

January 01, 2010

## Variation and defect tolerance for nano crossbars

Cihan Tunc  
*Northeastern University*

---

### Recommended Citation

Tunc, Cihan, "Variation and defect tolerance for nano crossbars" (2010). *Electrical and Computer Engineering Master's Theses*. Paper 44.  
<http://hdl.handle.net/2047/d20000901>

This work is available open access, hosted by Northeastern University.

# Variation and Defect Tolerance for Nano Crossbars

A Thesis Presented

by

**Cihan Tunc**

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements  
for the degree of

**Master of Science**

**in**

**Electrical and Computer Engineering**

Northeastern University

Boston, Massachusetts

May 2010

Northeastern University

# *Abstract*

Department of Electrical and Computer Engineering

Master of Science in Electrical and Computer Engineering

by Cihan Tunc

With the extreme shrinking in CMOS technology, quantum effects and manufacturing issues are getting more crucial. Hence, additional shrinking in CMOS feature size seems becoming more challenging, difficult, and costly. On the other hand, emerging nanotechnology has attracted many researchers since additional scaling down has been demonstrated by manufacturing nanowires, Carbon nanotubes as well as molecular switches using bottom-up manufacturing techniques. In addition to the progress in manufacturing, developments in architecture show that emerging nanoelectronic devices will be promising for the future system designs. Using nano crossbars, which are composed of two sets of perpendicular nanowires with programmable intersections, it is possible to implement logic functions. In addition, nano crossbars present some important features as regularity, reprogrammability, and interchangeability. Combining these features, researchers have presented different effective architectures.

Although bottom-up nanofabrication can greatly reduce manufacturing costs, due to low controllability in the manufacturing process, some critical issues occur. Bottom-up nanofabrication process results in high variation compared to conventional top-down lithography used in CMOS technology. In addition, an increased failure rate is expected. Variation and defect tolerance methods used for conventional CMOS technology seem inadequate for adapting to emerging nano technology because the variation and the defect rate for emerging nano technology is much more than current CMOS technology. Therefore, variations and defect tolerance methods for emerging nano technology are necessary for a successful transition.

In this work, in order to tolerate variations for crossbars, we introduce a framework that is established based on reprogrammability and interchangeability features of nano crossbars. This framework is shown to be applicable for both FET-based and diode-based nano crossbars. We present a characterization testing method which requires minimal number of test vectors. We formulate the variation optimization problem using Simulated Annealing with different optimization goals. Furthermore, we extend the framework for defect tolerance. Experimental results and comparison of proposed framework with exhaustive methods confirm its effectiveness for both variation and defect tolerance.

# *Acknowledgements*

I would like to thank all the people who helped this work possible by their valuable advice and support.

First of all, I would like to thank my advisor, Prof. Mehdi Baradaran Tahoori, for his precious help and support during my research as well as his encouragement. I especially would like to thank him for his frequent view of my work and helping me even during my brainstorming. By his guidance, I learnt a lot about being an independent researcher. Without his continuous support, this work could never been succeeded.

I would also like to thank to our Dependable Nano-Computing Lab (DNL) group at Northeastern University. During this study, they have put their valuable ideas and help which moved this study one step further. In addition, I would like to thank to other professors at Northeastern University who enlightened me with their valuable ideas.

In addition, I warmly thank to Prof. Miriam Leeser and Prof. Gunar Schirner for being in my committee.

Moreover, I am deeply grateful to my undergrad advisor, Prof. Fatih Ugurdag, for his advices that triggered me to be a researcher and an engineer.

Last but not least, of course, I would like to thank to my beloved family. With their continuous support and help, I find the power to succeed and without their love it was impossible to have any achievements.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Emerging Nano Technology</b>	<b>5</b>
2.1 Background . . . . .	5
2.1.1 Nanowire Based Devices . . . . .	5
2.1.2 Crossbar Structures . . . . .	7
2.1.3 Crossbar Based Architectures . . . . .	8
2.2 Issues with the Emerging Nano Technologies . . . . .	11
2.3 Related Work . . . . .	14
<b>3 Delay Modeling and Characterization Testing of Nano Crossbars</b>	<b>18</b>
3.1 Definitions . . . . .	18
3.2 Delay Models for Nano Crossbars . . . . .	21
3.2.1 Delay Models for Diode Based Crossbars . . . . .	21
3.2.2 Delay Models for FET Based Crossbars . . . . .	22
3.2.3 Optimization Goals . . . . .	23
3.3 Characterization Testing for Crossbars . . . . .	24
3.4 Modeling Defects in Crossbars . . . . .	26
3.5 Modeling Crossbar Arrays . . . . .	27
3.5.1 Diode Based Crossbar Arrays . . . . .	28
3.5.2 FET Based Crossbar Arrays . . . . .	28
3.5.3 2D Crossbar Arrays . . . . .	29
<b>4 Algorithms</b>	<b>31</b>
4.1 Exhaustive Search . . . . .	31

