I. INTRODUCTION

At present, computer systems are mainly based on the complementary metal-oxide semiconductor (CMOS). Nevertheless, a consensus benchmark set by semiconductor manufacturers and researchers known as International Technology Roadmap for Semiconductor (ITRS) has predicted that CMOS cannot be scaled beyond 22nm in year 2018 [1]. It is due to several challenges face by CMOS such as technology, economics, physical, and power thermal [2].

New devices called nanoelectronic devices have been aggressively developed with the purpose to complement and replace CMOS. Recently, various nanowires and carbon nanotubes (CNTs) devices have drawn much attention because of their possible use in future electronics applications [2][4][5]. These emerging technologies take advantage of self-assembly to avoid sophisticated and expensive nanoscale lithography. In self-assembly, the nanostructures are stochastically built-that is, self-assembled from the bottom up, on a molecule-by-molecule basis. Most “bottom-up” approaches are statistical by nature and are lacking the fidelity of top-down fabrication. Bottom-up approaches for making nanoscale devices circumvent some of the fabrication challenges and the high cost of equipment that are associated with established top-down CMOS processing.

Integrating nanowires or CNTs based nanodevices into computing systems is facing new challenges not encountered in conventional CMOS. Self assembly-based manufacturing imposes doping/layout constraints on nanoscale circuits, restricting routing and placement. CNTs and nanowires can be arranged into a nanoscale crossbar structure, a promising building block for nanoelectronics circuits[6]. The crossbar architecture consists of two parallel planes separated by a thin chemical layer (called the “interlayer”). Each plane contains parallel molecular wires. The wires in each plane are perpendicular to wires in the other. Various devices, such as switches, diodes, and FETs can be formed at the cross-junctions of nanowires and CNTs. Nanoscale crossbars are attractive for several reasons [7]. It is expected that large-scale nanoelectronics circuits will heavily rely on bottom-up approaches for manufacturing using crossbar structures. Researchers have proposed various high-level architectures based on the crossbar structure [8][9][10].

All architectures at the nanoscale level will face great fabrication challenges that will translate into high defect rate and important parameter variations. As a result extreme levels of random variation and defect rates are major issues that should be addressed in stochastic crossbar-based structures [11][12]. This paper presents a novel method that takes advantage of the reconfigurable nature of the crossbars devices, to mitigate process delay variation and tolerate high defect rates. In this paper, we develop a mathematical model for mapping a logic function onto a nanoelectronics crossbar subject to cost constrained. We then propose a greedy algorithm which successfully maps designs by exploiting the characterization of defect-map and delay-variation of transistors and interconnections if one exists.

The remaining parts of the paper are organized as follows; in section II an overview of variation and defects in crossbar-based nano-architectures is presented. Section III describes the problem formulation. Section IV discusses the proposed greedy logic mapping in detail while section V gets on with the results and analysis. Finally in the last section, we conclude the paper and probable future works in this scope are proposed.

II. DEFECTS AND VARIATIONS IN THE NANOCROSSBAR

Nanoscale crossbars consist of two sets of orthogonal wires. At the cross-point of any two wires is a reconfigurable switch or diode or FET. When the switch is in the “off” state, the wires are insulated from one another and have no contact. In the “on” state, the wires are connected to form either an AND or OR gate. Toggling between the “on” and “off” states

Abstract—High defect density and extreme process variation for nanoscale self-assembled crossbar-based architectures have been expected to be as fundamental design challenges. Consequently, defect and variation issues must be considered on logic mapping on nanoscale crossbars. In this paper, we investigate a greedy algorithm for the variation and defect aware logic mapping of crossbar arrays. Based on Mont-Carlo simulation, we compare the proposed technique with other logic mapping techniques such as variation unaware and exhaustive search mapping in terms of accuracy as well as runtime.

Defect and Variation Issues on Design Mapping of Reconfigurable Nanoscale Crossbars

Behnam Ghavami, Alireza Tajary, Mohsen Raji, Hossein Pedram

Computer Engineering & Information Technology Department
Amirkabir University of Technology, Tehran, Iran
Emails: {ghavamib,tajary,raji,pedram}@aut.ac.ir
is what gives these devices their reconfigurability. Figure 1 provides an example of AND-OR logic nanoscale crossbar.

