Efficient Function Mapping in Nanoscale Crossbar Architecture

Joon-Sung Yang and Rudrajit Datta
Computer Engineering Research Center
The University of Texas at Austin,
{js21.yang@mail.utexas.edu, rudrajit@utexas.edu}

Abstract

Nanoscale crossbar architectures have been proposed as viable alternatives for overcoming the fundamental physical limitations of CMOS technology. However due to the manufacturing processes for nanofabrication and their smaller feature sizes, defect densities are higher. This paper presents an efficient function mapping method in the presence of high defect rates for nanoscale crossbar arrays. Given a function and a defect map that describes fault patterns in the crossbar architecture, the approach described here tries to find a valid function mapping, if one exists, using a matrix representation. A set of constraints are derived to preserve semantics and then Integer Linear Programming (ILP) is used to solve the equations. Experimental results show the proposed approach provides efficient utilization of nanoscale crossbars in mapping functions in presence of high defect rates.

1. Introduction

Advances in nanotechnology have allowed researchers to focus their efforts on the development of viable nanoscale systems. 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 [Chaudhury 09], [Dehon 05], [Polian 08]. This offers the possibility of significantly denser circuits than is possible with current lithography-based manufacturing.

Nanoscale devices such as Carbon Nano Tubes (CNT) and Silicon Nanowires (SiNW) form the principal building blocks of many nanoscale logic devices and computing architecture recently proposed. Nanowires show various attractive properties such as simple and economical fabrication using bottom-up self assembly or nanoimprint lithography, regular structure and so on [Hogg 07]. A nanowire crossbar architecture is a two dimensional array of intersecting sets of orthogonal nanowires which can be programmed electronically to exhibit properties of various active and passive devices [Chaudhury 09], such as conventional diode, Field Effect Transistor or a resistor [Chaudhury 09], [Dehon 05]. This basic structure can be programmed as *on* and *off* states and also can be reconfigured [Han 02], [Heath 98], [Goldstein 01, 03], [Heath 98], [Rao 06] to implement the desired logic function post-fabrication [Huang 04].

Unlike prevalent CMOS devices, a nanocrossbar is more susceptible to transient faults as well as permanent faults introduced by manufacturing defects and have orders of magnitude higher defect rates [Farazmand 09], [Hogg 07], [Rao 06]. Hence, it is a significant design challenge to tolerate the high defect rates, enhance on-line detection capability and to utilize the defective nanowire crossbar arrays for logic functions to avoid low yield.

Various fault tolerance techniques have been proposed to compensate for the high defect rates of nanowire circuits. Fault masking methods can be applied to nanowire crossbars. A dynamically adaptive N modular redundancy (NMR) approach was proposed with reconfiguration algorithms [Rao 06]. This approach was shown to mitigate both permanent defects as well as online faults using flexible NMR and reconfiguration. [Rao 07] proposed a fault tolerance technique using Boolean logic tautology that focused on the class of faults caused by missing devices at nanocrossbar crosspoints. Multi-level logic approaches can also be applied to enhance their reliability, however, unlike in [Rao 06], the majority voter is not required. This approach can achieve a moderate level of fault tolerance while significantly reducing hardware redundancy. Selective hardening of NanoPLAs was introduced in [Polian 08]. In this method high probability faults were identified using an analytical approach as well as simulation. Once these are identified, a selective hardening process is performed, with the dual goal of increasing the robustness of NanoPLA and optimizing the area cost for extra hardware added for fault tolerance.

For reliable operation of nanocrossbar architectures, a concurrent multiple error detection scheme using multistage nanocrossbar architectures was presented in [Farazmand 09]. The proposed dual-rail logic implementation can detect all single (transient and permanent) faults as well as most multiple (transient and permanent) faults. The effectiveness of hardware duplication, triple modular redundancy (TMR) and parity checking methods were also investigated as part of this work.

It is important to utilize defective nanowire crossbar arrays to maintain high manufacturing yield. One technique to extract a defect-free array for defect-unaware design flow was proposed by [Tahooir 05]. This design flow

extracts a *k x k* defect-free sub-matrix from the *n x n* original defective nanofabric where $k < n$. This method helps to minimize the post-fabrication design efforts by finding a defect-free nanocrossbar. This work has been extended to show how it can be integrated with the design flow [Tahoori 06]. [Naeimi 04] proposes a greedy algorithm based on bipartite matching for mapping a logic function to a nanocrossbar where the inputs are pre-assigned. [Zheng 09] introduces a mapping technique using Boolean satisfiability (SAT) for both input and output planes optimizing for the run-time of their mapping algorithm.

