

# Selective Hardening of NanoPLA Circuits

Ilia Polian

*Computer Architecture Group  
Institute for Computer Science  
Albert-Ludwigs-University  
Georges-Köhler-Allee 51  
D-79110 Freiburg i. Br., Germany  
polian@informatik.uni-freiburg.de*

Wenjing Rao

*ECE Department  
University of Illinois at Chicago  
851 S. Morgan St., MC 154,  
1020 SEO building  
Chicago, IL, USA  
wenjing@ece.uic.edu*

## Abstract

*Nanoelectronic components are expected to suffer from very high error rates, implying the need for hardening techniques. We propose a fine-grained approach to harden a promising class of nanoelectronic circuits, called NanoPLAs, against errors. An analytical procedure and simulations are both incorporated into the algorithm to identify the most critical error locations. By targeting errors with the largest impact for a given circuit, the method can provide significant reliability boost at low cost. Furthermore, the method yields a plethora of alternative designs, trading off hardening costs against circuit robustness. In many cases, solutions found achieve both lower cost and higher robustness compared with the duplication-based hardening strategy introduced before.*

**Keywords:** NanoPLA circuits, robust design, selective hardening, fault tolerance

## 1: Introduction

Next-generation electronic circuits based on nanotechnology are commonly expected to outperform today's CMOS microelectronics with respect to integration density (area), power consumption, and computation speed [1, 2]. The flip side of the coin is the increased susceptibility of these circuits to errors. Systems employing nano components will presumably have to deal with non-negligible error rates [2, 3, 4, 5]. The error effects can be corrected using techniques such as hardware / time redundancy, error-correcting information encoding, software-based fault tolerance, or combinations thereof [6, 7, 8, 9, 10, 11, 12].

The optimal error correction strategy strongly depends on the expected error rates. For instance, commit-rollback recovery is only effective when error rates do not exceed a certain threshold. On the other hand, hardware redundancy imposes considerable costs. This is particularly true for nano components in which the integration of a large number of basic elements into a working system often becomes a challenge due to strict constraint of localized interconnections. Moreover, even massively redundant design techniques such as triple-modular redundancy fail to provide the flexibility to deal with the high and variable fault rates. Under such a severe reliability challenge, employing even more redundancy aiming at the worst case scenario will increase the cost to a level at which the advantage over CMOS becomes questionable. As a consequence, the optimal strategy should probably combine some degree of hardware-level hardening to tune the error rate of the component to a desired value and a system-level error correction mechanism which works sufficiently well for that error rate.

In this paper, we propose a strategy to selectively harden nanoelectronic circuits belonging to the class of NanoPLAs. Extensive research work has been carried out for NanoPLAs

in [13, 14, 15, 16, 17]. A NanoPLA logic is built on a crossbar architecture, consisting of two sets of perpendicular nanowires. Self-assembly based fabrication can be used to place a nanoelectronic device at each crosspoint, with two distinct states of connecting and disconnecting the two wires. The reconfigurability in nanoelectronic devices enables a nano crossbar to implement arbitrary functions in a two-level PLA logic form.

Reliability concern regarding CMOS PLAs, including online fault detection and diagnosis, has been existing in the literature for a long time [18, 19, 20, 21, 22, 23]. Focusing on the fault tolerance of NanoPLAs, hardening techniques have been proposed in [24, 12]. These techniques are associated with rather high cost and target at all the possible errors unanimously. The proposed strategy hardens a NanoPLA circuit at a per-defect based fine-grain level. The strategy pinpointedly determines the elements in the circuit with the largest contribution to the error rate and hardens them with higher priorities. This allows to achieve the desired robustness at minimized cost.

The algorithm to select the elements to harden optimizes a cost function which combines the two criteria: robustness improvement and hardening cost. A limited number of candidate locations are shortlisted. The robustness improvement and the area cost of hardening are determined for shortlisted candidates, and the candidates with the best overall outcome is selected. Experiment on NanoPLAs constructed from benchmark circuits confirm that focusing on most critical error locations leads to a spectrum of designs, which allow the designer to select the cost-minimal alternative which satisfies the robustness requirement given by the system-level considerations.