---

4.2	Simulated Annealing . . . . .	32
4.2.1	Concept . . . . .	32
4.2.1.1	Simulated Annealing for Diode Based Crossbars . . . . .	34
4.2.1.2	Simulated Annealing for FET Based Crossbars . . . . .	34
4.2.2	Moves . . . . .	35
4.2.3	Efficient Delta Cost Calculation . . . . .	36
4.2.3.1	Efficient Delta Cost Calculation for Diode Based Crossbars . . . . .	36
4.2.3.2	Efficient Delta Cost Calculation for FET Based Crossbars . . . . .	36
4.2.3.3	The Complexity of the Efficient Delta Cost Calculation . . . . .	37
4.2.4	Defect Tolerance . . . . .	37
4.3	Extension for the Crossbar Arrays . . . . .	38
<b>5</b>	<b>Experimental Studies</b>	<b>40</b>
5.1	Experimental Setup . . . . .	40
5.2	Experimental Study for FET Based Crossbars . . . . .	41
5.3	Experimental Study for Diode Based Crossbars . . . . .	45
5.4	Experimental Study for Crossbar Arrays . . . . .	48
<b>6</b>	<b>Conclusions</b>	<b>50</b>
	<b>Bibliography</b>	<b>52</b>

# List of Figures

2.1	Crossed Si doped nanowire junctions. (A) Typical electron microscope image of a crossed Si doped nanowire junction with Al/Au contacts. (B though D) I-V behavior of p-n, p-p, and n-n junctions, respectively [1]. . . . .	6
2.2	Crossed nanowires to obtain a Si doped nanowire bipolar transistor. (A) presents a schematic illustration and in (B) electron microscope image is presented. In (C) and (D), the data representing the current and voltage connection is demonstrated [1]. . . . .	7
2.3	A nano crossbar using bistable junctions. (A) shows the physical representation whereas (B) demonstrates circuit scheme [2]. . . . .	8
2.4	CMOL architecture [3]. . . . .	9
2.5	The layout of the nanoFabric as well as the schematic of a nanoBlock [4].	10
2.6	An overall architectural view for the nanoPLA [5]. . . . .	10
2.7	NASIC architecture [6]. . . . .	11
2.8	An inverter implementation using n- and p- type doped crossbars followed by a switch plane [7]. . . . .	11
3.1	The implementations of two different multi output logic functions, which are based on the same FM, for diode based crossbar (A) and FET based crossbar (B). . . . .	19
3.2	The implementation of the same logic functions using a different configuration. . . . .	21
3.3	The basic idea of implementing a crossbar array using crossbars. . . . .	27
3.4	2D crossbar arrays . . . . .	30
3.5	2D crossbar arrays with logic blocks and connection blocks . . . . .	30
5.1	The histogram of costs for a $6 \times 6$ FET based crossbar. . . . .	42
5.2	The histogram of costs (Objective 1) for a $16 \times 16$ FET based crossbar.	43
5.3	The histogram of costs (Objective 2) for a $16 \times 16$ FET based crossbar.	43
5.4	The comparison of runtime and cost reduction (for Optimization 1) for various sizes of crossbars. . . . .	44
5.5	The histogram of costs for a $6 \times 6$ FET based crossbar. . . . .	46
5.6	The histogram of costs (Objective 1) for a $16 \times 16$ diode based crossbar.	47
5.7	The histogram of costs (Objective 2) for a $16 \times 16$ diode based crossbar.	47
5.8	Runtime and cost reduction (for Optimization 1) comparison for various sizes of diode based crossbars. . . . .	48

# List of Tables

3.1	Function Matrix corresponding to the functions in Figure 3.1. . . . .	19
3.2	An example of a VM for a $4 \times 4$ crossbar. . . . .	20
3.3	Different mapping the same multi-output logic function . . . . .	21
3.4	Representing defects in VM . . . . .	26
3.5	VMs and configurations for the cascaded crossbars. . . . .	28
3.6	Adjusted VM for the diode based crossbar $i + 1$ ( $3 \times 2$ ). . . . .	28
3.7	Adjusted VM for the FET based crossbar $i + 1$ ( $3 \times 2$ ). . . . .	29
4.1	Perturbation only in input mapping vector (IMV) . . . . .	35
4.2	Perturbation only in output mapping vector (OMV) . . . . .	35
5.1	Comparison of Variation unaware mapping (random, RAND), exact method (EXH), and Simulated Annealing (SA) . . . . .	41
5.2	Constrained vs. unconstrained optimizations . . . . .	42
5.3	Success rate in defect-free mapping for diode based crossbars . . . . .	45
5.4	Comparison of Variation unaware mapping (random, RAND), exact method (EXH), and Simulated Annealing (SA) . . . . .	46
5.5	Constrained vs. unconstrained optimizations . . . . .	46
5.6	Success rate in defect-free mapping for diode based crossbars . . . . .	48
5.7	Cost comparison for crossbar arrays (10 $16 \times 16$ crossbars are cascaded) . . . . .	49

*To my beloved family...*

# Chapter 1

## Introduction

While Complementary Metal-Oxide-Semiconductor (CMOS) based structures are scaling down based on Moore's Law, challenges are getting more crucial due to quantum effects and manufacturing issues [8]. Higher performance results in more power dissipation [9] and low supply voltages bring parasitic issues [10]. Smaller size, that requires high doping, results in parasitic capacitance problems [9] as well as direct tunneling because of reduced oxide thickness for smaller gate length [11]. Moreover, smaller lithography requires more complex tools where manufacturing process gets too expensive [10]. The semiconductor industry has overcome with the similar issues so far using *top-down* methodology with the high controllability of the devices [10] where the manufacturing starts from Silicon and continues by adding layers using lithography [12]. However, for top-down lithography, it is extremely difficult and expensive to control nanoscale structures [13]. Consequently, emerging technologies with nanowires [14] and Carbon nanotubes [15] have been presented using *bottom-up* techniques (where the devices are manufactured first and then assembled) to achieve more scaling [13].