This paper presents a new constraint based method for mapping of a function to the nanowire array, using integer linear programming (ILP) techniques to solve the mapping problem. A given function is represented as a matrix and a defective nanowire array is represented using a defect map. The defect map is generated by identifying the locations and types of permanent defects through test and diagnostic procedures. In this paper, defects in the nanowire array are modeled as stuck-open and stuck-closed switches. Using these two primary defect types, a list of constraints is drawn up which needs to be conformed to in order for the mapping to be valid. While deriving constraints, illegal mappings are identified and invalidated, thereby pruning the solution space. This helps to reduce computational complexity while finding a valid mapping, especially for high defect rates. Once the constraints have been formulated, an ILP solver is used to solve the equations.

The rest of the paper is organized as follows. Sec. 2 gives the background of nanowire technology and explains the fault models used in the paper. In Sec. 3, the matrix representation and the constraints governing the mapping are described at length. Experimental results are presented in section 4 and conclusions are in section 5.

2. Nanowire Technology and Its Fault Models

Nanowire crossbar is fabricated in a bottom-up manner. This introduces a regular structure of nano-devices [Chaudhury 09], [Dehon 05], [Goldstein 01], [Hogg 07], [Rao 06], [Tahoori 05]. Fig. 1 shows a nanowire crossbar architecture which consist of two parallel planes of nanowires. Two planes are separated by a thin chemical layer with the nanowires running horizontally and vertically on each plane respectively. The region where the horizontal and vertical wires cross each other at right angles is called a crosspoint and it can be configured with voltage control. Depending on the configuration, they behave as a diode-connection for *"on"* state or a resistive open connection for *"off"* state. Hence, each crosspoint is modeled as a switch thus forming a programmable logic array (PLA) of nanowires. This can then be used to implement logical AND/ OR functions.

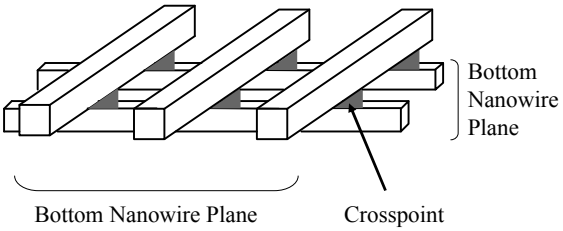


Figure 1. Nanowire Crossbar Architecture

For CMOS based PLAs, the manufacturing tests have been studied thoroughly. Similar to CMOS based PLAs, fault models for nanocrossbar arrays can be considered [Rao 07], [Tahoori 05]. In general, the nature of manufacturing defects in nanowires can be summarized as below –

Switch stuck-open fault: A crosspoint is considered stuck-open when the junction is permanently disconnected. It behaves similarly as an open switch. Hence, a crosspoint with stuck-open fault in nanocrossbar array can still be assigned to a function only if it is not used in a function.

Switch stuck-closed fault: A crosspoint modeled as stuck-closed can be considered as a shorted switch. A crosspoint with stuck-closed fault in nanocrossbar array can be used for a function mapping when it is always used in a function.

Nanowire open and bridging fault: All switches connected to a nanowire with an open fault and a bridging fault are treated as unusable as shown in [Huang 04], [Tahoori 05].

In the following section, the proposed scheme is explained in detail.

3. Details of Function Mapping in Nanowire Array

The following subsections describe each of the steps in the proposed procedure for function mapping in nanowire crossbar architecture.

3.1 Matrix Representation

For the function mapping technique, the logic function is represented by a matrix. Assume that the function *f*, expressed as sum-of-products, is

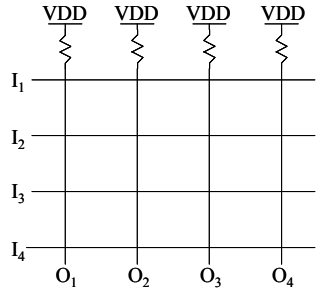
$$f = O_{f1} + O_{f2} \quad \text{---------} \quad (1)$$

where $O_{f1} = I_a I_b I_c$ and $O_{f2} = I_b I_c$.

