structures are promising alternatives for current CMOS technology, which is facing serious challenges for further down-scaling. One of the major challenges in this nanotechnology is elevated failure rate due to atomic device sizes and inherent lack of control in self-assembly fabrication. Therefore, high permanent and transient failure rates lead to multiple faults during lifetime operation of crossbar nano architectures.**

In this paper, we present a concurrent multiple error detection scheme for multistage crossbar nano-architectures based on dual-rail implementations of logic functions. We prove the detectability of all single faults as well as most classes of multiple faults in this scheme. Based on statistical multiple fault injection, we compare the proposed technique with other online error detection and masking techniques such as Triple Module Redundancy (TMR), duplication, and parity checking, in terms of fault coverage as well as area and delay overhead.

I. Introduction

Among different alternatives, Carbon Nano-Tubes (CNT) and semiconductor Nano-Wires (NW) have shown to be promising materials to be used in the fabrication of nano-electronic circuits for the continuation of Moore's law beyond CMOS limitations. These materials are suitable for the implementation of active devices such as diodes and transistors as well as interconnect wires and programmable switches [1], [2], [3].

Bottom-up self-assembly is the main fabrication method used to manufacture and combine these basic devices to form electronic circuits. Lack of full control and preciseness in self-assembly process makes it mostly suitable for creating regular crossbar structures [3]. Nano-crossbars are realized with crossed carbon nano-tubes and/or nano-wires. They provide interconnects as well as logic elements by implementing nano-devices and programmable switches at the crosspoints. Different Programmable Logic Array (PLA)-like architectures using nano-crossbars as the basic block have been proposed to enable the realization of large circuits [4], [2], [5].

Due to high susceptibility of nano-crossbars to transient faults and permanent defects during system operation, incorporating suitable Fault Tolerance (FT) techniques into their design is very important [6], [7], [8]. A major step in FT is *Concurrent Error Detection* (CED). A critical requirement for FT methods targeting nano-crossbars is the ability to cope with multiple faults, as opposed to the single fault assumption in traditional FT techniques. This is due to very high defect rates, both permanent and transient, arising from self-assembly process uncertainties and very small feature sizes.

Multiple fault testing has been addressed in PLA testing techniques [9], [10], [11], [12], [13]. The main problem with most of these techniques is large test time and offline test application, making them incapable of handling transient faults [12]. Also, some restrictive assumptions on the occurrence of faults (e.g. faults occurring one by one) [11], [13] or the circuit under test [12] limits their applicability for nano architectures. In addition, checkers are required to be inserted at every stage. This raises important challenges such as fault detection in the checkers and their relatively high area and performance overhead [12], [11].

There has been some recent work on fault tolerance of PLA-like nano-crossbars [8], [6], [14], [7]. The technique proposed in [8] relies on intensive connection for every nano-crossbar with its neighbor crossbars to recover from multiple faults. Availability of such interconnect resources seems to be infeasible in current nano-architectures. Furthermore, the method presented in [7] has the assumption of reliable connections to CMOS substrate at the input/output of every nano-crossbar stage, which is somehow impractical.

In this paper, we provide an efficient online error detection scheme for detection of multiple faults (permanent and temporary) in nano-crossbars and compare it with various existing techniques. The basic idea of this scheme is to exploit *dual rail* logic implementation, primarily used in diode-based nano-crossbars [3], [1], [2], for error checking. Detectability analysis of this method for single and multiple faults is presented. An important feature of this scheme is that it requires checking only at the final stage, eliminating large area and performance overhead associated with intermediate checkers (typically implemented in reliable CMOS substrate), as required by other techniques. We also discuss some alternative implementations of this scheme to further improve its fault coverage. We have performed extensive experimental comparison of this scheme with major online error detection/masking methods namely NMR, duplication,

This work was supported in part by the National Science Foundation Grant No. CCF-0746836.

978-1-4244-5028-2/09/$25.00 ©2009 IEEE

and parity checking with respect to multiple fault detection coverage as well as area and delay overhead.

The rest of the paper is organized as follows. Section II presents some preliminaries regarding nano-crossbars along with a review of previous work. The proposed dual rail error checking scheme is described in Section III. Implementations of various online error detection schemes are presented in Section IV. Experimental results for different schemes are discussed in Section V. Finally, Section VI concludes the paper.

II. Preliminary and Previous Work

A. Crossbar nano-architectures

Carbon Nano-Tubes and semiconductor Nano-Wires can be used to produce programmable logic and interconnects [15], [3]. The self-assembly process is used to form crossbar structures from CNTs and NWs [3]. In order to implement interconnects, programmable switches can be formed at the crosspoint of nano-wires [3] (Figure 1). Also, it is possible to implement diodes and transistors at the crosspoints [15], [16], [17]. By combining programmable switches and diodes, a programmable logic fabric can be formed. Different architectures have been proposed based on such diode-based programmable crossbar structures. Nano logic arrays called *Nanoblocks* implementing Resistor-Diode Logic (RDL) form the basic block of the FPGA-like architecture called *NanoFabric* [3]. In order to implement a complete logic family with such diode-based PLA-like crossbar structures, both desired functions and their complements should be implemented at each stage, assuming inputs and their complements are available. Crossbar structures realized with transistors at the crosspoints have also been proposed. Array-based nano architecture using *Programmable Logic Arrays* (PLAs) has been presented [5]. The main building block, called the nano Programmable Logic Array (nanoPLA), is built from a crossed set of N-type and P-type nanowires. The nanoPLA is programmed using lithographic-scale wires. *Molecular CMOS* (CMOL) is another architecture proposed in [4] designed using the same crossbar array structure as the nanoPLA design consisting of two levels of nanowires.