As an alternative approach for top-down manufacturing, bottom-up approach is considered where materials are created using chemical assembly instead of lithography (top-down manufacturing) [16]. This manufacturing includes methods such as Langmuir-Blodgett films, flow-based alignment, random assembly, biologically assisted assembly, and catalyzed growth [17]. Using bottom-up techniques, molecular switches [18], nanowires [14], and Carbon nanotubes [15] have been presented. In proposed nano materials, nanowires seem to be more promising than others since they can be doped with Silicon (Si) or Germanium (Ge) [19], [20], [21], [22]. Using nanowires, it is shown that interconnections [5], p-n-diode rectifiers by doping with

Silicon [23], Field-Effect transistors (FET) [1], [22], and logic gates [1], [24] can be built.

Structures built using bottom-up approach, as the building blocks for molecular-scale computing, are by their nature very regular and therefore well suited to the implementation of regular arrays similar to conventional *Field Programmable Gate Arrays* (FPGAs) [2], [3], [4], [25]. The main building block of nano architectures, nano crossbars, consists of two sets of perpendicular nanowires. For diode based crossbars, each intersection of these nanowires contains a programmable non-volatile diode [2] that can be (re)programmed as ‘on’ or ‘off’ by applying a different voltage [24], [26], [25]. In addition to diode based crossbars, FET based crossbars can be built using Silicon doped nanowires at the bottom set and metallic wires at the top set [7] where doped nanowires are oxidized to prevent direct connection from metallic ones to show p-n-junction behavior [22]. These structures have been considered configurable by controlling the charge or polarization of the individual junctions [13]. Thus, it is possible to use nano crossbar arrays when post-fabrication customization is needed. Using diode based crossbars, architectures like CMOL [3], NanoFabric [4], nanoPLA [27], etc. have been proposed. Moreover, using FET based crossbars, NASIC architecture [6] as well as the idea of implementation of configurable complementary n- and p- type FET arrays followed by a switch array [7] have been proposed.

While bottom-up method is useful for reducing top-down manufacturing cost, it limits to manufacture only basic, regular [8] and stochastic structures [28]. Hence, for nanowire-based structures with low control during the manufacturing process, defects and variations are two major issues that should be addressed [27]. Open or shorted nanowires as well as defects in crosspoints (stuck-open or stuck-short) are major issues [29], [25], [30]. The defects for a nano crossbar is expected much higher than current CMOS technology. For example, 10% defect rate for crossbars has been reported [31]. This means that 10% of the crosspoints will be unusable for that crossbar structure. Therefore, defect tolerant methods are necessary for the future systems.

Furthermore, also variations for emerging nano technologies should be tolerated since the variations for the new technology is expected to be much higher than the current CMOS technology. There are various sources of variations in the characteristics of nano devices. Due to lack of control during the manufacturing process, the length of nanowires may vary as well as the thickness. While resistance and capacitance are

based on the length and thickness of a wire, variation in resistance, capacitance, and also in inductance will apply [32], [33]. Even though nanowires can be doped with Si or Ge, at this atomic level doped region may not be fully controlled. Therefore, the resistance will not be fully determined and large variations may occur [25], [34]. In addition to length, for a nanowire based FET, also field effect regions as well as core shell thickness vary from device to device due to bottom-up statistical alignment process [34], [28]. In addition to FETs, while for diodes, diode region is composed of a small number of elements or bonds extreme random variation from crosspoint to crosspoint will be seen. Furthermore, connection resistances (and capacitance) between microwires and nanowires is another source of variation [35]. And, intersection resistance could be a limiting factor for the performance of this nanotechnology which cannot be fully determined [36]. Last but not least, environmental issues such as temperature gradient may cause resistance variations in nano structures [37]. In addition to random variations, variations due to fanout parameter may have significant range affecting charging and discharging of a circuit [34].

While defects result in useless crosspoints and useless nanowires, variations will affect the performance since with high variations in resistances and in capacitances, nano architectures will not meet timing constraints [25], [32], [35].

Due to high defect rate and extreme variation, it is necessary to have method for building systems immune to these issues. Therefore, in this work, we try to focus on mapping techniques to tolerate variation as well as defect. We present a variation-aware logic mapping technique for nano crossbar arrays (for both diode and FET based crossbars) to tolerate variations which are considered as delay differences of individual crosspoints. We take advantage of reprogrammability and interchangeability of nano architectures to be able to map the function while tolerating (delay) variations. Since there are different mappings of a function to the crossbar array, we try to find the one resulting in minimum variation (e.g. delay differences). We also extend this framework from crossbars to the crossbar arrays. Moreover, we revisit this problem with defect tolerance requirements.

In the next chapter (Chapter 2), we review the emerging nano technologies. We first talk about nano devices. Then, different structure based crossbars are investigated. Next, the architectures using nano crossbars are summarized. Furthermore, previous work on defect and variation tolerance have been mentioned.

In Chapter 3, delay modeling for both FET and diode based crossbars are introduced. Then, the necessary characterization testing methods are explained. In addition, we

extend the proposed method for defect tolerance. Last, the extension of the proposed method for crossbar arrays are explained.