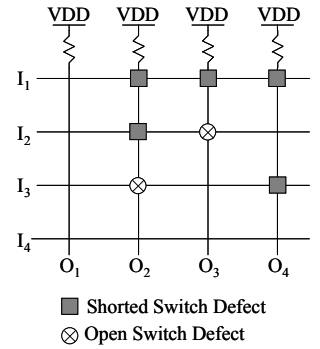
The function *f* require two wires for the two product terms (O_{f1}, O_{f2}) and three wires, one each for the three literals (I_a, I_b, I_c), to implement the function in a nanowire crossbar array. Input and output relations can be represented by a matrix as shown in Fig. 2. If the logic function, represented as sum-of-products, has *k* variables and *l* product terms, a function matrix of size *l* x *k* is generated. Each literal is designated by a column and each row signifies a product term. However, the mapping of each column to a particular variable is not fixed at this time and is assigned later.

$$\begin{array}{c} \\ O_{f1} \\ O_{f2} \end{array} \begin{array}{c} \begin{array}{ccc} I_a & I_b & I_c \end{array} \\ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{array}$$

Figure 2. Function Matrix Representation



(a) Defect-Free Nanowire Crossbar Array



(b) Nanowire Crossbar Array with Open and Short Defects

Figure 3. Defect-Free/Defective Nanowire Crossbar Array

Fig. 3 (a) shows a defect-free nanowire crossbar array of size 4 x 4 (4 outputs and 4 inputs). Fig. 3 (b) shows what a nanowire would look like in presence of defects.

After a nanowire array is manufactured, the location and type of defects are identified using test and diagnostic procedures [Tahoori 05] and a defect map is generated. Such a nanowire with manufacturing defects is shown in Fig. 3 (b). In a defect-aware design flow, a defect map is incorporated to a high-level design for function mapping, fault tolerance enhancement and so on. In this paper, as per the fault models described in the preceding section, the condition of each switch is classified as open, configurable or short. *0*, *1* and *2* are assigned for open, configurable and short switches respectively and nanowires with open and bridging faults are discarded as in [Huang 04], [Tahoori 05]. As shown in Fig. 4, the nanowire array with defects from Fig. 3 (b) can be represented as a matrix. The rows in the matrix represent the vertical wires and the horizontal wires are represented by the columns in the matrix. Since crosspoint between O_2 and I_1 has a shorted switch defect and O_2 and I_3 has an open switch, *0* and *2* are assigned to the corresponding columns and rows in Fig. 4. In this manner, nanowire crossbar in the presence of defects can be represented as a matrix.

$$\begin{array}{c} \\ O_1 \\ O_2 \\ O_3 \\ O_4 \end{array} \begin{array}{c} \begin{array}{cccc} I_1 & I_2 & I_3 & I_4 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 2 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Figure 4. Matrix Representation of Defective Nanowire Crossbar in Fig. 3 (b)

3.2 Constraint Equations Representation

Given a function to be mapped along with a defect map that describes the defects in the crossbar architecture, the constraint equations are derived which would govern a valid function mapping.

A variable X is defined for row mapping and X_{i_j} denotes the mapping of the i^{th} row in the function matrix to j^{th} row in the crossbar matrix. If there is a valid row to row mapping found, X_{i_j} will be assigned *'1'* and *'0'* will be assigned if no mapping is found. Two constraints are set in terms of row mapping. For a k_1 x k_2 function matrix to be mapped to a n_1 x n_2 nanowire matrix,

1) Rows in the function matrix can be mapped to at most one row in the crossbar matrix.
Equation (1)

$$\sum_{j=1}^{n_1} X_{i\_j} = 1, \qquad for\ each\ i = 1 \sim k_1$$

2) At most one row from the function matrix can be mapped to a row of the crossbar matrix.
Equation (2)

$$\sum_{i=1}^{k_1} X_{i\_j} \le 1, \qquad for\ each\ j = 1 \sim n_1$$

Similarly, constraints are laid down in terms of column to column mapping. The variable $Y_{i'\_j'}$ denotes mapping of the i'^{th} column in the function matrix to j'^{th} column in the crossbar matrix. The variable $Y_{i'\_j'}$ is assigned the value '*1*' when a valid mapping exists between the i'^{th} column in the function matrix and the j'^{th} column in the crossbar matrix. With respect to the same function and crossbar matrices, the column mapping relations are,

3) Columns of the function matrix can be mapped to at most one column in the crossbar matrix.
Equation (3)

$$\sum_{j'=1}^{n_2} Y_{i'\_j'} = 1, \qquad for\ each\ i' = 1 \sim k_2$$

4) At most one row from the function matrix can be mapped to a row of the crossbar matrix.
Equation (4)

$$\sum_{j'=1}^{k_2} Y_{i'\_j'} \le 1, \qquad for\ each\ j' = 1 \sim n_2$$

One more variable is introduced to maintain the coherency of the mapping. This variable $Z_{i\_j\_i'\_j'}$ is assigned '*1*' if and only if $X_{i\_j} = Y_{i'\_j'} = 1$ and *'0'* is assigned for a particular *i, j, i', j'*.