## 2: Hardening Techniques for NanoPLA Circuits

### 2.1: NanoPLA circuits

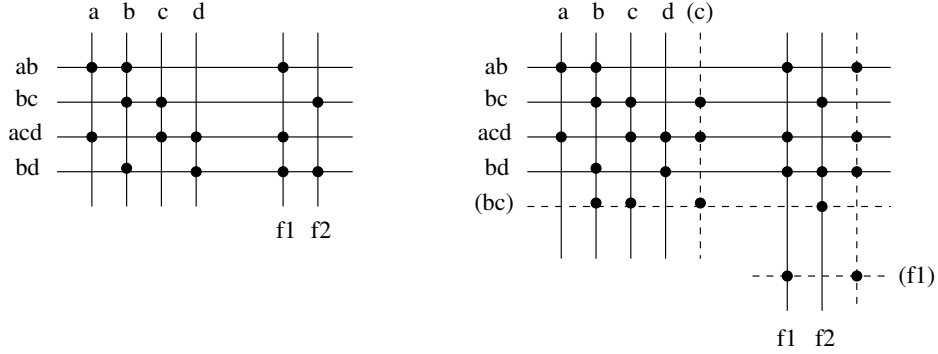
NanoPLA circuits have structure similar to conventional Programmable Logic Arrays (PLA). An (unhardened) NanoPLA circuit has  $n$  *input wires*  $i_1, \dots, i_n$ ,  $l$  *output wires*  $o_1, \dots, o_l$ , and  $m$  *row wires*  $r_1, \dots, r_m$  used for product terms. The input wires are connected with row wires by the *AND plane* formally described by an  $m \times n$  matrix  $\mathcal{A}$  with  $\mathcal{A}_{jk} = 1$  if input wire  $i_k$  is connected with row wire  $j$  and  $\mathcal{A}_{jk} = 0$  otherwise. The output wires are connected with row wires by the *OR plane* described by an  $m \times l$  matrix  $\mathcal{O}$  with  $\mathcal{O}_{jk} = 1$  if output wire  $o_k$  is connected with row wire  $j$  and  $\mathcal{O}_{jk} = 0$  otherwise.

When a Boolean vector  $(v_1, \dots, v_n)$  is applied to the input wires, the row wires and the output wires assume Boolean values given by the mapping  $val$ . The value  $val(r_j)$  of row wire  $r_j$  is the conjunction of values on the input wires connected with row wire  $r_j$ , while the value  $val(o_k)$  of output wire  $o_k$  is the disjunction of values on the row wires connected with output wire:

$$val(r_j) = \bigwedge_{\mathcal{A}_{jk}=1} v_k \quad val(o_k) = \bigvee_{\mathcal{O}_{jk}=1} val(r_j). \quad (1)$$

Obviously, NanoPLAs implement two-level sum-of-product functions. In principle, any arbitrary Boolean function can be transformed into the two-level sum-of-product form.

While PLAs implemented in CMOS are easily integrated into a larger CMOS circuit, embedding NanoPLAs into CMOS circuitry is expected to require special nano-CMOS interface modules [15, 16, 17]. We assume that such a module is present on each input and each output of the (unhardened) NanoPLA. In this way, input values for the NanoPLA are reliably generated, and the output values are reliably read out, using surrounding CMOS electronics.



**Figure 1. An example of hardening NanoPLAs**

## 2.2: Error model

We focus on the main types of errors due to the fabrication mechanism of NanoPLAs. The behavior of devices sandwiched between the two sets of nanowires will be heavily impacted by the process variation in a self-assembly based fabrication. In addition, the “on” and “off” states of each device can be influenced dynamically by the environmental effects and background noise. At the logic level, existing connections between two wires can disappear, and absent connections can appear. Consequently, based on the error direction and location, four error modes can be distinguished: appearance error in the AND plane, disappearance error in the AND plane, appearance error in the OR plane, and disappearance error in the OR plane. We denote an error leading to appearance of a new connection at the  $j$ 's row and  $k$ 's column in the AND plane as  $\mathcal{A}_{jk}^\uparrow$ . An error leading to disappearance of an existing connection of such a place is denoted as  $\mathcal{A}_{jk}^\downarrow$ . Similarly, appearance / disappearance errors at the  $j$ 's row and  $k$ 's column in the OR plane are written as  $\mathcal{O}_{jk}^\uparrow$  and  $\mathcal{O}_{jk}^\downarrow$ , respectively. Obviously, an appearance (disappearance) error is only defined for connected (unconnected) pairs of wires.

When considering errors in conventional technologies such as CMOS, the error rate is often assumed to be so low that maximally one error can be present in the circuit. This assumption is overly restrictive for NanoPLA circuits for which high error rates are expected. In this paper, we assume errors on individual locations (pairs of wires) to be independent stochastic processes. We define four probabilities:  $p(\mathcal{A}^\uparrow)$ ,  $p(\mathcal{A}^\downarrow)$ ,  $p(\mathcal{O}^\uparrow)$  and  $p(\mathcal{O}^\downarrow)$ . If wires  $i_k$  and  $r_j$  are connected (i.e., the device at the position  $(j, k)$  is turned on:  $\mathcal{A}_{jk} = 1$ ), then an error makes such a connection disappear (i.e., error  $\mathcal{A}_{jk}^\downarrow$  may occur) with probability  $p(\mathcal{A}^\downarrow)$ . Probability  $p(\mathcal{A}^\uparrow)$  describes the likelihood of unconnected wires in the AND plane to erroneously become connected.  $p(\mathcal{O}^\downarrow)$  and  $p(\mathcal{O}^\uparrow)$  are the respective probabilities for the OR plane.