B. Previous work

We review the previous work on online error detection for programmable fabric with respect to the following criteria:

- transient faults as well as permanent defects detection.
- handling very high failure rates
- multiple faults detection
- concurrent error detection/correction

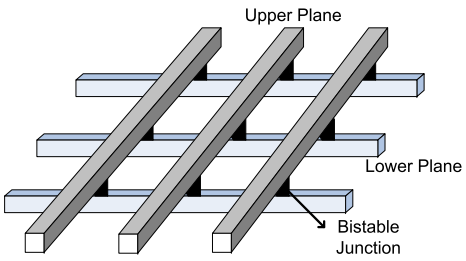

Fig. 1. Programmable switches at the crosspoints ([3])

Extensive work has been done on PLA testing ranging from test generation [18] (externally applied) to concurrent error detection techniques [19]. Design for testability (DFT) has been exploited to simplify test generation and application and improve test coverage [10], [9], [11], [12], [13].

Some techniques reduce test time and cost, and increase fault coverage by introducing Built-In Self Test (BIST) to PLAs [9]. The problem with these techniques is that they are non-concurrent, unable to detect transient faults. To solve this problem concurrent error detection techniques for PLAs have been introduced [12], [13], [11]. The general idea of these techniques is to assign some codewords to inputs/outputs in a way that the output in the presence of faults is either correct or outside the code space [20]. The assumption of incremental fault occurrence (multiple faults occur one by one in time) reduces their effectiveness for multiple fault detection in nano-electronics with high failure rates, causing simultaneous multiple faults. Also, the method in [13] has a limiting assumption of unidirectional errors (only 0-to-1 or 1-to-0 errors). In [12], inputs assumed to be error free, PLA is non-concurrent (only one product line is active for each input combination), different fault types do not occur simultaneously, and the primary focus is on single faults. Moreover, an important drawback of all these techniques is the need for checkers at the inputs and outputs of each PLA stage, causing the following limitations for nano-crossbars:

- Checkers impose high area and performance overhead for large multistage nano-crossbars [12].
- If checkers are placed at each stage, they have to be implemented with unreliable nano-crossbars (CMOS implementation of checkers at all stages is not feasible). Designing testable checkers is a challenging issue [11], [21].

Some recent work considers defect tolerance for yield enhancement [22] and fault tolerance [8] in nano-crossbar arrays based on the reconfigurability of such architectures. To be able to cope with transient faults, fault masking techniques for nanoPLAs have also been presented [7], [6], [14]. The techniques presented in [6] and [14] are based on replication of input, product, or output lines for fault masking with considerable overhead. The goal of [7] is to reduce overhead by selective replication. The required process for indication of critical lines to replicate might be restrictive. Moreover, all inputs/outputs are assumed to be in reliable connection with CMOS level which is not realistic.

The proposed scheme in this paper aims to eliminate the need for input/output checkers and CMOS connections at every stage for intermediate checkers to reduce associated area and performance overhead while preserving very high error coverage.

III. Dual-rail Online Error Detection

A. Definitions, assumptions, and fault model

We assume PLA-like crossbar structures (nano-crossbars) in the form of AND-OR PLA, as shown in Figure 2. A *crosspoint* exists at the intersection of each input and product (product and output) line if it is included in that product term (output). We also assume that the functions implemented in nano-crossbars are dual rail functions; i.e. for each output F_i there is an output $\overline{F_i}$ which is the complement of F_i. Also, it is assumed that both inputs and their complements are available at the inputs of nano-crossbar. This is the inherent characteristic of nano-crossbars based on two terminal devices [3], [1], [2]. Lastly, each logic circuit is implemented with a multi-stage nano-crossbar.

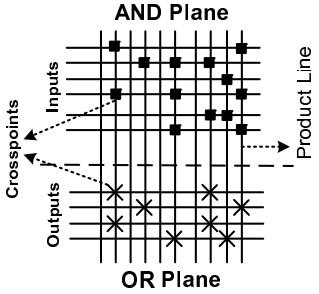

Fig. 2. AND-OR crossbar

Definition 1: Crosspoint insertion happens when some extra crosspoints, due to faults, appear in AND-plane or OR-plane.

Definition 2: Shrinkage refers to the reduction of the number of minterms of the function implemented in nano-crossbar, e.g. due to crosspoint insertion in AND-plane.

Definition 3: Crosspoint deletion occurs when some the existing crosspoints in AND-plane or OR-plane disappear.

Definition 4: Expansion refers to the increase in the number of minterms of the function implemented in nano-crossbar, e.g. due to crosspoint deletion in AND-plane.

The fault model in this paper are as follows [11], [12], [13]:

- *Crosspoint insertion:* any number of crosspoint insertion in AND-plane, OR-plane or both.
- *Crosspoint deletion:* any number of crosspoint deletion in AND-plane, OR-plane, or both.
- *Bridging fault between input lines:* includes any number of possible bridging (wired-or, wired-and) between input lines.
- *Non-dual rail input (NDRI):* any number of non-dual rail inputs to the nano-crossbar, i.e. $\exists\ i, in_i = \overline{in_i}$.