5) This variable ensures that no row (column) of the function matrix is mapped to a row (column) of the crossbar matrix when sufficient number of fault free crosspoints is not available. The constraint relation on $Z_{i\_j\_i'\_j'}$ can be expressed as,

Equation (5)

$$X_{i\_j} + Y_{i'\_j'} - 2 * Z_{i\_j\_i'\_j'} \ge 0$$

6) For a valid mapping to exists between the function and crossbar matrices, there needs to exist a valid mapping for each bit of the function matrix

Equation (6)

$$for\ each\ i = 1 \sim k_1, j = 1 \sim n_1, i' = 1 \sim k_2, j' = 1 \sim n_2$$

$$\sum_{i=1}^{k_1} \sum_{j=1}^{n_1} \sum_{i'=1}^{k_2} \sum_{j'=1}^{n_2} Z_{i\_j\_i'\_j'} = i * j,$$

Using these six constraint relations, a valid mapping between the function matrix and the crossbar matrix can be obtained, if one exists.

For the matrices shown in Fig. 2 and Fig. 4, the constraint equations would be as follows

$$\text{where } i = f_1, f_2 \text{ and } j = 1,2,3,4$$
$$i = f_1 : X_{f1\_1} + X_{f1\_2} + X_{f1\_3} + X_{f1\_4} = 1$$
$$i = f_2 : X_{f2\_1} + X_{f2\_2} + X_{f2\_3} + X_{f2\_4} = 1$$

$$j = 1 : X_{f1\_1} + X_{f2\_1} \le 1$$
$$j = 2 : X_{f1\_2} + X_{f2\_2} \le 1$$
$$j = 3 : X_{f1\_3} + X_{f2\_3} \le 1$$
$$j = 4 : X_{f1\_4} + X_{f2\_4} \le 1$$

$$\text{where } i' = I_a, I_b, I_c \text{ and } j' = I_1, I_2, I_3, I_4$$
$$i' = I_a : Y_{Ia\_I1} + Y_{Ia\_I2} + Y_{Ia\_I3} + Y_{Ia\_I4} = 1$$
$$i' = I_b : Y_{Ib\_I1} + Y_{Ib\_I2} + Y_{Ib\_I3} + Y_{Ib\_I4} = 1$$
$$i' = I_c : Y_{Ic\_I1} + Y_{Ic\_I2} + Y_{Ic\_I3} + Y_{Ic\_I4} = 1$$

$$j' = I_1 : Y_{Ia\_I1} + Y_{Ib\_I1} + Y_{Ic\_I1} \le 1$$
$$j' = I_2 : Y_{Ia\_I2} + Y_{Ib\_I2} + Y_{Ic\_I2} \le 1$$
$$j' = I_3 : Y_{Ia\_I3} + Y_{Ib\_I3} + Y_{Ic\_I3} \le 1$$
$$j' = I_4 : Y_{Ia\_I4} + Y_{Ib\_I4} + Y_{Ic\_I4} \le 1$$

$$\text{where } i = f_1, f_2, \text{and} j = 1,2,3,4$$
$$i' = I_a, I_b, I_c \text{ and } j = I_1, I_2, I_3, I_4$$
$$i' = f_1, j = 1, i' = I_a, j' = I_1 :$$
$$X_{f1\_1} + Y_{Ia\_I1} - 2Z_{f1\_1\_Ia\_I1} \ge 0$$
$$\vdots$$
$$i' = f_2, j = 4, i' = I_c, j' = I_4 :$$
$$X_{f2\_4} + Y_{Ic\_I4} - 2Z_{f2\_4\_Ic\_I4} \ge 0$$

3.3 State-Space Pruning

The above constraint relations if used as input to an ILP solver yield the desired mapping, if one exists. However, for a nanowire array with high defect density, the computational complexity could be fairly high if the full state-space is used. When the constraint equations are being solved, a bigger solution space would require more time before a valid mapping could be obtained, if one exists.

Hence, before invoking the ILP solver, the state-space is pruned to invalidate the illegal choices. The ILP solver demonstrates considerably lower run-times with a pruned state-space as shown in the results in section 5. Note that the concept of state-space pruning can be used for any mapping technique. The pruning of illegal mappings is explained in next using the example of Fig. 2 and Fig. 4.

