Defect-Tolerant Logic Hardening for Crossbar-based Nanosystems

Yehua Su and Wenjing Rao
ECE Department, University of Illinois at Chicago, IL 60607, USA
Email:{ysu8,wenjing}@uic.edu

***Abstract*— Crossbar-based architectures are promising for the future nanoelectronic systems. However, due to the inherent unreliability of nanoscale devices, the implementation of any logic functions relies on aggressive defect-tolerant schemes applied at the post-manufacturing stage. Most of such defect-tolerant approaches explore mapping choices between logic variables/products and crossbar vertical/horizontal wires. In this paper, we develop a new approach, namely fine-grained logic hardening, based on the idea of adding redundancies into a logic function so as to boost the success rate of logic implementation. We propose an analytical framework to evaluate and fine-tune the amount and location of redundancy to be added for a given logic function. Furthermore, we devise a method to optimally harden the logic function so as to maximize the defect tolerance capability. Simulation results show that the proposed logic hardening scheme boosts defect tolerance capability significantly in yield improvement, compared to mapping-only schemes with the same amount of hardware cost.**

I. Introduction

While the current CMOS technology is reaching its fundamental physical limits, nanoelectronic devices are proposed as alternatives for the next generation of electronic systems. Crossbar-based architectures have been shown to have significant potential for future nanoelectronic systems [1], [2]. At the crosspoint of any two wires is a bistable nanoscale switch capable of being toggled between the "on" and "off" states in connecting the two perpendicular wires. Such crossbar architectures are compatible with many nanoelectronic device candidates [3]–[7]. More importantly, crossbar architectures are similar to the traditional Programmable Logic Arrays (PLA), thereby supporting the implementation of arbitrary logic functions in two-level form [8]–[10].

Unlike the design flow for PLAs, rejecting defective crossbars becomes unacceptable in nanoelectronic environment, since most nanocrossbars are likely to contain massive number of defects. Instead, a map of defect types and locations for each nanocrossbar can be obtained based on well established testing methodologies in PLA architectures [11], [12]. Then, a unique configuration has to be performed for every chip individually, to somehow "bypass" or "utilize" the defects. As a result, the defect-tolerant logic implementation phase becomes the critical bottleneck during the design and manufacturing of crossbar-based nano circuits. Fig.1 provides a comparison between traditional CMOS based PLA and nanocrossbar design flows.

Most defect tolerance approaches employ some forms of logic mapping based schemes. In "defect-avoiding" approaches

This work is supported by NSF Grant 1149661.

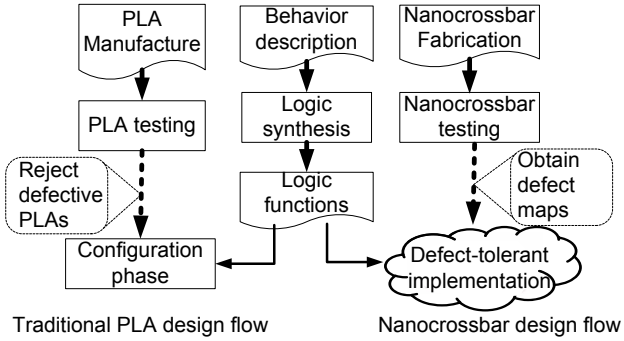

Fig. 1. Design flow comparison

[8], [13], circuits are built by avoiding the faulty wires and switches. In [13], a defect-free crossbar subset is searched for to map a logic function. Unfortunately, the chance of finding a large defect-free crossbar subset is low, given the high rate of defects. Furthermore, the complexity of finding a given sized perfect subcrossbar is NP-complete, and finding the maximum-sized one is NP-hard.

More aggressive approaches propose "defect-using" [14]–[23]. The work in [14], [15] formulates the problem with bipartite graph modeling, and proposes greedy algorithms based on the graph node weights to solve the graph isomorphism problem. Work in [16] expands defect types to open and bridging line defects. Heuristics were proposed in [14], [17] to reduce the search runtime. In [18], defect-tolerant logic mapping is translated into a SAT formulation. Work in [19] presents a yield model for logic mapping and identifies the threshold behavior in yield curves. In [20], logic mapping is performed under the constraints where defective switches are modeled with a delay cost. Work in [21] transforms the problem into a Bipartite SubGraph Isomorphism (BSGI) problem and also develops the mapping heuristic based on two dimensional sorting. Research work in [22] proposes to further tolerate defects by exploiting logical equivalences. In [23], the authors propose diversity mapping scheme by adding random operator into the greedy algorithm to improve the mapping success rate.