All other defects/failures manifest themselves as a combination of the fault types in the fault model as follows:

- Input stuck-at-0, SA0, (SA1) and output SA0 (SA1) are modeled by '00' ('11') NDRI faults.
- Product line SA0 is modeled by crosspoint deletion in OR-plane.
- Product line SA1 and output SA1 are equivalent to multiple '11' NDRIs.
- Bridging faults between input and product lines are equivalent to crosspoint insertion in the AND-plane. Also, bridging faults between product lines are equivalent to multiple crosspoint insertions in the AND-plane.

B. Online fault detection with dual rail checking

The basic idea of the proposed scheme is to use dual-rail implementation of a function in nano-crossbars and check this property for multiple fault detection. Fault detection is done by checking the outputs of the circuit, i.e.:
error signal raised iff $\exists\ i,\ F_i = \overline{F_i},\ F_i \in \{PO\}$

In multistage nano-crossbars only primary outputs of the circuit are checked. *Undetected error* occurs when some outputs are different from their fault-free (expected) values but they are still dual rail, i.e. $F_i \neq F_{i\_exp}$ and $F_i \neq \overline{F_i}$. Since output F_i and its complement $\overline{F}_i$ in a dual rail function have no common product terms, it is straightforward that any single fault affects only one of them and, hence. is detected. However, the behavior of the dual-rail nano-crossbar in the presence of multiple faults is more complicated. Multiple fault detectability is discussed next.

Lemma 1: Any number of multiple crosspoint deletion (insertion) faults in the AND-plane along with any number of crosspoint insertion (deletion) faults in the OR-plane will be detected by checking the dual rail primary outputs of a nano-crossbar.

Proof 1: In the fault free circuit, the set of minterms for F_i and $\overline{F_i}$, $m(F_i)$ and $m(\overline{F_i})$, have the following properties: $m(F_i) \cap m(\overline{F_i}) = \phi,\ \ m(F_i) \cup m(\overline{F_i}) = \underline{1}$

Crosspoint deletion in the product lines of F results in expansion in the $m(F)$. A crosspoint deletion behaves as a variable deletion in the boolean function of a product term which expands the number of minterms covered by the product term. Therefore, multiple crosspoint deletion in the AND-plane results in expansion in at least one output, F_i. When expansion happens in F_i, $\overline{F_i}$, or both, the intersection of resulting sets of minterms is no longer empty, i.e. $m(F_i) \cap m(\overline{F_i}) \neq \phi$. Therefore, for some input combinations both F_i and $\overline{F_i}$ are '1'. In such cases during normal operation, the output $F_i\overline{F}_i$ is either error free or non-dual rail in the form of '11'. By checking the values of F_i and $\overline{F}_i$ the non-dual rail erroneous outputs will be detected. Note that crosspoint insertion in OR-plane adds some product terms to the function which in turn results in the expansion in the function (similar to crosspoint deletion in AND-plane). The same conclusion is achieved when considering only crosspoint insertion in the AND-plane and crosspoint deletion in the OR-plane. □

Lemma 1 considers crosspoint faults with the assumption of fault free (dual rail) inputs. However, non-dual rail inputs (NDRI) are likely to occur in multistage nano-crossbars because of non-dual rail outputs due to crosspoint faults in previous stages. Also, in wired-and (wired-or) bridging faults, some input '$a_i\overline{a_i}$' may become '00' ('11'). These NDRI faults can change the behavior of nano-crossbars. Input and output checkers can be used for each stage to

ensure that they are fault free [7], [11], [12], [19]. However, it is not feasible in nano-crossbars, as discussed in Sec. II.

To mitigate checker-related limitations, we propagate errors through the nano-crossbar all the way to the primary outputs. Then, robust checkers could be incorporated only at the primary outputs. The effectiveness of this approach depends on the behavior of nano-crossbars in the presence of NDRI faults, as discussed next.

Lemma 2: All errors due to NDRI faults in the form of '00' ('11') are detectable by checking the primary outputs of nano-crossbar.

Proof 2: We show the proof for '00'. The other case , '11', can be concluded similarly. Having '00' at some inputs changes some product lines erroneously to '0', and in turn, changes at least one output F_i (or $\overline{F}_i$) erroneously '0'. Therefore, the outputs of the nano-crossbar stage with NDRI in the form of '00' are either correct or non-dual rail in the form of '00'. □

Corollary 1: All multiple bridging faults at the inputs of a nano-crossbar are detectable by dual rail implementation.

Multiple wired-and (wired-or) bridging faults are only capable of producing NDRIs in the form '00' ('11'). Due to Lemma 2, all multiple bridging faults are detectable.

Lemma 3: A combination of multiple '00' NDRI faults, wired-and bridging faults, crosspoint insertions in AND-plane and crosspoint deletions in OR-plane are all detectable by dual rail checker.

Lemma 4: A combination of multiple '11' NDRI faults, wired-or bridging faults, crosspoint deletions in AND-plane and crosspoint insertions in OR-plane are detectable by dual rail checker.

In some cases, combination of different fault types may produce undetectable errors, as described below.

Statement 1: A combination of crosspoint insertion and crosspoint deletion in the AND-plane (crosspoint insertion and deletion in OR-plane) may not be detected.