Because there are less number of available switches in the crossbar matrix than required in a function matrix, the row O_{f1} from the function matrix cannot be mapped either row O_2 or row O_4. Hence the variables X_{f1_O2} and X_{f1_O4} can be both assigned '*0*'. Similarly, column I_c cannot be mapped to column I_1. Hence, *'0'* is automatically assigned to the variable, Y_{Ic_I1}. Finally, the bit O_{f2}, I_a from the function matrix cannot be mapped to the bit O_3, I_2 of the crossbar matrix, which implies $Z_{f2\_3\_Ia\_I2}$ be assigned '*0*'. This pruning technique is performed before the constraint relations are used as an input to an ILP solver.

4. Experimental Results

Experimental results are presented in this section. 300 functions were randomly generated for the purpose of mapping them to defective nanocrossbar arrays using the proposed method. Functions with 6 variables and 6 product terms and 8 variables and 8 produce terms were generated and represented as 6 x 6 and 8 x 8 function matrices respectively. 6 x 6 and 8 x 8 crossbar arrays are used for experiments, however, the proposed method can be scaled to handle crossbar arrays of larger sizes. SCIP version 1.2.0 [Scip] is used as a solver to find a valid function mapping to nanocrossbar array.

The main goal of this paper is to find a valid mapping, when one exists, of a function to a nanowire corossbar array in the presence of high defect density. Fig. 5 plots the function mapping success against different defect rates in nanocrossbar array. Defect maps with switch stuck-open and stuck closed defects were simulated by randomly injecting errors at different defect rates. For each such defect map, 300 random functions were mapped to 300 random nanocrossbar arrays. Since the proposed method uses an ILP based technique, a valid function mapping is guaranteed to be found if it exists. As shown in Fig. 5, a 100% valid mapping is found for defect rates of 5% ~ 25%. This means that valid mappings were found for all 300 functions for these defect rates. For a defect rate of 45%, the proposed mapping succeeds in 96% of the cases. 4% of mapping failure is due to defect patterns in nanocrossbar array being such one or more of the constraints required for a valid mapping is not satisfied. Note that, for such high defect rates (≥ 30%), no other mapping algorithms and/or heuristics is able find a valid mapping [Tahoori 05], [Rao 06], [Zheng 09].

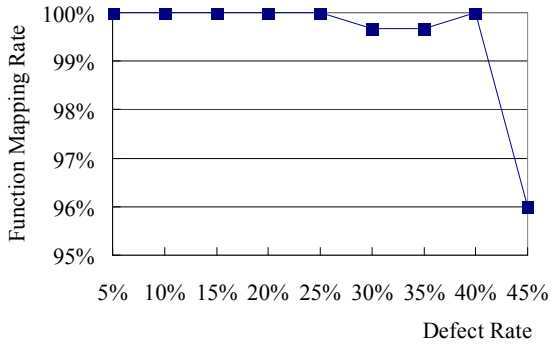


Figure 5. Function Mapping Rate vs. Defect Rate for 300 Functions

Runtimes for finding these valid mapping are shown in Fig. 6. Runtimes shown are normalized with respect to that required by the defect map with 5% defect rate. As expected, since the primary target of the proposed method is to find a valid mapping, if one exists, at higher defect rates runtime is traded off for functionality and consequently the ILP solver takes longer in these cases. But it needs to be mentioned here that unlike heuristics or SAT based mapping approaches, the mapping efficiency degrades only marginally in the proposed method for increasing defect rates.

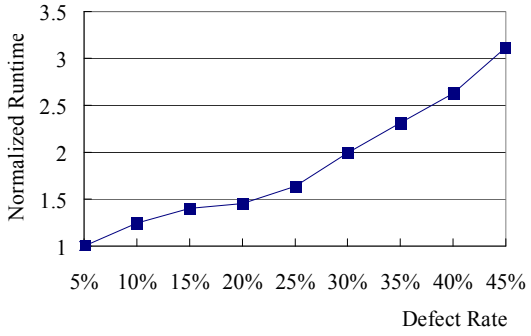


Figure 6. Normalized Runtime for Solving Proposed Constraints

For high defect rates, state-space pruning is applied to improve runtime. Pruning is done by pre-assigning values to some of the constraints as described in Sec. 3.3. The pruning technique can be applied to any mapping algorithms for runtime speed up. For 45% defect rate, experimental results show that state-space pruning technique achieves

44.8% improvement in runtime than when unmodified solution space is used. This confirms the previous claim that state-space pruning helps to reduce computation complexity for high defect rates.