When simulating an input vector under the employed error model, errors are injected at all locations simultaneously with the probabilities mentioned above. This means that no errors, one error or several errors may be injected. This methodology is essential for accurate simulation of selectively hardened NanoPLAs. As described below, hardening a NanoPLA against an error introduces new hardware which is also error-prone. We assume that errors occur on the redundant hardware with the same probability as on the original hardware.

### 2.3: Hardening techniques

We next introduce techniques to pinpointedly harden a NanoPLA against an error based on the error model used:  $\mathcal{A}_{jk}^\uparrow$ ,  $\mathcal{A}_{jk}^\downarrow$ ,  $\mathcal{O}_{jk}^\uparrow$ , and  $\mathcal{O}_{jk}^\downarrow$ . Different from the traditional fault tolerance schemes such as TMR, the hardening technique is based on Boolean tautology. Applying such a strategy eliminates the voting stage in TMR which imposes significant performance and area overhead in a two-level logic structure.

A hardening process deals with a particular error at a time. When the NanoPLA is hardened against error  $\mathcal{A}_{jk}^\uparrow$ , the occurrence of such an error will not generate any erroneous effect at the output for all the inputs. Comparing to the original area cost of the unhardened NanoPLA as  $m \cdot (n + l)$ , we also derive the area cost of hardening using the proposed techniques.

Hardening the NanoPLA against a **disappearance error in the AND plane**  $\mathcal{A}_{jk}^\downarrow$  is done by introducing a new input wire  $i_{k'}$ , connected with the same input signal and the same row wires as  $i_k$ :  $\mathcal{A}_{jk'} := \mathcal{A}_{jk}$ . This hardening step also immunizes the NanoPLA against all disappearance errors on input wire  $i_k$ .

Hardening the NanoPLA against an **appearance error in the AND plane**  $\mathcal{A}_{jk}^\uparrow$  is done by adding a new row wire  $r_{j'}$  connected with the same input and output wires as row wire  $r_j$ :  $\mathcal{A}_{j'k} := \mathcal{A}_{jk} \forall k \in \{1, \dots, n\}$ ,  $\mathcal{O}_{j'k} := \mathcal{O}_{jk} \forall k \in \{1, \dots, l\}$ . The hardened NanoPLA is also immune against all other appearance errors on row wires  $r_j$  and  $r_{j'}$  in the AND plane. Moreover, it is immune against **disappearance errors on that rows in the OR plane**.

Finally, to harden the NanoPLA against an **appearance error in the OR plane**,  $\mathcal{O}_{js}^\uparrow$ , a new output wire  $o_{s'}$  duplicating all the connections on wire  $o_s$  is introduced, with  $\mathcal{O}_{js'} = \mathcal{O}_{js}$ . The output wires  $o_s$  and  $o_{s'}$  are ANDed together, as the result of output  $s$ . There are two methods to implement this AND operation. First, a new AND plane could be introduced and the AND operation could be implemented using the same technology as in the first AND plane of the NanoPLA. The disadvantage of this solution is the susceptibility of the new AND plane to appearance errors (disappearance errors are protected by the logic in the OR plane). The second solution is to use nano-CMOS interfacing on output wires  $o_s$  and  $o_{s'}$  and adding a CMOS AND gate to connect that wires. In this paper, we choose the second solution and assume that the CMOS and gate is not susceptible to errors.

The *area cost* of hardening the NanoPLA against an error is calculated as the increase in its area due to hardening. For hardening the NanoPLA against an  $\mathcal{A}_{jk}^\uparrow$  or  $\mathcal{O}_{jk}^\downarrow$  error, the area overhead is the length of the added wire, or  $n + l$ . For hardening against an  $\mathcal{A}_{jk}^\downarrow$  error, the new input wire adds the area cost of  $m$ . To harden a NanoPLA against an  $\mathcal{O}_{jk}^\uparrow$ , one new output wire of length  $m$  must be complemented by a CMOS gate. Assuming that one CMOS transistor occupies ten times an area of a NanoPLA switch, a two-input CMOS gate amounts to an extra area cost of 40. Clearly, the NanoPLA sizes  $n$ ,  $m$  and  $l$  are increased by hardening which results in a higher area cost of subsequent hardening steps. It is possible to harden the NanoPLA multiple times against the same error, reducing its susceptibility to multiple errors.