To produce dual rail erroneous outputs, multiple faults should affect both F_i and $\overline{F}_i$ but in different directions. It means that some faults should cause shrinkage (expansion) in F_i and at the same time some other faults should cause expansion (shrinkage) in $\overline{F}_i$. Consider the following function: $F = b\overline{c}d + c\overline{d}$, $\overline{F} = \overline{c}\overline{d} + cd + \overline{b}d$. Suppose that a crosspoint deletion causes variable 'b' to be removed from the first product term of F. Also a simultaneous crosspoint insertion in $\overline{F}$ changes the product term $\overline{b}d$ to $\overline{b}cd$. For $abcd = 1001$, erroneous output ($F\overline{F}$) is '10' (fault-free value is '01'). However, since the output is still dual rail, this multiple fault is undetectable.

Note that even if some faults affect F and $\overline{F}$ in the different directions, they do not necessarily generate undetectable errors. For example in the above function, suppose a crosspoint deletion removes variable 'b' from the product term '$b\overline{c}d$' of F and results in an expansion in F. Also, a simultaneous crosspoint insertion in $\overline{F}$ changes the product term '$\overline{c}\overline{d}$' to '$a\overline{c}\overline{d}$', resulting in a shrinkage in $\overline{F}$. Recalculating the set of minterms for F and $\overline{F}$ shows that, for all input combinations, the output '$F\overline{F}$' is either error free or non-dual rail. Thus there will be no undetected errors.

Statement 2: A combination of multiple '00' and '11' NDRI faults may not be detectable.

Consider the same function as Statement 1. Assume that fault-free input pairs are '$abcd = 0101$' and '$\overline{a}\overline{b}\overline{c}\overline{d} = 1010$'. The fault-free output $F\overline{F}$ is '10'. Due to some faults, input pair '$c\overline{c}$' becomes '00'. At the same time, '$b\overline{b}$' becomes '11'. Then, the output becomes '01'.

C. Extensions of dual rail error checking

1) Hazard-free implementation:

Definition 5: Static Hazard refers to a momentary spurious (glitch) in an output due to a transition between two adjacent input combinations. In order to refer to such glitches as static hazard, the desired output value before and after input transition should be the same [23].

Static ('1') hazard is likely to occur when two adjacent minterms are covered by different cubes and there is no cube covering both. So, when that input changes between these cubes, the output should remain unchanged. However, due to relative delays of gates, both cubes might momentarily be deactivated (output becomes '0'). In order to prevent static hazards in the circuit it is required to incorporate the cube covering the adjacent minterms. Consider the function in Table I. The set of minterms for a logic function is presented along with two different realization for this function. The first one uses minimum cover prime implicants (PI) which minimizes the area used for the circuit. The second one includes another PI which covers the adjacent area between the other two. In the first implementation there is a possibility of having static hazard in the circuit when input changes from $abc = 110$ to $abc = 111$. By adding product term 'ab' to the circuit, for every transition between adjacent input combinations corresponding to output value '1', there is an active product term during the transition, eliminating the possibility of static hazards. In short, including all PIs in the realization of a function, instead of just essential PIs, results in a (static) hazard-free circuit.

Statement 3: Hazard free implementation of a dual rail circuit improves the detectability of errors.

In order to demonstrate the advantage of hazard-free implementation, consider the following situation in Table I. Assume that abc = 110, $\overline{a}\overline{b}\overline{c}$ = 001. Thus F_1 = F_2 = 1, $\overline{F_1}$ = 0. Due to some input errors $c\overline{c}$ becomes 00. This fault causes F_1 to become 0 while F_2 and $\overline{F_1}$ still preserve their values. In fact $F_1\overline{F_1}$ pair is a non-dual rail output while $F_2\overline{F_1}$ is still dual rail and correct. This is an example of the situation in which hazard free implementation prevents NDR input faults from producing NDR output faults in nano-crossbars. In general, NDR output faults, due to NDR input faults on the input term changing between two adjacent minterms covered by different products, could be eliminated by making the circuit hazard-free. Besides reducing the number of errors in the circuit, this implementation helps reducing undetected errors in two ways:

- As mentioned previously, NDR output faults propagating through another stage might result in undetected

errors at primary outputs. Reducing the number of NDR outputs (by hazard-free implementation) could reduce such undetected errors.

- The necessary condition for undetected errors is to have bidirectional faults, i.e. a combination of expansion ('0' to '1' changes) and shrinkage ('1' to '0' changes) in the circuit. Reducing faults in either direction (by hazard-free implementation) helps reducing the number of undetected faults.

TABLE I
MINIMUM COVER AND HAZARD FREE IMPLEMENTATION OF A FUNCTION

		ab 00	01	11	10
c	0	0	0	1	1
	1	1	1	0	0

F_1(minimum cover) = $a\overline{c} + bc$
F_2(hazard free) = $ab + a\overline{c} + bc$
$\overline{F_1}$(minimum cover) = $\overline{a}\ \overline{c} + \overline{b}c$

2) All-minterms implementation: Normally, in a dual rail circuit with $2 \times k$ inputs (k normal and k complement signals), the number of '0' and '1' inputs are the same, i.e. $n_0 = k, n_1 = k$ (n_v : the number of inputs holding value $'v'$). If due to some faults in the previous stages the number of inputs holding value '1' decreases, i.e. $n_0 > k, n_1 < k$, we say that a *Decrease In One* (DIO) has happened in the inputs. Similarly, we have Increase In One (IIO). Assume that the circuit in a PLA-like nano-crossbar is implemented in the form of *Sum Of Minterms* (SOM), in which individual minterms form the products. In this case, each product term is connected to exactly k inputs. Thus, for each input combination, only one product term is active among all products of F and $\overline{F}$, the one with all its k inputs being '1'.