5. Conclusions

In this paper, a technique for mapping a function to a nanocrossbar array for high defect rates is proposed. The function mapping is governed by a set of constraints, which are solved using an ILP to obtain a valid mapping if one exists. Computational complexity is reduced using state-space pruning techniques. Experimental results show that the proposed method shows high efficiency in finding valid mappings for high defect rates.

Acknowledgements

The authors would like to thank Professor Nur A. Touba at the University of Texas at Austin for providing many helpful suggestions.

References

[Chaudhury 09] Chaudhary, S., Minsu Choi, Yong-Bin Kim, "Probabilistic analysis of design mapping in asynchronous nanowire crossbar architecture," *Instrumentation and Measurement Technology Conference*, pp. 1116-1120, 2009

[Dehon 05] Dehon. A., "Nanowire-based programmable architectures," *ACM Journal on Emerging Technologies in Computing Systems*, July 2005

[Farazmand 09] Farazmand, N. M. B. Tahoori, "Online multiple error detection in crossbar nano-architectures," *IEEE International Conference on Computer Design*, pp. 335-342, 2009.

[Goldstein 01] Goldstein, S. C. and M. Budiu, "NanoFabrics: Spatial Computing Using Molecular Electronics," *International Symposium of Computer Architecture*, pp. 178–191, 2001

[Goldstein 03] Goldstein, S. C., M. Budiu, M. Mishra and G. Venkataramani, "Reconfigurable Computing and Electronic Nanotechnology," *International Conference on Application-Specific System, Architecture and Processors*, pp. 132–143, 2003

[Han 02] Han, J. and P. Jonker, "A System Architecture Solution for Unreliable Nanoelectronic Devices," *IEEE Transactions on Nanotechnology*, vol. 1, n. 4, pp. 201–208, December 2002

[Heath 98] Heath, J. R., P. J. Kuekes, G. S. Snider and S.Williams, "A Defect- Tolerant Computer Architecture: Opportunities for Nanotechnology," *Science*, vol. 280, pp. 1716–1721, June 1998

[Hogg 07] Hogg, T. and G. Snider, "Defect-tolerant Logic with Nanoscale Crossbar Circuits," *Journal of Electronic Testing*, pp. 117-129, 2007

[Huang 04] Huang, J., M. B. Tahoori and F. Lombardi, "On the defect tolerance of nano-scale two-dimensional crossbars," *Proc. of IEEE International Symposium of Defect and Fault Tolerance of VLSI Systems*, pp. 96-104, 2004

[Naeimi 04] Naeimi, H. DeHon, A. "A greedy algorithm for tolerating defective crosspoints in nanoPLA design," *Proc. of International Conference on Field-Programmable Technology*, pp. 49-56, 2004.

[Polian 08] Polian, I., W. Rao, "Selective Hardening of NanoPLA Circuits," *Proc. of IEEE International Symposium of Defect and Fault Tolerance of VLSI Systems*, pp. 263-271, 2008.

[Rao 06] Rao, W. and Orailoglu, A. Karri, R., "Nanofabric topologies and reconfiguration algorithms to support dynamically adaptive fault tolerance," *Proc. of IEEE VLSI Test Symposium*, 2006.

[Rao 07] Rao, W. Orailoglu, A. Karri, R., "Logic level fault tolerance approaches targeting nanoelectronics PLAs," *Design, Automation and Test in Europe*, pp. 1-5, 2007.

[Scip] Solving Constant Integer Program, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Division Scientific Computing, Department Optimization, http://scip.zib.de/

[Tahoori 05] Tahoori, M.B., "A mapping algorithm for defect-tolerance of reconfigurable nano-architectures," *International Conference on Computer-Aided Design*, pp. 668-672, 2005.

[Tahoori 06] Tahoori, M.B., "Application-independent defect-tolerant crossbar nano-architectures," *International Conference on Computer-Aided Design*, pp. 730-734, 2006.

[Tunc 10] Tunc, C. and M. B. Tahoori, "On-the-flyp Variation Tolerant Mapping in Crossbar Nano-Architectures," *Proc. of IEEE VLSI Test Symposium*, 2010

[Zheng 09] Zheng, Y. and C. Huang, "Defect-aware logic mapping for nanowire-based programmable logic arrays via satisfiability," *Design, Automation and Test in Europe*, pp. 1279-1283, 2009.