Applying the hardening operations for every input wire, row wire and output wire results in a NanoPLA *completely protected against all modeled single errors*. This is similar to the AOA architecture from [24, 12], except that the AND operation with newly introduced output wires was implemented as an AND plane and not in CMOS. The area cost of the NanoPLA thus hardened is  $4 \cdot m \cdot (n + l) + 40 \cdot l$ . Figure 1 shows an example of partially hardening a NanoPLA circuit. The left side shows the original circuit, while the right side shows the hardened circuit with duplicated variable  $c$ , product term  $bc$  and output  $f1$ .

**Procedure calculate\_OER**  
**Input:** NanoPLA  $C$ , number of vectors  $S$ , convergence bound  $\varepsilon$   
**Output:** Observed error rate OER  
**int** iterations = 0, no\_errors = 0;  
**double** OER =  $-\infty$ , OER\_old;  
**input\_vector** B;  
(1) **do**  
(2)   no\_iterations := no\_iterations + 1;  
(3)   OER\_old := OER;  
(4)   **for** i := 1 **to**  $S$  **do**  
(5)     B := random\_input\_vector();  
(6)     **if** (good\_sim( $C$ , B)  $\neq$  error\_sim( $C$ , B))  
(7)       **then** no\_errors := no\_errors + 1;  
(8)     **end for**  
(9)   OER := no\_errors / ( $S \cdot$  no\_iterations);  
(10) **while** ( $|\text{OER} - \text{OER\_old}| > \varepsilon$ );  
(11) **return** OER;  
**end** calculate\_OER;

(a)

**Procedure harden**  
**Input:** NanoPLA  $C$ , area bound ref\_area  
**Output:** Hardened NanoPLA  $\hat{C}$   
**error\_array** error\_list, short\_list;  
**NanoPLA**  $\hat{C}$ ,  $\tilde{C}$ ;  
(1)  $\hat{C} := C$ ;  
(2) Create error\_list; mark all errors unprotected;  
(3) Calculate det. probabilities of all errors;  
(4) Sort error\_list according to det. probability;  
(5) **while** (area( $\hat{C}$ ) < ref\_area) **do begin**  
(6)   short\_list := 5 unhardened errors with  
   largest det. prob. + 5 random errors  
(7)   **for each** error  $e$  from short\_list **do**  
(8)      $\tilde{C} := C$  hardened against error  $e$ ;  
(9)     cost( $e$ ) := area( $\tilde{C}$ )  $\cdot$  calculate\_OER( $\tilde{C}$ );  
(10)    **if** ( $e$  marked protected)  
   **then** cost( $e$ ) := cost( $e$ )  $\cdot$  1.5;  
(11)   **end for**  
(12)    $e_{opt} := e$  with minimal cost;  
(13)    $\hat{C} := \hat{C}$  hardened against error  $e_{opt}$ ;  
(14)   Output area( $\hat{C}$ ), calculate\_OER( $\hat{C}$ );  
(15)   Mark protected errors in error\_list;  
(16) **end while**  
(17) **return**  $\hat{C}$ ;  
**end** harden;

(b)

**Figure 2. Algorithm to estimate observed error rate (a) and to harden a NanoPLA (b)**

### 3: Selective Hardening Algorithm

#### 3.1: Estimation of Observed Error Rate (OER)

Accurate estimation of the error rate under error model from Section 2.2 is essential for guiding the hardening process of the partially hardened NanoPLA. When  $S$  input vectors are applied to a NanoPLA with errors injected according to probabilities  $p(\mathcal{A}^\downarrow)$ ,  $p(\mathcal{A}^\uparrow)$ ,  $p(\mathcal{O}^\uparrow)$  and  $p(\mathcal{O}^\downarrow)$ , the *Observed Error Rate* (OER) is defined as the number of input vectors producing erroneous output divided by  $S$ . OER is therefore a metric for a NanoPLA's robustness.

It is possible to determine OER by simply simulating  $S$  random input vectors, thus value of OER depending on both the random vectors and the errors injected. In general, the accuracy of estimation increases with  $S$ . To bound the inaccuracy in OER estimation without resorting to computationally prohibitive values of  $S$ , we employ the adaptive procedure outlined in Figure 2 (a). The number of vectors simulated is iteratively increased by  $S$  until the difference between two last values of OER falls below  $\varepsilon$ . Larger values of  $S$  and smaller values of  $\varepsilon$  yield more accurate result with longer simulation time, and vice versa.