Lemma 5: In SOM dual-rail circuits, all NDRI faults in the form of DIO are detected by NDR outputs in the form of '00'.

Proof 3: In order F or its complement $\overline{F}$ to be '1', there should be an active minterm in the circuit. When DIO occurs in the circuit, the number of inputs holding value '1' is less than k (the number of inputs in each product term). Therefore, no minterm will be activated, resulting in '00' output on $F\overline{F}$.

Since DIO results in '00' NDR outputs in the same stage, it in turn causes DIO in the next stage and the effect propagates all the way to the primary output and will be detected. It is also possible to have DIO in the circuit even if there are some '11' NDRIs. As long as the number of '00' NDRIs is more than '11' NDRIs, the circuit is in DIO situation. According to Statement 2, a combination of '00' and '11' NDRIs may potentially cause undetected errors. However, Lemma 5 suggests that in SOM implementation of a function, whenever DIO happens, the error is guaranteed to be detected.

Lemma 6: In SOM dual-rail circuits, all combinations of DIO along with crosspoint insertion faults in AND-plane and deletion faults in OR-plane are detected.

Proof 4: Even if there is a combination of '00' and '11' NDRIs, while '00' inputs are dominant (the effect is DIO), shrinkage occurs in the set of minterms for some function in the SOM implementation. So, addition of crosspoint insertion (deletion) in AND-plane (OR-plane) cause more shrinkage in the circuit resulting in some NDR output in the form of '00'.

Note that having '11' NDRIs along with crosspoint insertion (deletion) in AND-plane (OR-plane) is one of the cases causing possible undetected errors (Statement 1), which is eliminated here by the SOM implementation.

Lemma 7: All combinations of DIOs along with crosspoint insertion faults in OR-plane are detected.

Proof 5: In SOM implementations, a DIO inactivates all minterms. So, with any number of crosspoint insertions in OR-plane, all circuit outputs will be NDR in the form of '00'.

The above statements suggest that the SOM implementation of dual rail functions could potentially improve the coverage of dual-rail checking. Due to relatively high overhead of this implementation, it might be useful to implement only the critical parts of the circuit, e.g. PO checkers, with this scheme.

IV. CED IMPLEMENTATIONS IN NANO-CROSSBARS

Here we discuss the implementation details of various CED techniques in diode-based nano-crossbars. Figure 3 presents implementation of different techniques for the function provided in Table I.

A. Dual-rail checking

Consider a multi-output function implemented as *sum of products* in a AND-OR PLA (Figure 2). In order to obtain dual rail implementation of this function, the complements of all outputs need to be implemented and mapped together with the function in the same PLA stage, assuming that all inputs and their complements are available (Figure 3.b).

B. NMR fault masking

N-Modular Redundancy (NMR) technique is a general error masking approach which can be used with different odd values for N, starting at 3. We have implemented both 3MR (TMR) and 5MR to evaluate the effect of N on multiple fault detection/masking. Figure 3.d shows TMR implementation for the original circuit of Figure 3.a. As shown in this figure, TMR implementation requires triplication of all product terms and outputs. All three copies of products are identical. Each copy of output uses one set of the triplicated products. Also, another stage is added next to the triplicated stage implementing voter to generate one set of outputs. As can be seen in the figure, for each of the three copies of outputs (which are concluded to one output), three product terms are used in the voter stage. Converting circuits to 5MR is similar. The area overhead of these two methods provided in Table II is based on this implementation.

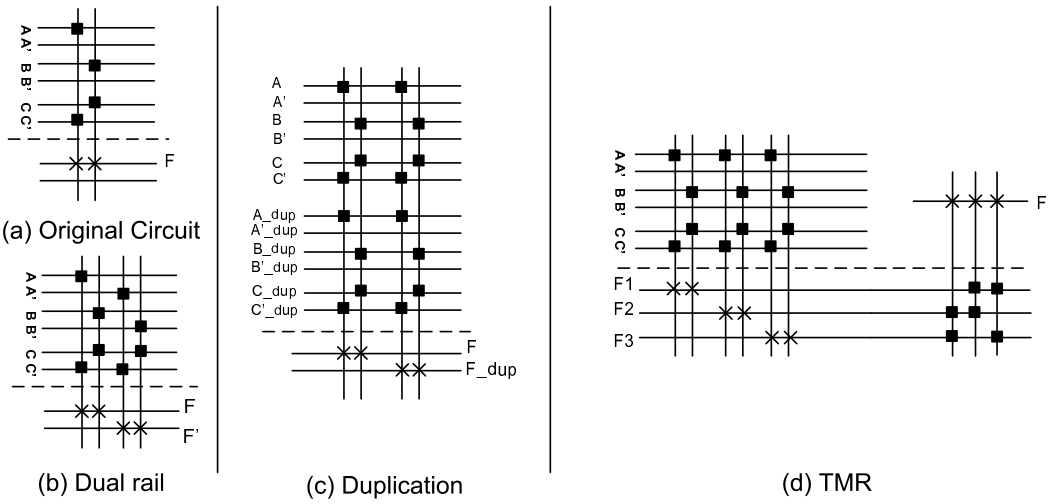

Fig. 3. Implementation of various CEDs in nano-crossbars

C. Duplication

We implement a variation of duplication in nano-crossbar as originally proposed in [14]. In this approach, inputs as well as products and outputs are duplicated (Figure 3.c). Duplicated outputs can be directed to the duplicated inputs of the next stage, all the way to the Primary Outputs (POs). So, Only one checker circuit at POs is required and implemented. This implementation of duplication is suitable for PLAs which provides some level of fault masking [14], [6].