![Figure 1. An AND-OR logic nano-crossbar with 5-inputs and 2-outputs](image)

In general the feature size scaling reduces control over the fabrication process which results in higher defect and variation rates. Even for well-studied and largely manufactured conventional lithographic systems, the technology is facing reliability challenges. It is expected that circuit designers can no longer design simply by technology design rules and expect a functional, let alone a scalable design. For emerging nano-technologies the defect and variation rate is even higher due to the small feature size and bottom-up nature of the design.

In the manufacturing process of nano crossbars, defects can be introduced to both switches and wires. Defective switches might lose reconfigurable abilities. Two kinds of switch defects are typical: close switches and open switches. Nanoscale wires may suffer from broken and bridging defects in the self-assembly process. Broken wire defects are catastrophic since they make many switches unusable. For bridging defects, only one of the bridging lines can be usable. The broken interconnect or junction defects could be because of lack of deposited molecules, or a too-high resistance interconnect due to lack of proper number of doping atoms, or a misplaced connection between devices because of extra molecule deposition.

Figure 1 depicts the proposed method includes both the generation of the known defects and variation values in the regular structure, and practical logic mapping.  A weighted tripartite graph is a 5-tuple \( \{H, V, E, W, W\} \), where set \( H \) represents the set of horizontal input nanowires, sets \( V \) represent the set of vertical nanowires, sets \( F \) represents the set of horizontal output wires, \( E \) is a set of edges and \( W \) is set of weight correspond to edges. Every edge \( e \in E \) with weight \( w \in W \) connects a node \( n \in H \) or \( F \) with a node \( m \in V \). Indeed, we represent a crosspoint connection between two orthogonal nanowires as an edge in the tripartite graph. We propose a mathematical model to formulate the process of simultaneously variation and defect aware logic mapping on a crossbar-based architecture by translating the process to the matching problem of two bipartite weighted graphs.

### A. Single-Output Nano Crossbar Architecture Model

A weighted tripartite graph is a 5-tuple \( \{H, V, E, W, W\} \), where set \( H \) represents the set of horizontal input nanowires, sets \( V \) represent the set of vertical nanowires, sets \( F \) represents the set of horizontal output wires, \( E \) is a set of edges and \( W \) is set of weight correspond to edges. Every edge \( e \in E \) with weight \( w \in W \) connects a node \( n \in H \) or \( F \) with a node \( m \in V \). Indeed, we represent a crosspoint connection between two orthogonal nanowires as an edge in the tripartite graph. The broken interconnect or junction defects could be because of lack of deposited molecules, or a too-high resistance interconnect due to lack of proper number of doping atoms, or a misplaced connection between devices because of extra molecule deposition.

Hence, variability and defects are key challenges in making nano-crossbars a viable technology. Various works are working on problems related to defect tolerance in nanotechnologies [19][20][21][24]. Given an \( n \times n \) crossbar, an algorithm to find out a \( k \times k \) defect free subset has been proposed in [19][21]. The actual design then can be mapped onto this \( k \times k \) defect free crossbar. A technique to maximize \( k \) has also been presented in [20]. In [24], a mathematical model and algorithm that address the problem of logic function mapping based on defect map in a nanoelectronic environment was presented. Then an enhancement technique improves the algorithm’s runtime by significantly cutting down on unnecessary backtracking processes was introduced. However, few researches are working on problems related to variation aware logic mapping of nano-crossbars. Lately, researchers tried to consider the variation in logic mapping using a characterization of the circuit [22]. Importantly, this approach encountered by some limitations. This technique assume that at least one \( k \times k \) (\( k \) is determined before fabrication) defect free crossbar block must exist after fabrication, hence, will have a drastic effect on the overall yield under high defect density. Further, due to the fundamental goal of using only one \( k \times k \) defect free crossbar block to construct the desired circuit, many defect free blocks may be left unused, which will also have an adverse effect on overall yield. In addition, one disadvantage of this method is that it doesn’t suitable for low-level nanoscale crossbars such as AND-OR and NOR-NOR logics.