#### 3.2: Hardening

Selective hardening of a NanoPLA is done using the procedure from Figure 2 (b). All modeled errors are organized in an error list. We use the *Detection Probability* ( $DP$ ) of an error as a quick estimate of its contribution to circuit-wide OER. The calculation of this number is based on *expected signal probabilities* on row wires. Assuming equiprobable

input patterns, the expected signal probability on row wire  $r_j$  is

$$ESP(r_j) = 1/2^{|\{t \in \{1, \dots, m\} \mid \mathcal{A}_{jt}=1\}|}. \quad (2)$$

For an error in the AND plane ( $\mathcal{A}_{jk}^\uparrow$  and  $\mathcal{A}_{jk}^\downarrow$ ) to be stimulated, all input wires connected to row wire  $r_j$  except input wire  $i_k$  must be set to logic-1. Furthermore, for such an error to manifest at an one output wire connected with  $r_j$ , all other row wires connected with it should carry logic-0. The detection probability thus is

$$DP(\mathcal{A}_{jk}^\uparrow) = DP(\mathcal{A}_{jk}^\downarrow) = \frac{1 - \left(1 - \prod_{\mathcal{O}_{jv}=1} \left( \prod_{\mathcal{O}_{wv}=1, w \neq j} (1 - ESP(r_w)) \right)\right)}{2^{|\{t \in \{1, \dots, k-1, k+1, \dots, m\} \mid \mathcal{A}_{jt}=1\}|}}. \quad (3)$$

The denominator accounts for error propagation to the row wire. The numerator is the probability that the propagation is blocked in the OR plane for every connected output wire, subtracted from one.

For errors  $\mathcal{O}_{jk}^\uparrow$  and  $\mathcal{O}_{jk}^\downarrow$ , the detection probabilities are

$$DP(\mathcal{O}_{jk}^\uparrow) = DP(\mathcal{O}_{jk}^\downarrow) = \prod_{\mathcal{O}_{wk}=1, w \neq j} (1 - ESP(r_w)). \quad (4)$$

The error list is sorted by the detection probability. Furthermore, each error is assigned a *protection status* which indicates if it has been protected by any of the hardening operations done so far. The initial protection status is 0 for all errors.

The NanoPLA is hardened iteratively: one error per iteration is selected (lines 6–12) and the NanoPLA is hardened against this error (lines 13–15). This process is terminated when the area cost of the NanoPLA exceeds a user-defined bound (Line 5), although other criteria are possible. The selection is done in two steps: first, a *shortlist* of errors is generated by taking 5 unprotected errors with the largest single-error detection probability from the error list accompanied by 5 random errors, protected or not (Line 6).

For each of the errors in the shortlist, we calculate the area cost which a NanoPLA hardened against that error would have. We also calculate the OER of that NanoPLA using Procedure `calculate_OER` (Lines 8–9). We compute the *cost function* as the product of these two numbers and finally select error  $e_{opt}$  with the lowest cost function (Line 12). This compensates for potential inaccuracy of single-error detection probability as a metric. In addition, we bias the algorithm towards selecting errors not yet protected (Line 10).

The NanoPLA is then hardened against error  $e_{opt}$  and the protection status of newly protected errors is updated and the area cost and the OER of the solution found so far are reported. It is important to point out that we repeat the measurement of the OER in Line 14 rather than simply using the value computed in Line 9. This is done to handle the situations in which the OER estimated by procedure `calculate_OER` happens to be much lower than the actual OER due to fortunate selection of input vectors and error locations. These situations cannot be ruled out if procedure `calculate_OER` is called frequently. By repeating the calculation, the likelihood of reporting overly optimistic numbers is greatly reduced.

## 4: Experimental Results

We derived benchmark NanoPLAs from LGSynth93 by setting all input and output don't cares in these circuits to 0 and keeping the structure of the circuit unchanged. Based on this setup, we ran the proposed algorithm (procedure `harden` from Figure 2 (b)). For reference,