Next, Chapter 4 presents the proposed algorithm both variation and defect tolerance. We also analyze the runtime of the algorithms in this chapter. Then, in Chapter 5, the experimental results are presented which show the effectiveness of the proposed framework.

Finally, Chapter 6 concludes this thesis.

# Chapter 2

## Emerging Nano Technology

This chapter provides an overview of the emerging nano electronics, focusing on the devices, crossbars, and architectures. In addition, the challenges in this technology and some related work for tolerating these issues are also mentioned.

### 2.1 Background

#### 2.1.1 Nanowire Based Devices

It has been shown that Carbon nanotubes and nanowires can be used for more than interconnection [17]. Among emerging nano devices, nanowires seem to be very promising due to the fact that it is possible to control the carrier type and the concentration during growth [1], [19], [20], [21], [22]. Thus, it is possible to build p-type or n-type doped semiconductor nanowires.

Using semiconductor nanowires, researchers have presented diodes and FETs based on p-n junction behavior where oxide can be grown to avoid contacting [23], [1], [22]. FETs are also used to manufacture logic gates based on conventional Silicon technology [20]. An example study by Cui and Lieber [1] shows the implementation of nanowire based diodes and FETs using nanowire based emerging technology.

Two semiconductor nanowires, one p-type and the other one n-type, can be used to form a junction diode at their crossing. Since two crossed nanowires may have a short circuit, oxide is grown on the nanowires by applying high current to flow through which is used for heating the junction to get oxidized from the air [38].

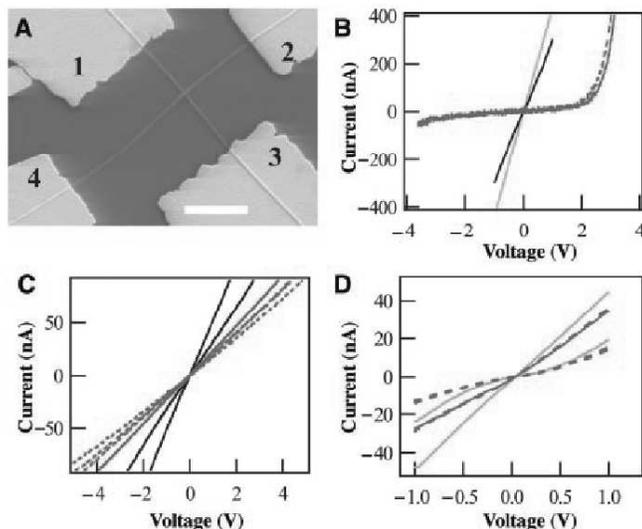


FIGURE 2.1: Crossed Si doped nanowire junctions. (A) Typical electron microscope image of a crossed Si doped nanowire junction with Al/Au contacts. (B through D) I-V behavior of p-n, p-p, and n-n junctions, respectively [1].

It should be noted that in the Figure 2.1, (B) demonstrates a p-n junction that shows a diode like behavior. As conventional CMOS based diodes, the nanowire based crosspoint diodes allow current flow through after a certain threshold voltage such as  $I = I_S(e^{(V_D/V_T)} - 1)$ .

It has been shown that by using the p-n-junctions for crosspoint diodes, diode arrays have been made, with 85% to 95% yield, where each of them shows an independent operation [38].

While two doped nanowires are used to create a junction diode, three nanowires with adequate crosspoints can be used to produce a three terminal device (i.e. FETs) as shown in Figure 2.2 [1]. In (C), the base-emitter voltages are shown with collector-base voltage versus collector current. In (D), common base current gain versus collector-base voltage is presented. It can be seen that Si doped nanowire based bipolar transistors exhibit very good current gain. The observations gained with this study show that nanowire based transistors may be used in the future systems instead of CMOS technology based transistors.

In addition to the nanowire based devices (diodes and FETs), molecular devices have been presented. Molecular resonant tunneling diodes have been presented where these diodes show negative differential resistor (NDR) behavior [39]. The main importance of these devices is that they show a negative resistance for a region of

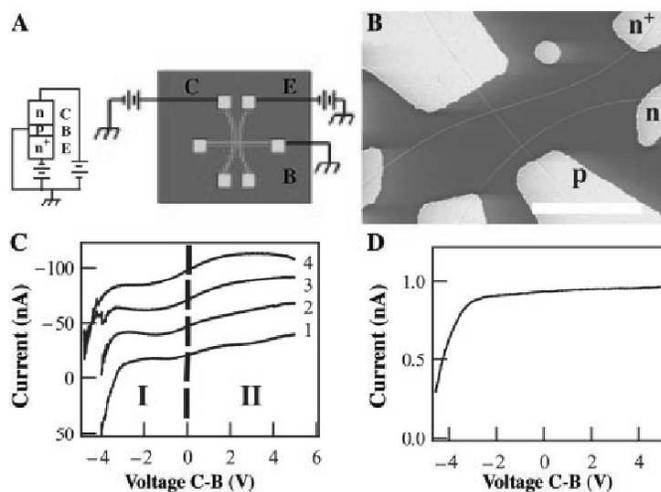


FIGURE 2.2: Crossed nanowires to obtain a Si doped nanowire bipolar transistor. (A) presents a schematic illustration and in (B) electron microscope image is presented. In (C) and (D), the data representing the current and voltage connection is demonstrated [1].

their I-V curves. By the appropriate usage, it is possible to obtain molecular latches which can also be used for signal restoration as well as I/O isolation.