This work proposes a novel algorithm to simultaneously variation and defect aware logic mapping on a crossbar-based architecture by translating the process to the matching problem of two bipartite weighted graphs.

III. PROBLEM FORMULATION

We propose a mathematical model to formulate the process of simultaneously variation and defect aware logic mapping on a crossbar-based architecture by translating the process to the matching problem of two bipartite weighted graphs.
In case of a crossbar, various types of crossbar defects can be modeled as follows:

- **Switch stuck-open defects**: A switch stuck-open fault corresponds to a missing switch at the crosspoint. It can be modeled by simply deleting edges corresponding to the defective switches.

- **Switch stuck-closed defects**: If there is a stuck-closed (short) defect on a switch corresponds to an edge, a horizontal nanowire and the vertical nanowire are shorted together and both become unusable. Therefore, we can either generate a mapping with this switch turning on, or use the vertical (horizontal) nanowire and leave the horizontal (vertical) nanowire unused in AND (OR) plane. It can be modeled by adding a constant edge corresponding to this switch.

- **Nanowire open defects**: All switches connected to a nanowire with a broken defect, become unusable. In this situation, the segment that connects to the crossbar input/output can still be used. It can be modeled by deleting edges corresponding to the switches on disconnected segments.

- **Nanowire/tube bridging defects**: In case of a nanowire/nanotube bridging defects, two (or more) nanowires are shorted together. As a result one of them is usable. So new graph is obtained by deleting the two (or more) nodes corresponding to one (or more) of shorted nanowires/nanotubes.

In the proposed weighted tripartite graph model, the variation information for resources within the crossbars is obtained during the characterization testing phase for these devices, in the form of a path delay testing procedure. Since crossbars have a regular structure and they are also reprogrammable, an efficient delay testing technique can be devised for delay variation characterization [22]. This characterization is done such that the delay variation of nanowires/nanotubes, contact, and crosspoints are lumped into delay variation values for individual crosspoints. The lumped delay variation model in the crossbar is represented by weight set $W$. Weight set has elements of real numbers where each entry indicates delay variation ratio of individual crosspoints in the crossbar.

![Figure 2](image2.png)

(a) A 5×5×1 nano crossbar  
(b) Weighted tripartite graph representation of the crossbar

Defect maps of nano crossbars can be obtained through a post-fabrication testing phase due to the regular structure of crossbars. Various types of crossbar defects can be modeled as follows:

- **Switch stuck-open defects**: A switch stuck-open fault corresponds to a missing switch at the crosspoint. It can be modeled by simply deleting edges corresponding to the defective switches.

- **Switch stuck-closed defects**: If there is a stuck-closed (short) defect on a switch corresponds to an edge, a horizontal nanowire and the vertical nanowire are shorted together and both become unusable. Therefore, we can either generate a mapping with this switch turning on, or use the vertical (horizontal) nanowire and leave the horizontal (vertical) nanowire unused in AND (OR) plane. It can be modeled by adding a constant edge corresponding to this switch.

- **Nanowire open defects**: All switches connected to a nanowire with a broken defect, become unusable. In this situation, the segment that connects to the crossbar input/output can still be used. It can be modeled by deleting edges corresponding to the switches on disconnected segments.

- **Nanowire/tube bridging defects**: In case of a nanowire/nanotube bridging defects, two (or more) nanowires are shorted together. As a result one of them is usable. So new graph is obtained by deleting the two (or more) nodes corresponding to one (or more) of shorted nanowires/nanotubes.

In the proposed weighted tripartite graph model, the variation information for resources within the crossbars is obtained during the characterization testing phase for these devices, in the form of a path delay testing procedure. Since crossbars have a regular structure and they are also reprogrammable, an efficient delay testing technique can be devised for delay variation characterization [22]. This characterization is done such that the delay variation of nanowires/nanotubes, contact, and crosspoints are lumped into delay variation values for individual crosspoints. The lumped delay variation model in the crossbar is represented by weight set $W$. Weight set has elements of real numbers where each entry indicates delay variation ratio of individual crosspoints in the crossbar.