In this paper, we propose a new scheme of *logic hardening*, which enhances defect tolerance ability via carefully injected redundancy for a logic function. The technical contributions of this paper include: 1) hardening technique through duplicating logic variables which targets at the dominant open defects; 2) an analytical framework on achieving the optimal hardening for a logic function; and 3) an integrated algorithmic framework of hardening and mapping. Overall, the proposed hardening scheme is shown to improve yield significantly.

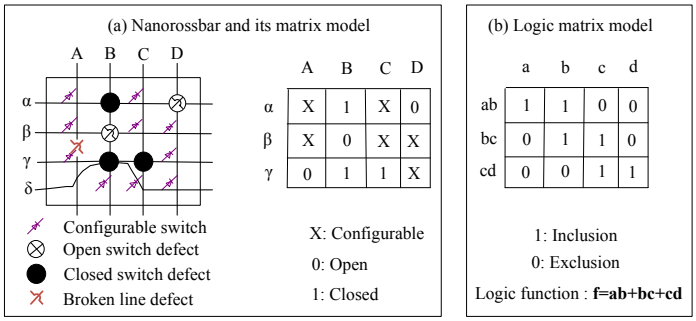

Fig. 2. (a) A defective crossbar and its crossbar matrix and (b) Logic function matrix

II. Preliminary: Defect-tolerant Logic Mapping

A. Defect model for nanoscale crossbar

Even though it is widely acknowledged that defect level will be exceedingly high for nanoelectronic systems, precise defect models might vary depending on further advancements in device and fabrication technologies. Nonetheless, for crossbar-based architectures, representative behavior-level models can be used for most defect-tolerant schemes.

From a functional perspective, device (switch) defects and line defects are of particular interests. The variety of physical defects may go beyond these cases, yet the basic strategy to deal with them can be categorized in two ways: 1) *catastrophic defects*, such as bridging wires, need to be avoided in the mapping process; 2) *non-catastrophic defects*, including stuck open/closed and misplaced switches, can be exploited in the mapping process. Fig.2(a) shows an example, where a wire with catastrophic defects needs to be excluded for the mapping process, while the non-catastrophic defects of open/closed switches can be utilized by exploring logic mapping choices.

In the paper, we assume: 1) the elimination of wires with catastrophic defects, and 2) a logic mapping framework exploiting "usable", non-catastrophic defects. A crossbar can be modeled by a matrix of cells, each representing one of the three possible connection types (as is shown in Fig.2(a)):

- *Configurable(X)*: the connection between perpendicular wires can be flexibly configured (defect-free).
- *Closed(1)*: the perpendicular wires are permanently stuck closed.
- *Open(0)*: the perpendicular wires are permanently disconnected.

B. Logic function model

A logic function[1] can be modeled by a matrix, based on two types of connectivity:

- *Inclusion*(1): a variable included in a product term.
- *Exclusion*(0): a variable not included in a product term.

Fig.2(b) shows the *logic matrix* for $f = ab + bc + cd$.

[1]Without loss of generality, two-level logic functions in the form of SOP (sum of products) are considered in describing the approaches, and we focus on the AND plane.

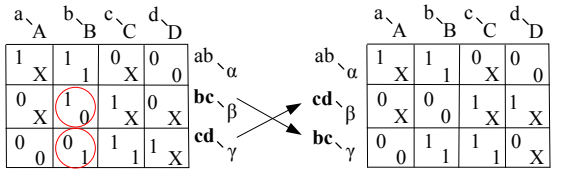

Fig. 3. Two mapping trial examples

C. Logic mapping

The problem of implementing a logic function onto a defective nanocrossbar translates into a *matrix mapping* problem: assign the rows / columns of a logic matrix to that of a crossbar matrix, under the following constraints imposed by the defects:

- Both *inclusion*(1) and *exclusion*(0) cells in a logic matrix are compatible with *Configurable(X)* cells.
- *Inclusion*(1) / *Exclusion*(0) cells in a logic matrix cannot be mapped onto *open(0) / closed(1)* cells in a crossbar matrix. We denote these two cases as $1 \rightarrow 0$ and $0 \rightarrow 1$ *mismatches*, respectively.
- *Closed(1) / Open(0)* cells in a crossbar matrix, though defective, can be used to support *inclusion*(1) / *exclusion*(0) cells in a logic matrix, thus forming the basis of defect using schemes.