NanoPLA	$n$	$l$	$m$	Unhardened		Complete		First lower OER			First larger cost		
				cost	OER	cost	OER	iter	cost	OER	iter	cost	OER
9sym	9	1	87	870	0.002186	3520	0.000000	–	–	–	71	3575	0.001230
alu4	14	8	1028	22616	0.005095	90784	0.000000	–	–	–	493	90842	0.001250
b12	15	9	431	10344	0.010636	41736	0.000625	12	14020	0.000620	457	42567	0.000000
bw	5	28	87	2871	0.047290	12604	0.001664	99	10764	0.001244	120	12659	0.001025
clip	9	5	167	2338	0.013896	9552	0.000300	–	–	–	108	9584	0.003176
con1	7	2	9	81	0.040658	404	0.005633	9	202	0.004845	20	420	0.001850
duke2	22	29	87	4437	0.045856	18908	0.000500	129	17650	0.000500	141	18910	0.000656
e64	65	65	65	8450	0.113185	36400	0.002200	194	34970	0.001844	202	36492	0.001700
ex5p	8	63	256	18176	0.028365	75224	0.000280	79	33568	0.000243	329	75395	0.000275
inc	7	9	34	544	0.048000	2536	0.001883	44	2113	0.001662	53	2566	0.001442
misex1	8	7	32	480	0.042828	2200	0.002167	37	1679	0.001575	52	2288	0.001255
misex2	25	18	29	1247	0.119100	5708	0.004200	57	4448	0.004055	75	5752	0.002989
misex3c	14	14	305	8540	0.020464	34720	0.000520	–	–	–	226	35148	0.000964
rd53	5	3	32	256	0.027029	1144	0.001200	34	988	0.000400	39	1178	0.000683
rd73	7	3	141	1410	0.004800	5760	0.000100	92	5722	0.000080	94	5774	0.000025
rd84	8	4	256	3072	0.010172	12448	0.000300	–	–	–	102	12501	0.001224
sao2	10	4	58	812	0.014083	3408	0.000667	9	1174	0.000471	75	3522	0.000471
sqrt8	8	4	40	480	0.022337	2080	0.001100	37	1778	0.000977	44	2103	0.000807
squar5	5	8	32	416	0.078462	1984	0.004381	39	1668	0.004014	49	2048	0.002746
vg2	25	8	110	3630	0.015758	14840	0.000300	81	10060	0.000257	143	14893	0.000300
xor5	5	1	16	96	0.010620	424	0.001500	14	288	0.001489	24	429	0.001450

**Table 1. Cost and observed error rate of unhardened, completely hardened, and selectively hardened NanoPLAs**

we estimated the observed error rate (using Procedure `calculate_OER` from Figure 2 (b)) for the original NanoPLA and the NanoPLA completely hardened against single errors as discussed in the end of Section 2.3. We used the cost of the completely hardened NanoPLA as the cost limit for Procedure `harden`.

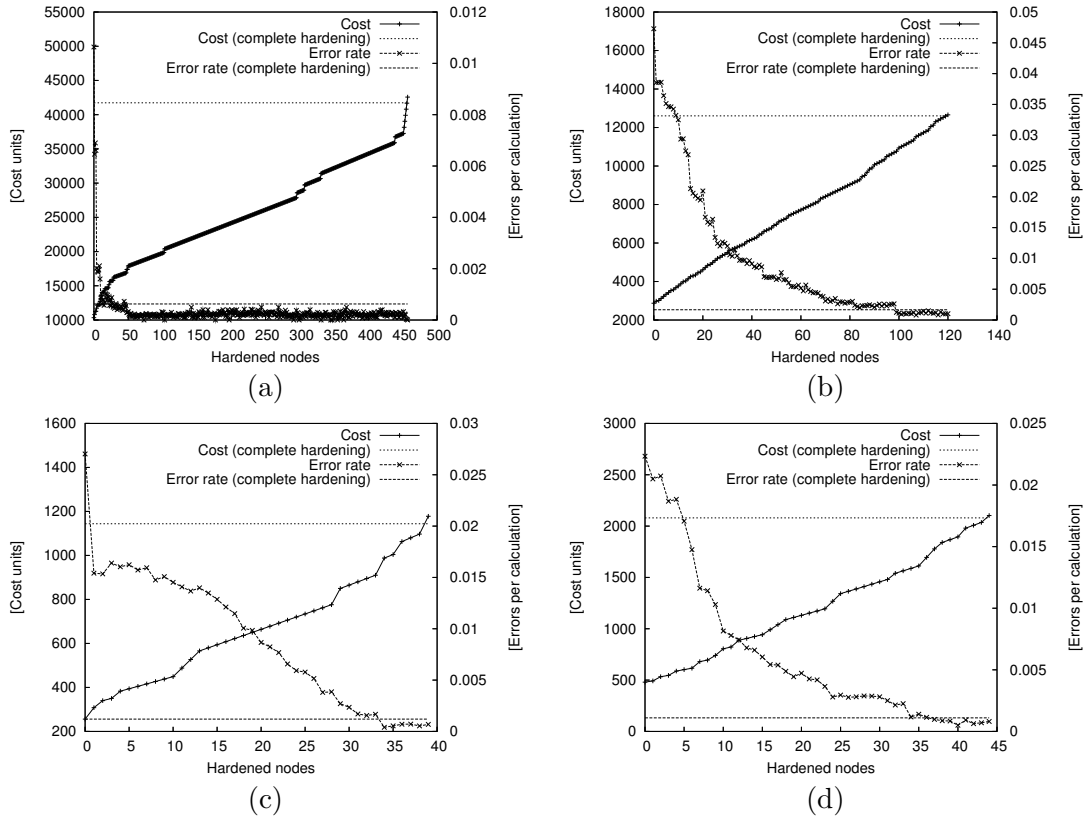
We set error probabilities  $p(\mathcal{A}^\uparrow)$ ,  $p(\mathcal{A}^\downarrow)$ ,  $p(\mathcal{O}^\uparrow)$  and  $p(\mathcal{O}^\downarrow)$  to identical value of  $1/(m \cdot (n + l))$ . This implies that the expected number of errors injected per input vector is one. We employed this very aggressive error injection rate to study the performance of our algorithm for a technology which might be classified as totally unreliable.