![Figure 3](image3.png)

Figure 3. The bipartite graph (FBG) of a logic function

**B. Logic Function Model**

Any two-level single-output logic function in the form of SOP can be modeled by a bipartite graph based on the relationship between its variable set and product set. A set of logic functions can be expressed as a 3-tuple $F(I,P,Y)$, where sets $I$, $P$, $Y$ represent the set of input variables, product terms, and edges, respectively. Figure 3 shows a logic function and its bipartite graph model. We will name this model as $FBG$ in the remaining parts of the paper.

![Figure 4](image4.png)

Figure 4. Converting the Weighted tripartite graph to Weighted bipartite graph (WCBG)

Now, we formulate this logic mapping problem as embedding the FBG graph into the WCBG graph.

![Figure 5](image5.png)

Figure 5. Conversion of the weighted bipartite graph to the weighted tripartite graph

**C. Logic Mapping Model**

The mapping process of a single-output two-level logic function onto a crossbar-based structure determines the implementation of a variable or a product by a specific wire. Thus, the relationship between a variable and the product term pair can be represented by activating the device at a crosspoint in the crossbar. For a given function, there exist several different mappings (i.e. configurations) to the crossbar, i.e. how the inputs and the outputs of the function are assigned to the horizontal and vertical nanowires of the crossbar which makes using different crosspoints possible. This flexibility allows us to efficiently tolerate variations as well as defects.

To formulate the variation and defect aware logic mapping, we first convert the weighted tripartite graph model of the crossbar to a weighted bipartite graph. This is done via back annotate of the node $F$ to the nodes $V$ and summing weight of each edge between node $V$, and node $F (W_{V,F})$ to the weights of edges between node $V$, and nodes $H (W_{H,V})$.

$$W_{H,V} = W_{H,V_i} + W_{V,F} \quad j \in H, i \in V$$

If a node in $V$ doesn’t connect via an edge to node $F$, during converting process, this node in $V$ and all of the associated edges are removed. We will name this resulted graph model as $WCBG$ in the remaining parts of the paper. Figure 4 shows an example of this converting process.
Given a crossbar WCBG(H,V,E,W) and a function set FBG(I,P,Y), the joint variations and defects aware logic problem can be defined as finding a node mapping such that:

$$ SVFLM(\text{FBG, WCBG}) : I \rightarrow H, P \rightarrow V $$

\[ \forall (n,m) \in Y \ | \ (n \in I, m \in P) \ \exists (M(n), M(m)) \in E \ \ \ \ \ \text{SVFLM}^* \ | \ \text{VarCost(SVFLM}^*) < \text{VarCost(SVFLM)} \]

where variation cost (VarCost) is the maximum delay variation associated to the output of a crossbar after logic mapping and is calculated in the proposed model as bellow:

$$ \text{VarCost(SVFLM)} = \max \left( \sum_{i \in \text{SVFLM}(h) \text{and } e_{ij} \notin E} W_{ij} \right) \ \ \ \ \ \forall \ j \in V $$

This problem is an extended version of well-known subgraph isomorphism problem. Subgraph isomorphism is an important and very general form of exact pattern matching [23]. It is well-known, among the different types of graph matching (monomorphism, isomorphism), subgraph isomorphism is an NP-complete problem.

Unlike the straightforward logic mapping onto a defect-free and variation-unaware crossbar, defective and variation aware logic mapping onto a crossbar is no longer trivial and demands to exploit mapping flexibility through permutation of variables and products. When an LBG is of size \( n \times m \) and a WCBG is of size \( N \times M \), the number of all possible mapping trials is \( N!(N-n)!M!(M-m)! \), composing the solution space. The volume of the solution space grows exponentially to the number of nodes in both graphs.