A *valid mapping* is one that contains no mismatches. Fig.3 shows an example of two matrix mapping trials. The first is an invalid mapping with two mismatches, and the second, as a valid mapping, with no mismatches at all.

III. Logic Hardening

A. Motivations

Redundancy-based design for defect or fault tolerance purposes, known as "hardening", has been proposed for nanoelectronics [24]–[26]. For nanocrossbars with massive defects, thorough analysis and novel design for hardening are required. In this paper, the low cost hardening is essentially achieved by exploiting defect characteristics as well as logic function particularities.

In nanocrossbar fabrication, it is generally acknowledged that defects are dominated by missing and misplaced switches [17]. Therefore, it is expected that open defects are far more common than closed defects in the crossbar plane. In fact, open defects are the ones that can be effectively targeted in defect tolerance schemes. This paper presents a technique to tolerate open defects.

When mapping a two level logic function onto a crossbar, open defects result in $1 \rightarrow 0$ mismatches. For instance, f=ab+bc+ad+cd may become f'=$\cancel{ab}$+bc+ad+cd, because of an open defect. To overcome this, the logic function $f = ab+bc+ad+cd$ can be hardened as $F = ab_1b_2+b_1b_2c+ad+cd$, where b_1, b_2 are redundant copies of the original variable b. Function F is more robust than f, and can tolerate $(1 \rightarrow 0)$ mismatches introduced to variable b. A hardened form containing mismatches $F' = a\cancel{b_1}b_2+b_1\cancel{b_2}c+ad+cd = ab_2+b_1c+ad+cd$ is equivalent to the original logic form f.

Because variables are implemented by vertical wires of crossbars, when a variable is hardened, redundant vertical wires

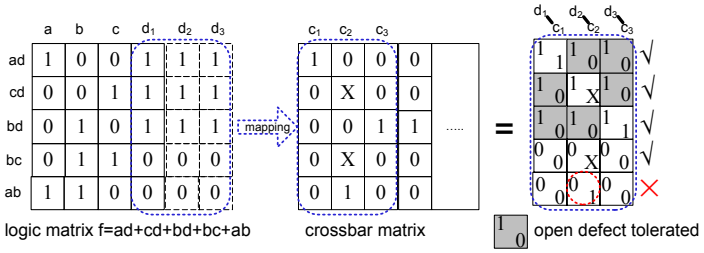

Fig. 4. Logic hardening tolerates open defects yet adds an extra $0 \rightarrow 1$ mismatch

need to be introduced in the crossbar. The duplicated copies of the variable will then appear in all the product terms due to the intersection of vertical wires and horizontal wires.

Two scenarios need to be considered when a variable x is hardened: 1) for the product terms containing x, no negative effects will be introduced; 2) for those product terms which do *not* contain x, extra $0 \rightarrow 1$ mismatches might be introduced. For instance, in Fig.4, logic hardening (through duplicating variable d) tolerates a number of $1 \rightarrow 0$ mismatches, and yet at the same time adds an extra $(0 \rightarrow 1)$ mismatch to the product term ab (which does not contain variable d).

Such tradeoff demands a *hardening equality checking* process to ensure that the hardened version of a logic function can be indeed successfully mapped. Basically, the following two criteria need to be satisfied, when mapping is conducted for all the duplicated columns in the logic function matrix:

- For every duplicated inclusion cell (represented by 1), as long as *one* of these copies is mapped without mismatches, it is considered as a correct mapping. In other words, hardening can tolerate many $1 \rightarrow 0$ mismatches (open defects).
- For every duplicated exclusion cell (represented by 0), *all* the column copies have to be mapped without mismatches. In other words, hardening cannot tolerate $0 \rightarrow 1$ mismatches (closed defects).

B. Algorithmic framework for hardening and mapping

The defect-tolerant logic mapping problem is essentially a constraint satisfiability problem, and we use a general backtracking framework to show the integration of mapping and hardening in Algorithm.1.

The backtracking algorithm explores all the mapping possibilities (correspondence of rows / columns between the logic and crossbar matrices) recursively. The inputs of this algorithm are a logic matrix (assuming hardening decisions have already been made), and a crossbar matrix (with open and close defects). The output of this algorithm is either a valid mapping (i.e., a successful implementation of the hardened logic function), or failure (no mapping is to be found). In the backtracking process, we found that it typically works the best to map rows and columns in an interleaving way. Such an approach makes it easier to screen out impossible mappings at an early stage.