D. Parity checking

This error checking scheme is used for multi-output combinational circuits. For an n output combinational circuit, additional *parity* output is added such that for each input combination, the $n+1$ outputs hold an even (odd) parity (and this is how the truth table for the parity output is constructed). During normal operation, a parity checker for $n+1$ outputs detects the existence of errors in the circuit. In general, this scheme can be extended for more checksum output bits, for better detection/correction capabilities. More information regarding the implementation details of parity checking for nano-crossbars is provided in Section V-B.

Here, we formulate the area and performance overhead of these error checking schemes with respect to the original circuit. The following assumptions and parameters are considered in this analysis.

- The implementations are done for AND-OR PLAs based on diode-based nano-crossbars (i.e. without inversion). Therefore, the original circuit is implemented as dual rail. Area and delay values are presented in terms of number of inputs (i), products (p), and outputs (o) of the dual rail implementation of the function.
- Area cost is represented in terms of the number of crosspoints required to implement each technique.
- Delay values are assumed to be proportional to the number of crosspoints (whether they are active or not) in the critical path from inputs to outputs.

Table II summarizes area and delay costs for different error checking techniques. NMR area cost is due to replicated circuit as well as voter stages. It is obvious that dual rail scheme is far more efficient than NMR in terms of area and delay costs. For the duplication method implemented here, the area cost is about $4.(i.p+p.o)$ which is four times more than dual rail scheme. If the architecture can implement inversion (e.g. FET-based nano-crossbars), the area cost of duplication and dual rail schemes would be the same. The area and delay costs of parity checking are strongly function dependent and cannot be formulated similar to the other techniques.

TABLE II
AREA AND DELAY COSTS FOR VARIOUS CED TECHNIQUES

Method	Area	Delay
Dual rail	$i.p+p.o$	$i+p+o$
Duplication	$4.(i.p+.p.o)$	$2.(i+p+o)$
TMR	$3.(i.p+3.p.o+4.o^2)$	$i+3.p+10.o$
5MR	$5.(i.p+5.p.o+12.o^2)$	$i+5.p+21.o$
Parity	N/A	N/A

V. EXPERIMENTAL RESULTS

A. Fault injection and simulation

To evaluate multiple fault coverage of various CED techniques discussed in Section IV, fault injections into different benchmark circuits using a PLA simulator written in C++ programming language have been performed. Our fault simulator program is capable of injecting random multiple faults into multistage nano-crossbars. Please note that simulation of all possible multiple faults is infeasible ($O(3^n)$ for n fault sites). Both crosspoint insertion/deletion and input bridging faults are implemented in the simulator (NDRI faults are modeled by combination of these faults). The pseudo code for each fault injection step is as follows.

Function InjectFault()
  for (all inputs & all crosspoints) **do**
    rnd ← random_number(0,1)
    if (rnd ≤ fault_probability) **then**
      inject fault into the corresponding fault site

In each step of fault injection, we have the option to preserve previous faults and add some new faults to the circuit, or to remove the faults from previous step. This way the simulator resembles the effect of permanent or transient faults, respectively.

Each fault site (crosspoint or input) has a probability of being faulty (*fault rate*) in each step of fault injection by which the defect density in the simulations is controlled. This way all multiple-type multiple faults have the opportunity, with respective probabilities, to occur during fault injection experiments. The probabilities are set so that in each fault injection step there is at least one fault in the circuit, to avoid unnecessary simulation cycles. Also, the probability of bridge between two inputs decreases exponentially with the distance between the inputs.

B. Benchmark implementation

Benchmark circuits used in the experiments adapted from a subset of MCNC benchmarks provided with *RASP logic mapping* toolset. These circuits are in *BLIF format* used by *SIS synthesis* tool. In order to convert the benchmarks to multi-stage PLAs suitable for our experiments, we have used *PLAMAP* from RASP mapping tool. It converts a circuit described in BLIF format to custom-sized PLAs described in the format defined by *ESPRESSO logic minimization* tool. Dual rail multistage benchmarks can then be obtained by combining these PLAs and their dual which have been obtained by ESPRESSO. In order to obtain simulation files for NMR and duplication, we use these dual rail benchmarks and convert them to NMR and duplicate versions, based on the steps provided in Section IV. To obtain parity checking version of the benchmarks, we have modified the BLIF files at the very beginning steps of the above process to incorporate an additional even parity output for the original primary outputs.