We set parameters  $S$  and  $\varepsilon$  of Procedure `calculate_OER` to  $10^4$  and  $10^{-5}$ , respectively, when evaluating the actual OER of the found (intermediate) solution (Line 14 of procedure `harden`) and estimating the OER of unhardened and completely hardened NanoPLA. When grading the OER of shortlisted hardening candidates (Line 9 of Procedure `harden`), we use  $S = 10^3$  and  $\varepsilon = 10^{-4}$ . This yields less accurate results but saves computational effort.

Table 1 summarizes the results. Columns 1 through 4 contain the NanoPLA name, its number of input wires, row wires and output wires. Columns 5 through 8 quote the cost and the OER of the unhardened NanoPLA and the NanoPLA completely hardened against all single errors. Note that the latter OER is typically not 0 because multiple errors occur.

While Procedure `harden` yields a plethora of solutions (NanoPLAs with different degrees of hardening), we report two of the data points. The first solution with an OER below that of the completely hardened NanoPLA is given in Columns 9 through 11 (circuits for which no lower OER was achieved are marked by ‘–’). The first solution which exceeded the cost of the completely hardened NanoPLA is quoted in Columns 12 through 14. For both solutions, the iteration in which they have been obtained, their cost and their OER are reported. The complete solution space produced by Procedure `harden` is shown in Figure 3 in graph form. The cost and the OER of the NanoPLA obtained after a number of iterations are plotted, cost and OER of the completely hardened NanoPLA are shown as horizontal lines for reference.

Both the cost and the robustness are monotonic in the number of iterations of Procedure `harden`. While monotonic increase can be observed for cost in Figure 3, the OER (which should be tracking the robustness) is not decreasing monotonically in all cases. This is because the OER is measured using random input vectors and randomly injected errors. If the NanoPLA has been protected against an error, this error may never be injected during



**Figure 3. All (intermediate) solutions found by Procedure harden for NanoPLAs b12 (a), bw (b), rd53 (c) and sqrt8 (d)**

the next invocation of Procedure `calculate_OER`, while errors against which the NanoPLA is unprotected are injected instead. As a consequence, the OER may actually increase.

While it appears advisable to set up some sort of a “reference error injection experiment” using the same input vectors and error injection sites to avoid the counter-intuitive OER increase, it is not possible to perform such an experiment in a meaningful way. The error injection sites cannot stay constant because new hardware is added to the NanoPLA and errors must be injected on locations not present in previous iterations. A fixed set of input vectors cannot be used because Procedure `calculate_OER` terminates on convergence which happens after a different number of vectors for different simulations. Last but not least, having reference parameters would bias the experiment towards hardening the NanoPLA against the errors most critical for these parameters rather than on average. Hence, we are forced to accept the OER increase due to measurement inaccuracy.

From Table 1, the proposed procedure quite often produces solutions with both cost and OER below the completely hardened NanoPLA. This must not be over-emphasized due to the above-mentioned measurement inaccuracy, yet it demonstrates that it is possible to outperform the complete hardening by directed search. The solution with equivalent cost typically has lower OER than the completely hardened NanoPLA. However, the main feature of the method is the large range of alternative NanoPLA it provides. This allows the system integrator to pick the implementation with maximal error rate the application can tolerate (or resort to a CMOS implementation if such an error rate cannot be achieved). Given the measurement inaccuracy, it appears advisable to validate the found solution by a more thorough simulation or to use a safety margin.

## 5: Conclusions

We demonstrated the ability of selective hardening to perform fine-grained control of robustness in NanoPLA circuit minimizing the area cost. The proposed method is based on a fully-automated sequence of systematic design transformations. The approach is transparent, i.e., it does not change the circuit's functionality and does not require extra control functionality such as commit-rollback architectures. Hence, the process does not require any specific information on the design and can be done by the system integrator with no need to interact with the designer.