Whenever one row (or duplicated columns) x from the crossbar matrix is mapped to the row (or multiple columns) l from the hardened logic matrix, any mismatches introduced in such a step is checked to see whether the mismatches can be tolerable.

Algorithm 1 Backtracking for Mapping and Hardening

Global variables: Hardened_Logic_Matrix, Xbar_Matrix

BT_Mapping(mapped_set)

1) if all rows and columns of Hardened_Logic_Matrix are in mapped_set
   return success
2) pick a row (duplicated columns), l, from Hardened_Logic_Matrix, such that $l \notin mapped\_set$
3) for every unmapped row (columns) x in Xbar_Matrix
   if (***Harden_Eq_Check***$(l \rightarrow x)$ == equivalent)
   //map l to x with constraints satisfied
   a) add $l \rightarrow x$ to mapped_set
   b) if ***BT_Mapping***(mapped_set) == success
   //recursive call for the rest of the mapping
   return success
   else remove $l \rightarrow x$ from mapped_set
   //$l \rightarrow x$ fails, try a different x in Xbar_Matrix
4) return failure
   //failed to map l to any possible x, backtrack

Harden_Eq_Check$(l \rightarrow x)$

for every mismatch m in mapping $l \rightarrow x$ {
  if($m \in mismatch(1 \rightarrow 0)$)
    if(x contains no 1 or X) *//cannot be tolerated*
      return inequivalent
  else *//$0 \rightarrow 1$ mismatch*
    return inequivalent }
return equivalent *//all mismatches are tolerated*

The two hardening equivalence criteria provided in the previous subsection are verified by the function $Harden\_Eq\_Check$. When a mismatch-free or equivalent mapping is found, the algorithm returns success. When no successful mapping exists, the algorithm eventually returns failure.

C. Hardening considerations

Since only inclusion cells in logic matrix can lead to mismatches $1 \rightarrow 0$, and variable duplication is helpful for tolerating $1 \rightarrow 0$ mismatches, the most frequently appearing variables (with more inclusion cells in a column) are the best candidates for hardening. Therefore, to maximize hardening benefits, frequently appearing variables should be chosen for hardening with a high priority.

IV. Optimal Hardening

Variable duplication is effective for tolerating open defects, yet adversely adds extra $0 \rightarrow 1$ mismatches. This means an "optimal hardening approach" needs to strike the tradeoff balance, since the overall benefits are determined by the combination of: 1) hardening selection and degree, 2) the heuristic-based mapping process, 3) logic function structure, and 4) crossbar size. In this section, we address the fundamental question of hardening selection and degree (as represented

by $Hardened\_Logic\_Matrix$ in the Algorithm.1) in order to achieve the maximum level of yield in the mapping process.

We propose to measure quantitatively the benefits brought by hardening, regardless of the specific mapping algorithms. A hardened logic function will be of a larger size, and the corresponding solution space (all the mapping possibilities) grows significantly. However, hardening (hopefully) turns more mapping trials into valid mappings without invalidating many existing solutions, thus rendering a higher percentage of valid solutions. Thus, the deciding factor will be an increase in *solution density*, which is the percentage of valid mappings in the entire solution space. Regardless of the mapping algorithms, it is generally easier to find a valid mapping when solution density is higher.

A. General formulation of solution density

We assume defects occur with a probability d, and consider a *closed defect ratio* r, thus making each device of $d \times r$ to be closed, and of probability $d \times (1-r)$ to be open. Given a logic function with *logic inclusion ratio* l (the percentage of inclusion cells in a logic matrix) and size n (the total number of matrix cells), each mapping trial is then valid with probability (when each cell is mapped without a mismatch):

$$p = (1 - d(1-r))^{nl}(1-dr)^{n(1-l)} \quad (1)$$

For each cell in the crossbar matrix to be mapped without a mismatch, it has to be either: 1) not an open cell (probability $1 - d(1-r)$) and mapped for one of the (nl) inclusion cells of the logic matrix; or 2) not a closed cell (probability $1-dr$) and mapped for one of the $(n-(1-l))$ exclusion cells of the logic matrix.

B. Solution density with a common hardening degree k

Hardening degree k is defined as the number of columns in a logic matrix that will be used for every variable to be hardened ($k = 1$ means no hardening). We start by assuming a common hardening degree k for all the columns of a logic matrix.