In addition to NDRs, organic molecules have been presented. The main idea of these devices is that they are combined of mechanically distinct parts, such as a ring and a rod or interlocking rings. By applying a programming voltage across the molecule adds or subtracts an electron (oxidation-reduction), shifting the ring and changing the molecules conductivity. Also, the molecule saves its state which in turn can be used as a non-volatile programmable molecular switch [17]. The most known examples are catenane and rotaxane molecules.

Last but not least, it is possible to use Carbon nanotubes for mechanical switch behavior. Arranging two Carbon nanotubes as a crossbar where the upper half and the lower one are distinct, it is possible to program them by applying voltage so that they attract each other and by the help of Van der Waals force they keep their states [40].

## 2.1.2 Crossbar Structures

Using the devices mentioned above, crossbar structures have been built for logic function implementation. For example, researchers have demonstrated using rotaxane molecules (with the feature of resistances changing in different states) between

two perpendicular nanowires in order to build nano crossbars for memory, logic blocks and programmable interconnect [24], [23], [41]. Configuring the nano crossbar arrays can be achieved by programming crosspoints with applying voltage difference [42], [26], [25] where the junction can be addressed by two nanowires (horizontal and vertical). An example of a nano crossbar is shown in Figure 2.3.

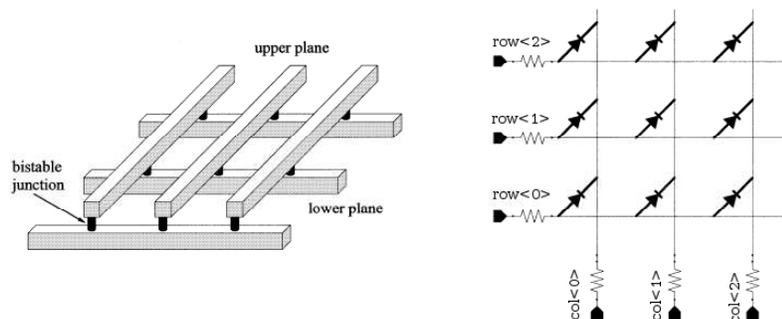


FIGURE 2.3: A nano crossbar using bistable junctions. (A) shows the physical representation whereas (B) demonstrates circuit scheme [2].

In addition to switch based crossbars, FET based crossbars can be built using Silicon doped nanowires at the bottom set and metallic wires at the top set [26]. Doped nanowires are oxidized to prevent direct connection from metallic ones to show p-n-junction behavior [22]. These structures can be customized by using decoders to move the desired crosspoints into a close position (activation) for FET behavior or a separate position (deactivation) for regular wire behavior [23].

### 2.1.3 Crossbar Based Architectures

The nano crossbars have been considered as the main building block of the future architectures like NanoPLA, nanoFabrics, CMOL (a hybrid architecture), complementary n- and p- type arrays since nano crossbars are very regular, reconfigurable, and interchangeable. The proposed architectures aim using nano crossbars for the logic function implementation which can be considered as the heart of the architectures. Following architectures are the most known and most accepted architectures among the proposed ones.

A hybrid architecture called CMOL uses diode based nano crossbar arrays on the top of CMOS cells where integration between micro and nano blocks are achieved on the same level [3]. Using pins between CMOS and crossbar arrays and turning crossbar array by some angle  $< 90^\circ$ , it is shown that it is possible to access each nanowire

even though they are not precisely aligned as shown in Figure 2.4. Therefore, even though the expected defect rate is extremely high, it will be possible to use this hybrid architecture for future systems.

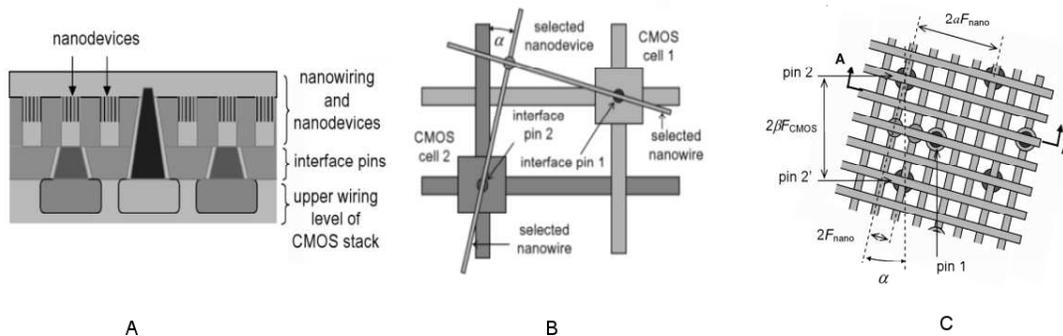


FIGURE 2.4: CMOL architecture [3].

For CMOL (Figure 2.4), on the left, the schematic view is presented. The pyramid like pins are considered for the interface between CMOS and nano crossbars. Since nano crossbars are expected in smaller size, it will be easier to apply pyramid like pins. In the middle, the addressability feature of a particular nanodevice is shown. It should be noted that by choosing two nanowires, any nanodevice can be chosen. In addition, on the right of the figure, the overall view of the addressability using any pins (as an example pin 1 and pin 2) are presented.

The nanoFabric architecture, shown in Figure 2.5, uses the idea of today's FPGAs where logic blocks (nano blocks) are routed using switch blocks [4]. Each nano block contains a *molecular logic array* (MLA) which is based on diode-resister logic (RDL). Since RDL suffers from voltage degradation, restoration is necessary. For the nano blocks, restoration is achieved by the molecular latches that are orthogonal to output wires.