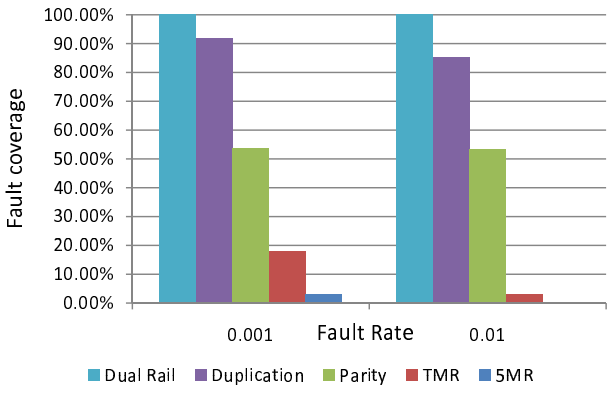

Fig. 4. The effect of fault rate on fault coverage of various CED schemes

C. Simulation results

Table III summarizes the experimental results for different error checking methods. All numbers in the table are rounded to two digits after the decimal point. Information provided in this table contains error detection coverage and fault detection coverage. *Error coverage* refers to the percentage of the errors at the POs which are detected by the corresponding technique. Thus, for TMR and 5MR which are masking techniques, error coverage is meaningless. *Fault coverage* refers to the percentage of faults which are detected by each technique. For NMR techniques, the percentage of faults successfully masked by the technique is considered as the fault coverage since the rest of the faults produces undetected errors. The values in Table III correspond to fault rate of 10^{-3}. This fault rate translates to up to 407 simultaneous multiple faults for alu4 circuit in dual rail scheme (this becomes much higher for other techniques such as 5MR). Along with coverage, area and delay overhead of different techniques are reported as well. The area costs shown in this table are in terms of the number of crosspoints used in each technique. Also, critical path delays are based on the assumption that delay is proportional to the number of crosspoints in both AND-plane and OR-plane from the PIs to POs. Delays and area costs have been normalized to the values for the dual-rail scheme. It can be seen that other techniques have considerable area and performance overhead ranging from 1.87 to 17.14 times the cost of dual rail. As shown in this table, the coverage for dual rail scheme is much more than NMR and parity techniques. Duplication technique has reasonable fault coverage. However, considering area and delay overhead as well, dual-rail checking outperforms.

Figure 4 shows the coverage of these error checking schemes versus different fault rates. This experiment helps to understand the effect of increasing fault rate on the effectiveness of different schemes. As can be seen in this figure, the coverage of NMR techniques drops very quickly with higher fault rate. Also, 5MR coverage is less than TMR. However, the coverage of dual-rail checking, unlike duplication, has almost not been affected with higher fault rates.

VI. Conclusion and Summary

Fault tolerance techniques are essential in the design of nano architectures due to very high permanent and transient failure rates in nanoscale devices. In this paper, we proposed an online multiple error detection scheme, for both permanent and transient faults, based on dual rail implementation of logic functions. This scheme is capable of detecting multiple faults and best suited for nano-crossbar implementations. The proposed scheme eliminates high hardware and performance overhead imposed by input/output checkers in multistage nano-crossbars by incorporating checkers only at the primary outputs. Moreover, in diode-based nano-crossbars which are inherently dual rail, it has virtually no hardware overhead. We proved that the proposed scheme is capable of detecting all single faults as well as most cases of multiple faults. Two alternative implementations were presented to further improve fault coverage of this scheme. Extensive experiments based on random multiple fault injection were performed to compare the effectiveness

TABLE III

SIMULATION RESULTS: FAULT DETECTION COVERAGE, AREA AND PERFORMANCE OVERHEADS OF DIFFERENT CED TECHNIQUES

	Method		duke2	x4	term1	alu1	alu4	rd48	**average**
Error Coverage(%)	Parity		49.05	50.00	44.10	44.09	49.30	52.45	48.17
	Duplication		97.38	99.37	95.32	94.34	76.03	61.85	87.38
	Dual Rail		100.00	100.00	100.00	100.00	100.00	99.96	99.99
Fault Coverage(%)	TMR	Permanent	9.93	1.68	27.60	6.71	12.41	50.15	18.08
		Transient	62.33	41.74	79.43	65.09	61.13	96.44	67.69
	5MR	Permanent	0.02	0.32	0.54	1.06	5.77	8.12	2.64
		Transient	36.43	8.41	18.89	8.06	26.54	75.63	28.99
	Parity	Permanent	49.45	50.74	56.89	59.96	51.80	53.21	53.67
		Transient	59.32	41.68	67.75	93.17	49.63	54.00	60.92
	Duplication	Permanent	97.66	99.39	96.01	94.50	88.06	75.21	91.80
		Transient	99.25	99.87	99.57	98.56	99.80	99.03	99.35
	Dual Rail	Permanent	100.00	100.00	100.00	100.00	100.00	99.98	100.00
		Transient	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Area	TMR		6.66	6.42	6.75	5.93	6.02	4.13	5.98
	5MR		20.33	19.11	20.33	16.94	17.05	9.06	17.14
	Parity		1.42	2.41	1.55	3.32	1.49	1.05	1.87
	Duplication		4.00	4.00	4.00	4.00	4.00	4.00	4.00
	Dual Rail		1.00	1.00	1.00	1.00	1.00	1.00	1.00
delay (critical path)	TMR		3.15	3.33	3.37	3.27	3.13	2.82	3.18
	5MR		5.59	6.00	6.05	5.86	5.50	4.76	5.63
	Parity		5.00	10.33	1.93	2.00	1.35	1.06	3.61
	Duplication		2.00	2.00	2.00	2.00	2.00	2.00	2.00
	Dual Rail		1.00	1.00	1.00	1.00	1.00	1.00	1.00

of this error checking scheme with other online fault detection/masking methods, such as NMR, duplication, and parity checking, in terms of multiple fault coverage as well as area and delay overhead. In average, the error coverage of the proposed scheme for multiple faults is about 99.99% and its fault coverage is 100.00%, compared with 2.64%, 18.08%, 53.67%, and 91.80% for 5MR, TMR, parity checking, and duplication, respectively, when the fault rate is 10^{-3}. These results confirm that dual-rail error checking seems to be the most promising approach for online multiple error detection in crossbar nano-architectures.