- When an inclusion cell is duplicated with k copies, these copies will be correctly mapped as long as one out of k copies can be mapped without mismatches. Thus, the probability for an inclusion cell with k copies to be correctly mapped is $1 - (d(1-r))^k$.
- When an exclusion cell is duplicated with k copies, all k copies have to be mapped without mismatches so as to be considered correctly. Thus, the probability that an exclusion cell with k copies can be successfully mapped is $(1-dr)^k$.

Based on the two cases, the solution density associated with a hardened logic function of hardening degree k is:

$$p_k = (1 - (d(1-r))^k)^{nl}((1-dr)^k)^{n(1-l)} \quad (2)$$

As hardening degree k grows, solution density p_k consists of two parts: the increasing part $1 - (d(1-r))^k$ and the decreasing part $(1-dr)^k$. Obviously, when the exponent nl for the increasing part becomes larger or the exponent $n(1-l)$ for the decreasing part becomes smaller, p_k increases. In other words, when a logic function has a high logic inclusion ratio l (high percentage of inclusion cells), leading to large nl and small $n(1-l)$, solution density p_k is improved significantly.

column	logic inclusion ratio	optimal hardening degree	p_k/p
col_0	0%	1	1
$col_{1\sim3}$	3.13%	1	1
$col_{4\sim6}$	6.25%	1	1
col_7	9.38%	1	1
col_8	28.13%	2	2.78
col_9	31.25%	2	3.36
col_{10}	34.38%	2	4.04
col_{11}	40.63%	2	5.86
col_{12}	43.75%	2	7.05
col_{13}	50.00%	2	10.23
col_{14}	53.13%	2	12.34
col_{15}	62.50%	3	14.42

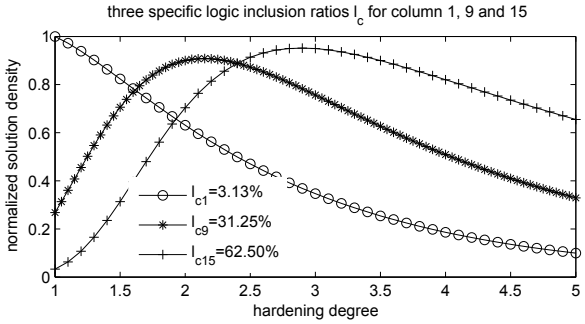

TABLE I
OPTIMAL HARDENING FOR $misex1$

C. Optimal hardening with fine-grained degrees k_i

To achieve better hardening result, we have to explore the possibility of having different hardening degree k_i for each variable. The equation of solution density, indicating the probability of mapping success, can be also applied to each column in the logic matrix, to determine how likely it can be mapped successfully. Suppose such probabilities for all the columns in a logic matrix are p_{k1}, p_{k2},...,p_{kc}. Then the solution density for the entire function will be $p_k = \prod p_{ki}$. The probability for each column being correctly mapped can be calculated independently using Eq.2, given the number of cells in the column n_c, and the logic ratio of the column l_c. Thus, Eq.2 for p_k can be applied to calculate the success rate p_{ki} for mapping a column i, when n and l are replaced with n_i and l_i, respectively.

The objection of optimal hardening is to maximize the solution density so as to increase the chances of finding a valid mapping (eventually yield improvement). This means to maximize the product $\prod p_{ki}$, by precisely calculating the hardening degree for each column in a logic matrix.

We analyze a specific benchmark $misex1$ from [27] as a case study. This logic function has 16 columns and 32 rows. Table I shows the logic inclusion ratio, from low to high, for all the columns. Suppose the columns are mapped onto a crossbar with closed defect ratio $r = 10\%$ and defect probability $d = 20\%$, then the optimal hardening degrees for these columns can be developed, as shown in the Table I. It is obvious that when columns have logic ratio less than 10%, they should not be