Another nano architecture, NanoPLA (Figure 2.6), contains nano crossbar arrays (2D diode based crossbars) for logical operations and uses Silicon doped nanowires for addressing, restoration, and inversion [23, 25, 27, 43]. For the addressing of nano crossbars, stochastic address decoders are used which are composed of doped nanowires. Since lightly doped regions will be sensitive to inputs, address decoders can be built. The outputs of programmable nano crossbars are restored with restoration plane using nanowire based FETs. In addition, these planes can be used for inversion so that any universal logic functions can be built.

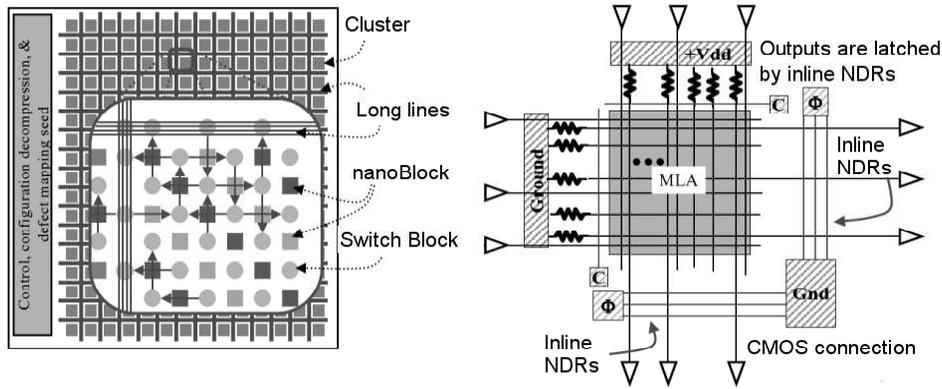


FIGURE 2.5: The layout of the nanoFabric as well as the schematic of a nanoBlock [4].

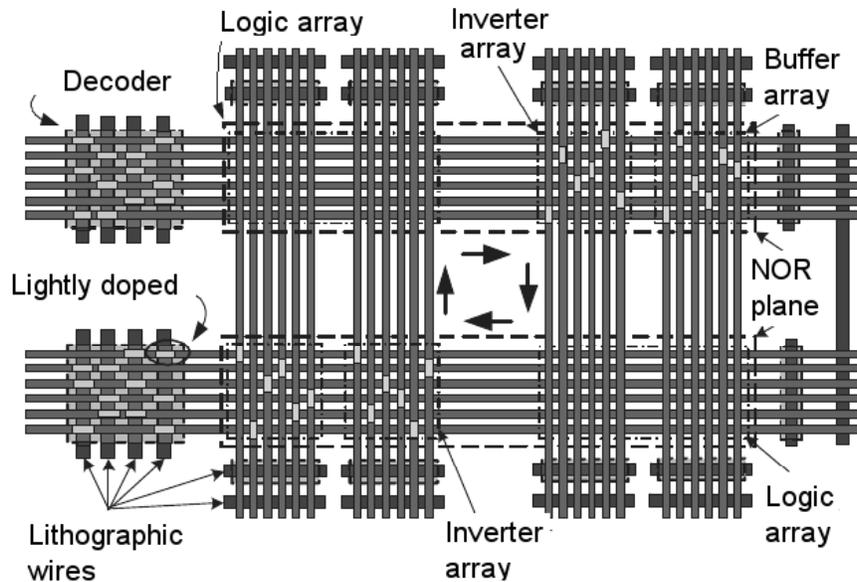


FIGURE 2.6: An overall architectural view for the nanoPLA [5].

In addition to the diode based crossbars, FET based crossbars are also recommended for the usage of the logic functions as the main building block of some architectures. For example, it is also suggested to use FET based crossbars for NOR planes of crossbars [5]. Moreover, another architecture based on FET crossbars is NASIC [6] as shown in Figure 2.7.

For the NASIC architecture, using dynamic circuits built with nanowire FETs, different logic circuits can be obtained (the left figure) and by combining these circuits any logic functions can be implemented [6]. The main reason of using nanowire FETs is to build more tuned nano architectures under application dependent domain. Hence, they try to reach denser designs with better utilization, efficient cascading, and better routing [44].

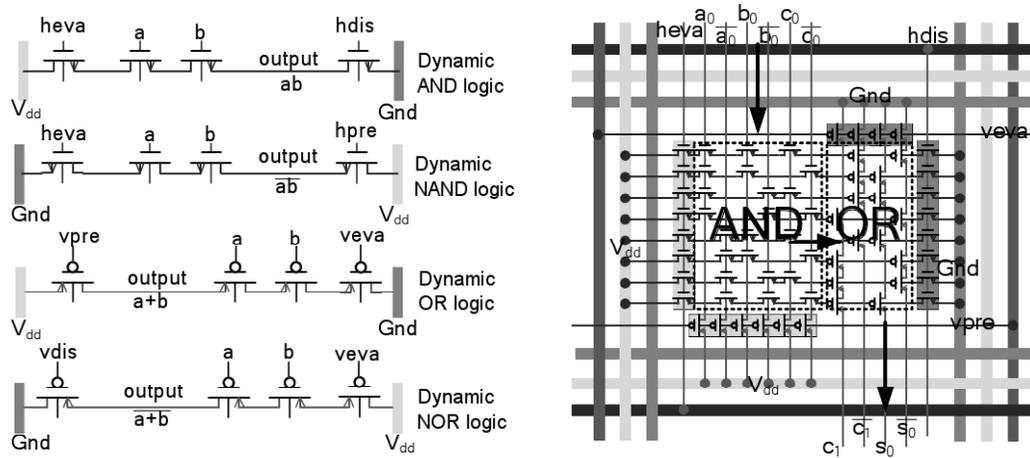


FIGURE 2.7: NASIC architecture [6].