Figure 5 shows an example of a logic function implemented in a nanowire-based crossbar with breaks and variations in crosses. In this example, if we map like case (a), then variation objectives can be calculated as follows.

$$ \text{VarCost}(%_1) = \max \{ (W_{41} + W_{51}), (W_{41} + W_{44}), (W_{25} + W_{55}) \} $$

Another mapping represented by case (b), this new mapping can reduces/increase variation objective compared to the mapping in case (a).

$$ \text{VarCost}(%_1) = \max \{ (W_{22} + W_{52}), (W_{14} + W_{34}), (W_{35} + W_{55}) \} $$

Since each mapping uses a different set of wires, the total variation of each mapping could be different. Our goal is to select one of these possible mappings which minimizes the variation of logic output. There is no known polynomial time algorithm for graph isomorphism. Therefore, we require a heuristic method for this optimization problem.

IV. JOINT DEFECT AND VARIATION AWARE LOGIC MAPPING

In this section, we investigate a method for logic mapping to nano crossbars in order to tolerate variation and defect simultaneously. Figures 6-8 show the proposed recursive algorithm for embedding the logic function bipartite graph FBG into the nanowire crossbar bipartite graph WCBG, such that uses greedy approach to select edges in order to tolerate defects and minimize delay variation at the same time.

Algorithm: SVFLM

Input: BGL, BCWG

Output: SVFLM

1. Begin
2. IF (there is no node in I and A)
3. Return True;
4. ELSE
5. Begin
6. UL_BGL = (set of unmapped nodes from I such that a connected node from A is mapped)
7. While (UL_BGL is not empty)
8. Begin
9. l = a node from UL_BGL;
10. r = a node from A such that is connected to node l;
11. r' = the mapped node to r from BWGC;
12. CBM = the set of unmapped nodes from I connected to r';
13. While (CBM is not empty)
14. Begin
15. 1' = Best-Node(CBM);
16. IF (MAPPER (BGL - l, BWGC - l') )
17. Begin
18. Return true;
19. Delete 1' from CBM;
20. End
21. Return False;
22. End
23. r = an unmapped node from A such that its fan-out is maximum;
24. UR_BWGC = unmapped nodes from V
25. While (UR_BWGC is not empty)
26. Begin
27. r' = Best-Node-2 (UR_BWGC)
28. IF (MAPPER (BGL - r, BWGC - r') )
29. Begin
30. delete r' from UR_BWGC;
31. End
32. End
33. Return False;
34. End
35. End

Figure 6. The variation and defect aware mapping algorithm

Algorithm: Best-Node

Input: CBM

Output: node Y

1. Begin
2. the best node is an unmapped node that the cost of connecting edge from left side to r'...
is minimum

3. End

Figure 7. The Best-Node algorithm

Algorithm: Best-Node-2

Input: UR_BWGC
Output: node Z

4. Begin
5. the best node from BGWC has the maximum priority (the minimum average cost of relating edges)
6. End

Figure 8. The Best-Node-2 algorithm

The above algorithm takes two steps to embed BGF into WCBG. First, it selects the best node (r) from right side of BGF. The best node is an unmapped node has the maximum fan-out. Then it selects the best unmapped node (r') from right side of BGWC that is candidate for mapping of r'. Node r' is a node that has a fan-out more than fan-out of r and is selected with a priority manner. The priority is determined as bellow:

\[
\text{Priority}(n) = \left(1/\left(\sum_{v \in H \text{ and } e_{in}=1} W_{in}/|\text{fanout}(n)|\right)\right)
\]

Using this greedy selection, we first map nodes to lower delay variability edges. This leads to delay variation cost of crossbar will be low in often time (Refer to section V). In a case that this algorithm can’t map r to r', selects another node for r'. If there isn’t a mapping choice for node r', it returns false.

The second step comes when there is an unmapped node in the left side of BGL. In this step, we map a node (l) to an unmapped node (l') from left side of BWGC. We suppose that l is connected to r in BGL and r is mapped to r' in BWGC. The candidates for l' is a set of unmapped nodes from the left side of BWGC that are connected to r'. The best candidate for l' is a node that its edge to r' has the minimum variation cost. If it can’t map l to l', selects another node for l'. If there is no mapping choice for l, it returns false.