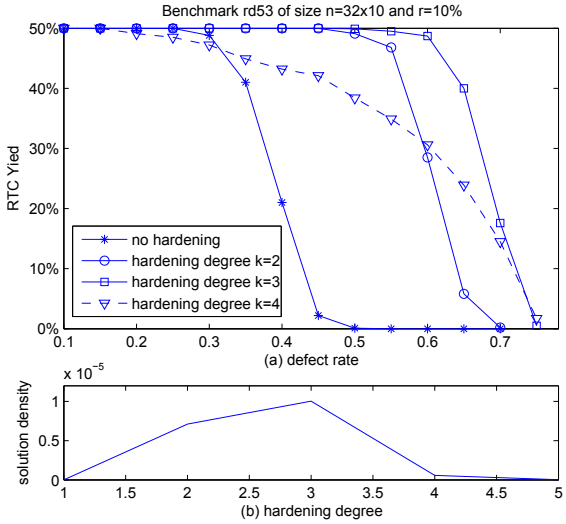

Fig. 5. Yield improvement with various hardening degree k

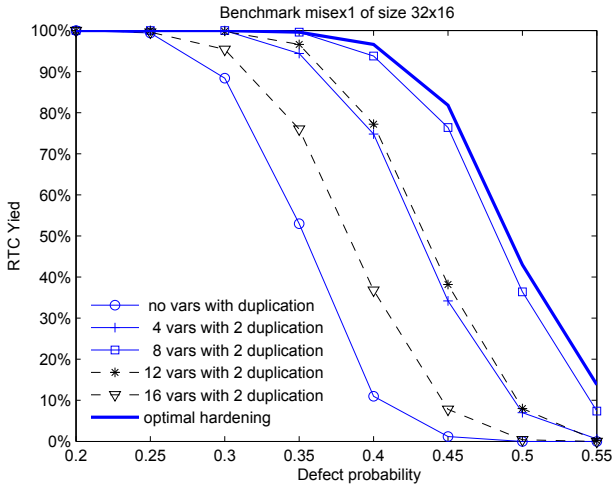

Fig. 6. Yield improvements with different hardening degree and $r = 10\%$

hardened. The column with a large logic ratio 62.5% can be hardened to the degree of 3 to produce the best performance. The figures in Table I show the three specific logic inclusion ratios, which represent the optimal hardening degree being 1, 2 and 3, for the corresponding columns. As logic inclusion ratio increases, the solution density improvement (p_k/p) becomes more significant.

V. Simulation Results

We examine the performance of the logic hardening scheme, represented by yield, defined as the percentage of successful logic mappings in a number of trials. In particular, we use the metric of *RunTime-Constrained (RTC)* yield [28], by setting a runtime upperbound for searching for a logic implementation in the experiments, and gathering yield data over 10^4 defective crossbars. The runtime limit is set to be 1 second for yield comparison. We use *size overhead*, defined as (row ratio)$\times$(column ratio) between crossbar and logic function, to indicate hardware cost. The algorithms[2] are implemented in Java on an Intel Core Duo 2.4GHz workstation with 2GB memory, and all the experiments are performed with the benchmarks in [27].

A. Hardening with a common degree

We first illustrate how yield can be improved by various hardening degrees. We expect that logic hardening can significantly improve yield when crossbars have mostly open defects. Because hardening adds redundancy in crossbars, for a fair comparison, crossbars of the same size are used for both hardening and non-hardening cases. Fig.5(a) shows the yield improves as hardening degree increases, where closed defect ratio $r = 10\%$, and the size overhead of crossbar versus logic function is 2×4. The following observations can be made based on the results:

- Yield improves as the degree increases until a threshold is reached, beyond which yield decreases as hardening degree increases.
- The benefit of yield improvement diminishes as the hardening degree increases. Yield improvement is most significant when the duplication level increases from one (no hardening) to two.

Fig.5(b) shows the corresponding solution density trend as the hardening degree increases. We can see that solution density is maximized with the hardening degree of 3, which explains why yield peaks at the same hardening degree as well.

B. Hardening frequently-appearing variables

We exhibit the effect of optimal hardening by studying two benchmarks. We proceed by choosing the most frequently-appearing variables during the hardening process, such that the effect can be illustrated clearly. The yield results of benchmark $misex1$ are shown in Fig.6, with size overhead being 2×3.

As the first 4 to 8 variables are hardened, yield is greatly improved. When less frequently-appearing variables are hardened in addition, yield decreases. Apparently, hardening most frequently-appearing variables improves yield substantially. As we can see from the case study in the previous section (Table I), only 8 variables need to be hardening so as to improve the yield. Besides, the yield curves of all kinds of hardening are bounded by the yield curve with optimal hardening (all column hardening degrees are shown in Table I).

C. Yield improvements by optimal hardening

We examine the overall yield improvement using the optimal logic hardening technique on the benchmarks shown in Table II (at closed defect ratio $r = 20\%$). These data points shown in Fig.7 are obtained at various defect probabilities, and the last column shows the average degree with optimal hardening technique of all the columns. The same size overhead 2×3 is applied to both hardening and non-hardening schemes. In all the cases, the proposed hardening technique displays significant yield improvement.