REFERENCES

[1] M.M. Ziegler and M.R. Stan. Cmos/nano co-design for crossbar-based molecular electronic systems. *IEEE Trans. on Nanotechnology*, 2(4):217–230, Dec. 2003.

[2] S. C. Goldstein and M. Budiu. Nanofabrics: Spatial computing using molecular electronics. In *Proc. of the annual international symposium on Computer architecture*, pages 178–191, 2001.

[3] M. R. Stan, P. D. Franzon, S. C. Goldstein, J. C. Lach, and M. M. Ziegler. Molecular electronics: From devices and interconnect to circuits and architecture. *Proc. of the IEEE*, 91:1940–1957, 2003.

[4] D. B. Strukov and K. K. Likharev. A reconfigurable architecture for hybrid cmos/nanodevice circuits. In *Proc. of the ACM/SIGDA international symposium on Field programmable gate arrays (FPGA)*, pages 131–140, New York, NY, USA, 2006. ACM.

[5] A. DeHon. Array-based architecture for fet-based, nanoscale electronics. *IEEE Trans. on Nanotechnology*, 2(1):23–32, Mar 2003.

[6] W. Rao, A. Orailoglu, and R. Karri. Logic level fault tolerance approaches targeting nanoelectronics plas. In *Design, Automation and Test in Europe Conf. and Exhibition(DATE)*, pages 1–5, April 2007.

[7] I. Polian and W. Rao. Selective hardening of nanopla circuits. In *Proc. of IEEE Int'l Symp. on Defect and Fault Tolerance of VLSI Systems(DFT)*, pages 263–271, Washington, DC, USA, 2008. IEEE Computer Society.

[8] W. Rao, A. Orailoglu, and R. Karri. Nanofabric topologies and reconfiguration algorithms to support dynamically adaptive fault tolerance. In *in Proc. of IEEE VLSI Test Symp. (VTS)*, pages 214–221, April-4 May 2006.

[9] S. J. Upadhyaya and K. K. Saluja. A new approach to the design of built-in self-testing plas for high fault coverage. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 7(1):60–67, Jan 1988.

[10] B. D. Liu and J. J. Sheu. A new low overhead design for testability of programmable logic arrays. In *IEEE Int'l Sympoisum on Circuits and Systems(ISCAS)*, pages 1972–1975 vol.4, Jun 1991.

[11] Y. Min and J. Li. Strongly fault secure plas and totally self-checking checkers. *IEEE Trans. on Computers*, 37(7):863–867, Jul 1988.

[12] J. Khakbaz and E.J. McCluskey. Concurrent error detection and testing for large pla's. *IEEE Trans. on Electron Devices*, 29(4):756–764, Apr 1982.

[13] W.K. Fuchs, C.-Y.R. Chen, and J.A. Abraham. Concurrent error detection in highly structured logic arrays. *IEEE Journal of Solid-State Circuits*, 22(4):583–594, Aug 1987.

[14] W. Rao, A. Orailoglu, and R. Karri. Fault tolerant approaches to nanoelectronic programmable logic arrays. In *IEEE/IFIP Int'l Conf. on Dependable Systems and Networks(DSN)*, pages 216–224, June 2007.

[15] Y. Huang, X. Duan, Y. Cui, L. J. Lauhon, K. Kim, and C. M. Lieber. Logic gates and computation from assembled nanowire building blocks. *Science*, 294:1313 – 1317, 2001.

[16] A. Bachtold, P. Hadley, T. Nakanishi, and C. Dekker. Logic circuits with carbon nanotube transistors. *Science*, 294, 2001.

[17] Y. Cui and C. M. Lieber. Functional nanoscale electronic devices assembled using silicon nanowire building blocks. *Science*, 291:851–853, 2001.

[18] R. S. Wei and A. Sangiovanni-Vincentelli. Platypus: A pla test pattern generation tool. In *Design Automation Conf.(DAC)*, pages 197–203, June 1985.

[19] F. Somenzi and S. Gai. Fault detection in programmable logic arrays. *Proc. of the IEEE*, 74(5):655–668, May 1986.

[20] V.K. Agarwal. Multiple fault detection in programmable logic arrays. *IEEE Trans. on Computers*, C-29(6):518–522, June 1980.

[21] D. A. Anderson and G. Metze. Design of totally self-checking check circuits for m-out-of-n codes. *IEEE Trans. on Computers*, C-22(3):263–269, March 1973.

[22] M. B. Tahoori. Application-independent defect-tolerant crossbar nano-architectures. In *IEEE/ACM Int'l Conf. on Computer-Aided Design(ICCAD)*, pages 730–734, Nov. 2006.

[23] U.K. Bhattacharyya, I. Sen Gupta, S. Shyama Nath, and P. Dutta. Pla based synthesis and testing of hazard free logic. In *VLSI Design, 1995., Proceedings of the 8th International Conference on*, pages 121–124, Jan 1995.