Last but not least, using both switch based and FET based crossbars, another architecture is presented. In this architecture, p- and n- FET reconfigurable crossbars are used for pull up and pull down networks with the help of switch based crossbars [7]. An example for an inverter is shown in Figure 2.8. In this example, the pull-up transistor is gathered by the left plane (p- type transistor plane) and the pull-down transistor is obtained using the right plane (n- type plane). Then, these two transistors are connected to each other using switch based crossbar plane.

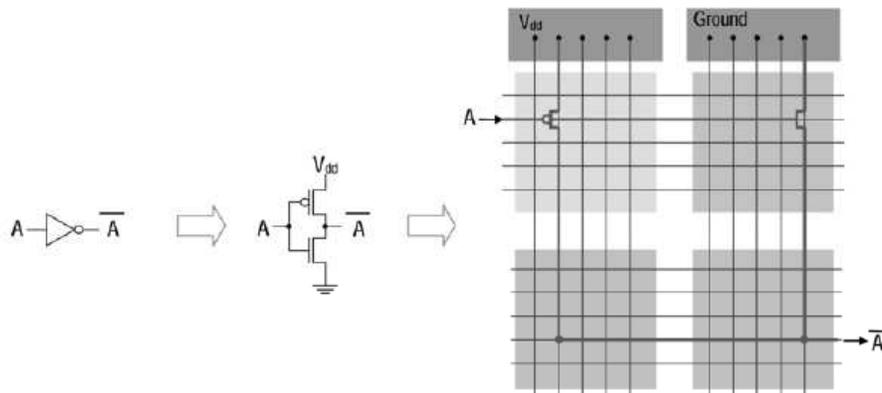


FIGURE 2.8: An inverter implementation using n- and p- type doped crossbars followed by a switch plane [7].

## 2.2 Issues with the Emerging Nano Technologies

While bottom-up manufacturing approach is useful for additional shrinking, it fails the control of each device during the manufacturing process. Therefore, due to

the lack of the control during the manufacturing, nanowires that are grown using bottom-up techniques may be broken or misaligned. Hence, these nanowires will become unusable. Additionally, for a crossbar based architecture, crosspoints may contain defects which make them unusable. As a result, the defect rate expected for emerging nano technology is much higher than current CMOS technology. The effects of nanowire and crosspoint defects can be shown with the following faults [45].

**Stuck-open crosspoints faults:** A stuck-open fault for a crosspoint corresponds to a missing device at that crosspoint. Therefore, it will be impossible to use (e.g. activate) the crosspoint.

**Stuck-closed crosspoint faults:** If there is a stuck-closed fault for a crosspoint, this crosspoint will behave as a short circuit and the intersecting nanowires (i.e. both horizontal and vertical nanowires) will be shorted. The corresponding crosspoints will not be programmable and cannot be deactivated. Therefore, both horizontal and vertical nanowires become unusable.

**Nanowire open fault:** Broken nanowires will not be able to carry signals. Hence the entire nanowire and the devices connected to it will be malfunctioning and should be omitted during the design phase.

**Nanowire bridging fault:** In case of a nanowire bridging fault, two (or more) nanowires are shorted together meaning that both (all) of them become unusable.

It should be noted that all defects in a crossbar can be represented by using only crosspoints defects as Stuck-open and Stuck-closed.

Extreme process variation is one of the major challenges in emerging nano technologies. While current CMOS technology is facing issues with variations due to doping, annealing process, oxide thickness, etc. as well as photolithographic issues, nanowire based architectures are affected by the additional sources of random variation [34]. The sources of variations for emerging nano technologies can be considered as follows.

Due to lack of control during the manufacturing process, variation in the length of nanowires will occur as well as the thickness. Since resistance and capacitance is based on the length and thickness of a wire, variation in resistance, capacitance, and also inductance will apply [25].

As mentioned before, one of the main advantages of nanowires is they can be doped using Si or Ge. However, at this nano regime and low controllability, the doped

region of nanowires may not be fully controlled. Therefore, even though there are some small fluctuations in doping of nanowires, fluctuations in the electrical characteristics of each nanowire will be expected [34]. For example, it has been shown that nanowires, with 3-nm diameter, contain approximately one dopant atom per nanometer of length [46]. When the doping concentration is taken into account, even a single impurity has a substantial contribution to the total electrostatic potential. Therefore, when the the doping has a large variation, the doped nanowires can be useless [25].

In addition, for nanowire FETs, field effect regions as well as core shell thickness vary from device to device due to bottom-up statistical alignment process [34], [28]. In addition to FETs, while for diodes, diode region is composed of a small number of elements or bonds, extreme random variation from crosspoint to crosspoint will be seen.