From the perspective of a logic function, two factors weigh the most: logic function size n and logic inclusion ratio l. Benchmark $5xp1$ and $sao2$ have significant yield improvement, mainly because of their large sizes. Even though benchmark $9sym$ has a relatively large size, the yield improvement is still

[2]Regardless of using hardening technique or not, the backtracking algorithm in the paper already adopts several defect tolerance heuristics, especially the previous work in [22].

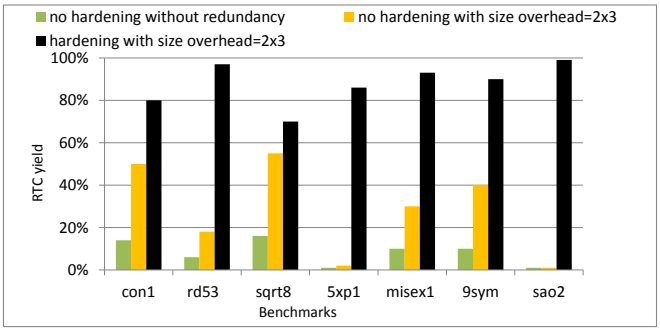

Fig. 7. Yield improvements with optimal hardening

low due to very low logic inclusion ratio. Benchmark $rd53$ has significant yield improvement, mainly because of its high logic inclusion ratio.

It is worth noticing in Fig.7 that the non-hardening scheme benefits significantly from added hardware redundancy of 2×3 as well (due to the enlarged solution space of mapping), compared to the case of no hardware redundancy. However, when the crossbar area is increased, the non-hardening scheme cannot take full advantage of the extra rows / columns. With logic hardening, these extra rows / columns in the mapping-only scheme can now be exploited in a more efficiently way, thus boosting yield significantly.

benchmark	size	inclusion ratio	average k with optimal hardening
con1	126	18.3%	1.28
rd53	320	45.0%	2.20
sqrt8	640	20.9%	1.56
5xp1	1050	12.5%	1.71
misex1	512	26.7%	1.56
9sym	1566	8.6%	2.00
sao2	1160	11.3%	1.62

TABLE II
BENCHMARK SIZE AND INCLUSION RATIO.

VI. CONCLUSIONS

Defect-tolerant logic implementation onto nanocrossbars becomes a new fundamental challenge in the post-fabrication design phase of future nanoelectronic systems. We propose a scheme to tolerate defects, targeting the dominating open defects, from the perspective of logic hardening with redundant logic variables. An analytical framework is developed to evaluate and fine-tune the amount of redundancy to be added to a given logic function, so as to optimally harden the logic function. We show the proposed scheme boosts defect tolerance capability with significant yield improvement.

REFERENCES

[1] A. DeHon, "Nanowire-based Programmable Architectures," *ACM Journal on Emerging Technologies in Computing System*, vol. 1, no. 2, pp. 23–32, July 2005.

[2] W. Robinett, G. S. Snider, P. J. Kuekes, and R. S. Williams, "Computing with a Trillion Crummy Components," *Commun. ACM*, vol. 50, pp. 35–39, 2007.

[3] M. A. Kastner, "The Single-Electron Transistor," *Review of Modern Physics*, vol. 64, pp. 849–858, 1992.

[4] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, "Quantum Cellular Automata," *Nanotechnology*, vol. 4, pp. 49–57, 1993.

[5] S. A. Wolf, D. D. Awschalom, R. A. Buhrman, J. M. Daughton, S. von Molnar, M. L. Roukes, A. Y. Chtchelkanova, and D. M. Treger, "Spintronics: A Spin-based Electronics Eision for the Future," *Science*, vol. 294, pp. 1488–1495, 2001.

[6] P. Mazumder, S. Kulkarni, M. Bhattacharya, J. P. Sun, and G. I. Haddad, "Digital Circuit Applications of Resonant Tunneling devices," *Proceedings of the IEEE*, vol. 86, no. 4, pp. 664–686, Apr. 1998.

[7] P. Avouris, J. Appenzeller, R. Martel, and S.Wind, "Carbon Nanotube Electronics," *Proceedings of the IEEE*, vol. 91, no. 11, pp. 1772–1784, 2003.