Our future research will focus on systems in which certain failing patterns are acceptable. We will identify error behavior which is non-critical with respect to the application and consider only critical error behavior as optimization target. Another promising research direction is the application of our approach to other classes of nanoarchitectures.

## 6 References

- [1] European Commission. *Technology Roadmap for Nanoelectronics*, 2001.
- [2] ITRS. *International Technology Roadmap for Semiconductors Emerging Research Devices*, 2006.
- [3] P. Beckett and A. Jennings. Towards nanocomputer architecture. In *Asia-Pacific Computer System Architecture Conference*, pages 141–150, 2002.
- [4] M. Forshaw, R. Stadler, D. Crawley, and K. Nikolic. A short review of nanoelectronic architectures. In *Nanotechnology*, volume 15, pages 220–223, 2004.
- [5] R. I. Bahar, D. Hammerstrom, J. Harlow, W. H. Joyner, C. Lau, D. Marculescu, A. Orailoglu, and M. Pedram. Architectures for silicon nanoelectronics and beyond. *IEEE Computer*, 40(1):25–33, January 2007.
- [6] J. H. Patel and L. Y. Fung. Concurrent error detection in alus by recomputing with shifted operands. *IEEE Transactions on Computers*, 31:589–592, December 1982.
- [7] K. Nikolic, A. Sadek, and M. Forshaw. Fault-tolerant techniques for nanocomputers. In *Nanotechnology*, pages 357–362, 2002.
- [8] W. Rao, R. Karri, and A. Orailoglu. Fault tolerant arithmetic with applications in nanotechnology based systems. In *ITC*, pages 472–478, 2004.
- [9] Y. Qi, J. Gao, and J. A. B. Fortes. Markov chains and probabilistic computation - a general framework for multiplexed nanoelectronic systems. *IEEE Transactions on Nanotechnology*, 4(2):194–205, March 2005.
- [10] J. Han, J. Gao, Y. Qi, P. Jonker, and J. A. B. Fortes. Toward hardware-redundant, fault-tolerant logic for nanoelectronics. *IEEE Design and Test of Computers*, 22(4):328–339, July-August 2005.
- [11] W. Rao, A. Orailoglu, and R. Karri. Towards nanoelectronics processor architectures. *JETTA Special Issue on Test, Defect Tolerance, and Reliability of Nanoscale Devices*, 23:235–254, 2007.
- [12] W. Rao, A. Orailoglu, and R. Karri. Fault tolerance approaches to nanoelectronic programmable logic arrays. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 216–224, 2007.
- [13] P. J. Kuekes, D. R. Stewart, and R. S. Williams. The crossbar latch: Logic value storage, restoration, and inversion in crossbar circuits. *Journal of Applied Physics*, 97(3):034301, July 2005.
- [14] G. Snider, P. J. Kuekes, and R. S. Williams. Cmos-like logic in defective, nanoscale crossbars. *Nanotechnology*, 15:881–891, Aug 2004.
- [15] A. DeHon and M. J. Wilson. Nanowire-based sublithographic programmable logic arrays. In *FPGA*, pages 123–132, 2004.
- [16] A. DeHon. Array-based architecture for fet-based, nanoscale electronics. *IEEE Transactions on Nanotechnology*, 2(1):23–32, 2003.
- [17] D. B. Strukov and K. K. Likharev. Cmol fpga: A reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices. *Nanotechnology*, 16:888–900, Apr 2005.
- [18] V. K. Agarwal. Multiple fault detection in programmable logic arrays. *IEEE Transactions on Computers*, 29:518–522, June 1980.
- [19] J. Khakbaz and E. J. McCluskey. Concurrent error detection and testing for large pla's. *IEEE Journal of Solid-State Circuits*, 17(2):386–394, April 1982.
- [20] W. K. Fuchs, C. R. Chen, and J. A. Abraham. Concurrent error detection in highly structured logic arrays. *IEEE Journal of Solid-State Circuits*, 22(4):583–594, August 1987.
- [21] T-Y. Chang and C-L. Wey. Design of fault diagnosable and repairable pla's. *IEEE Journal of Solid-State Circuits*, 24(5):1451–1454, October 1989.
- [22] P. K. Lala and D. L. Tao. On fault-tolerant pla design. In *IEEE Southeastcon*, pages 945–947, 1990.
- [23] M. Demjanenko and S. J. Upadhyaya. Yield enhancement of field programmable logic arrays by inherent component redundancy. *TCAD*, 9(8):876–884, 1990.
- [24] W. Rao, A. Orailoglu, and R. Karri. Logic level fault tolerance approaches targeting nanoelectronic plas. In *IEEE Design, Automation, and Test in Europe (DATE)*, pages 865–869, 2007.