It is more important to note that, another restriction condition exist when try to map a node which has one or more edge that associated left-side nodes was previously mapped. In this case, an edge must be exist between the candidate nodes (right-side) and associated mapped nodes (left-side) in BWGC. This condition must be checked in Figure 9.

The algorithm recursively solves the reduced embedding problem with a bipartite graph contains remaining unmapped nodes as input.

V. Experimental Results

In order to evaluate the effectiveness of the proposed method, in terms of accuracy and runtime reduction, we have compared it against the variation-unaware and exact (based on exhaustive search) methods. The variation-unaware method is the same as variation-defect aware one except all of weights of WCBG seem are zero in mapping process. The exhaustive method uses all the permutations to find the exact solution. We evaluate these methodologies by using 3 benchmarks from the LGSynth93 benchmark set [25]. We compare the effects when different defect and variation rates are present. While the naturally random variations are often modeled by Gaussian distributions, we assume independent Gaussian distributions for nano-crossbar delay values. Since exhaustive method is only tractable for very small crossbars, we have presented results for only 8, 12, 12 crossbars. These methods have been implemented in C++ and are run on a computer with a 2.7-GHz CPU and a 2-Byte memory under the Linux operating system.

A. Comparison with exact method

Figures 9 and 10 include the experimental results for the embedding a bipartite graph onto 1000 weighted crossbar bipartite graphs. We analysis variation costs and runtimes through a benchmark. These various crossbars with size of 8 8 was produced based on Mont-Carlo simulation method using defect rate 10% and variation rate 100%. Figure 9 gives the corresponding variation cost for various crossbars for 3 mapping algorithms. As it present, the proposed method is closed to exact method.

Figure 9. variation cost for tree mapping algorithm (Red:Exact Method, Blue: Proposed Greedy method, Yellow: Variation un-aware mapping)

Figure 10. Runtime expectation for different algorithm (Red: Exact Method, Yellow: Proposed Greedy method, Blue: Variation un-aware mapping)

Figure 10 gives the corresponding run time. As expected, the proposed mapping method has a great improvement in runtime compared to exact method.

B. Variation rate issue

Table I summarizes the experimental results of solving mapping problems at different variation rates. We randomly generate 1000 crossbars with delay variation ratio of 50% and defect rate 15%. This process was repeated for 90%, 120% and 200% variation ratio and the same defect rate for other
three crossbars. The sizes of the crossbars are 12*12 related to the size of the benchmark circuit. The actual run time is shown under rows time. Compared with the exact method our method can be find good answer yet with great improvements in runtime.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameter</th>
<th>Variation ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>Exact Mapping</td>
<td>Mean of Delay Variation Cost</td>
<td>734</td>
</tr>
<tr>
<td></td>
<td>Mean of Runtime (s)</td>
<td>1565</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>Mean of Delay Variation Cost</td>
<td>993</td>
</tr>
<tr>
<td></td>
<td>Mean of Runtime (s)</td>
<td>2.03</td>
</tr>
</tbody>
</table>

C. Defect rate issue

Figure 12 presents the results of mapping problems at different defect rates. We randomly generate defects. The size of the crossbar is 12*12. As the defect rate increases, the constraints for a feasible mapping become large. The probability of a feasible mapping under a given defective crossbar is directly related to the defect rate. Finding a mapping solution or proving that no solution exists may require longer computation time.

VI. CONCLUSION AND FUTURE DIRECTIONS

Although the nanoelectronic technologies evolving a rapid clip towards practical realization of nanoelectronic computing systems, high defect and variation ratio due to nondeterministic assembly at nanoscale is still the major issues that should be addressed. This paper identifies a new challenge of mapping a logic function onto a nano crossbar under the variation and defect issues. We formulated this difficulty using sub weighted graph isomorphism problem and introduce a greedy algorithm to resolve it. Simulation results show that the proposed technique can achieve high yield in a reasonable time. The future research directions include the decomposition of large logic functions onto the crossbar architectures and an incremental bipartite graph creation integrated with the defect and variation ratio considerations.

REFERENCES