[8] A. DeHon, "Array-Based Architecture for FET-Based, Nanoscale Electronics," *IEEE Transactions on Nanotechnology*, vol. 2, no. 1, pp. 109–162, 2003.

[9] A. DeHon and B. Gojman, "Crystals and Snowflakes: Building Computation from Nanowire Crossbars," *Computer*, vol. 44, no. 2, pp. 37 –45, 2011.

[10] W. Rao, C. Yang, R. Karri, and A. Orailoglu, "Toward Future Systems with Nanoscale Devices: Overcoming the Reliability Challenge," *Computer*, vol. 44, no. 2, pp. 46 –53, 2011.

[11] P. Bose and J. Abraham, "Test Generation for Programmable Logic Arrays," *19th Design Automation Conference*, pp. 574 –580, June 1982.

[12] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-signal VLSI Circuits*, 2000.

[13] M. B. Tahoori, "Defect Tolerance in Crossbar Array Nano-Architectures," *Emerging Nanotechnologies: Test, Defect Tolerance, and Reliability, Springer*, pp. 121–151, 2007.

[14] W. Rao, A. Orailoglu, and R. Karri, "Topology Aware Mapping of Logic Functions onto Nanowire-base Crossbar Architectures," *IEEE/ACM Design Automation Conference*, pp. 723–726, July 2006.

[15] B. Ghavami, A. Tajary, M. Raji, and H. Pedram, "Defect and Variation Issues on Design Mapping of Reconfigurable Nanoscale Crossbars," *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on*, pp. 173 –178, july 2010.

[16] J. Huang, M. Tahoori, and F. Lombardi, "On the Defect Tolerance of Nano-scale Two-Dimensional Crossbars," *In 19th IEEE International Symposiumon Defect and Fault Tolerance (DFT) in VLSI Systems*, pp. 96–104, 2004.

[17] H. Naeimi and A. DeHon, "A Greedy Algorithm for Tolerating Defective Crosspoints in NanoPLA Design," *In Proc. Intl Conf. on Field-Programmable Technology*, pp. 49–56, 2004.

[18] Y. Zheng and C. Huang, "Defect-aware Logic Mapping for Nanowire-based Programmable Logic Arrays via Satisfiability," *Design, Automation and Test in Europe (DATE)*, pp. 1279–1283, Apr. 2009.

[19] T. Hogg and G. Snider, "Defect-tolerant Logic with Nanoscale Crossbar Circuits," *Journal of Electronic Testing*, vol. 23, pp. 117–129, Jun. 2007.

[20] C. Tunc and M. Tahoori, "On-the-fly Variation Tolerant Mapping in Crossbar Nano-Architectures," *VLSI Test Symposium (VTS), 2010 28th*, pp. 105 –110, 2010.

[21] S. Go andren, H. Ugurdag, and O. Palaz, "Defect-aware nanocrossbar logic mapping using bipartite subgraph isomorphism and canonization," *Test Symposium (ETS), 2010 15th IEEE European*, p. 246, may 2010.

[22] Y. Su and W. Rao, "Defect-Tolerant Logic Implementation onto Nanocrossbars by Exploiting Mapping and Morphing Simultaneously," *International Conference on Computer Aided Design (ICCAD) 2011*, pp. 456–462, 2011.

[23] B. Yuan and B. Li, "Diversity Mapping Scheme for Defect and Fault Tolerance in Nanoelectronic Crossbar," *Information Science and Technology (ICIST), 2011 International Conference on*, pp. 149 –154, march 2011.

[24] I. Polian and W. Rao, "Selective Hardening of NanoPLA Circuits," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pp. 263–271, Oct. 2008.

[25] M. Stanisavljevic, A. Schmid, and Y. Leblebici, "Optimization of Nanoelectronic Systems Reliability Under Massive Defect Density Using Distributed R-fold Modular Redundancy (DRMR)," in *24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2009*, Oct. 2009, pp. 340 –348.

[26] J. Dai, L. Wang, and F. Jain, "Analysis of Defect Tolerance in Molecular Crossbar Electronics," pp. 529 –540, Apr. 2009.

[27] "Collaborative Benchmarking Laboratory," *1993 LGSynth Benchmarks*, North Carolina State University, Department of Computer Science,1993.

[28] Y. Su and W. Rao, "Defect-tolerant Logic Mapping on Nanoscale Crossbar Architectures and Yield Analysis," *IEEE International Symposium on Defect and Fault Tolerance (DFT) in VLSI Systems*, pp. 322–330, Oct. 2009.