It should be noted that connection resistances (and capacitance) between microwires and nanowires is another source of variation [35]. And, intersection resistance could be a limiting factor for the performance of this nanotechnology which cannot be fully determined [36]. For example, a programmable molecule composing crosspoint for a crossbar may have extreme resistance whereas for another programmable molecule, the resistance may be lower. Further, environmental issues such as temperature gradient may cause resistance variations in nano structures [37].

In addition to random variations, [34] focuses on also variations due to fanout parameter which may have significant range. Fanout in a NanoPLA is due to the fact that when NAND term outputs are needed to be multiplied, the input wire must have the associated diodes programmed to connect to the required output wires. Also, they must charge the output wires resulting in being affected by the capacitances.

To sum up, due to low controllability in bottom-up self-assembly fabrication, nanowires will contain high variations in resistance, capacitance and inductance as well as in the threshold voltage of diodes and FETs [32], [33] [34]. The effect of the fluctuations in electrical characteristics of a nanowire and nanowire based devices can be explained as follows.

Nanowires with excessively high resistance due to doping variations or poor contacts will not be able to meet timing constraints when they are used for pull-up or pull-down networks [25]. Self capacitance and coupling capacitances will affect the

performance of these devices since capacitance is an important factor in performance (due to charging and discharging) [32], [35].

Since nanowires are doped using Si, conventional MOSFET current equations should still hold. Therefore, it is possible to say that saturation current is linear for threshold voltage and supply voltage, but exponential for cut-off region in a circuit [34]. Therefore, variations in threshold voltage will make the current may not be fully estimated.

Moreover, the variation in threshold voltage of transistors can be modeled as the fluctuations of the resistances as  $R_{off}$  when a FET is in cut-off region and  $R_{on}$  when in saturation region. When the nanowire FET is active, switching time as well as time for discharging are dependent on  $R_{on}$  and  $R_{off}$  which can be used in the modeling of the transistors to show the effect of the variation [34].

## 2.3 Related Work

Many studies for nano architectures have been presented for defect tolerance. Researchers took the advantage of reprogrammability and interchangeability of nano crossbars as well as spare resources to tolerate defects. With the inspiration from Teramac study by HP Labs., where high defect rate is overcome using the reconfigurable structures [47], researchers developed methods for tolerating high defect rate for crossbars with the reconfigurability, abundance of sources, interchangeability features.

A defect aware design flow (application dependent design with defect map in every design level) is provided by [48]. In this study, the reconfiguration feature of crossbars is used to be able to find a defect free mapping. Defective crosspoints are tried to be mapped to defect free crosspoints. Finding a defect free mapping is stated as ‘Bipartite Matching’. However, since using an exact method will require long time for finding an appropriate mapping for defect tolerance, in addition to the exact algorithm, a heuristic algorithm is presented. Using the method, they propose that they can tolerate defects upto 20%.

In [45], the idea of using smaller possible crossbars in a larger crossbar is presented for defect tolerance. While a crossbar can be represented by bipartite graph representation, maximum flow is used in order to tolerate defects. Moreover, it is

also shown that the effect of stuck-closed faults is much higher than the stuck-open faults. Therefore, for increasing the yield, it is recommended to develop manufacturing methods to bias the crosspoints in a way that the possibility of having stuck-closed faults reduce.

A defect unaware design flow, where defect map is only needed at the final mapping process, is presented in [49], [29]. Finding a maximum biclique for defect tolerance was discussed and a heuristic method was presented for runtime reduction. Moreover, a design independent scheme in which a defect-free subset of fabricated resources are extracted and used in the design flow. Furthermore, techniques to reduce the area overhead of the proposed defect-tolerant flow are presented.

A mathematical model for mapping crossbars is proposed in [50]. They additionally improve their study with a heuristic defect tolerant mapping method based on bipartite graph.

Moreover, [51] recommends using Built-In-Self-Test to tolerate defects. During the mapping process, nano blocks in a system can be searched whether any of them can be used for mapping logic function. Therefore, the per-chip placement and routing would not be needed anymore. The biggest advantage of such an idea is that while the defect rate is extremely high for emerging nano technologies, using a defect map will require high complexity. With the removal of the need for per-chip placement and routing, the complexity behind the defect map will be eliminated. In addition to the complexity, the storage is another problem. Since the future systems will take advantage of smaller device sizes and more dense logic functions compared to current technology, storing defect map for highly defective devices will require extreme space.

In addition to the mapping algorithms to tolerate defects, architectural techniques have also been presented. For example, while CMOS technology presents a more robust plane, CMOL takes the advantage of using nano crossbars on CMOS creating a hybrid architecture [3].

Defect tolerance for emerging nano technologies is a major issue, but not only the one. As mentioned before in Section 2.2, the expected variation for emerging technologies is extremely high. For the variation tolerance, researchers have presented some methods to minimize the effect of variations on single nanowires such as adding buffers, changing wire size and width as well as wire length [32].

In [52], manufacturing techniques in order to increase the control over variation are mentioned. In their work, they focused on nanowire based decoders. They connect

the variation as threshold voltage deviation. Using Grey code, the effect of variation due to manufacturing can be reduced. Additionally, they recommend using longer codes for decoders where the effect of the variation seems decreasing with the length of the codes because longer codes will require less transitions.

Moreover, another study focuses on the modeling [53]. Since a more realistic model should be used when the variation is extremely high, the researchers work on Carbon nanotube based FETs. Using models those represent the real devices more accurately, it is possible to know the effect of variation on circuits where worst case scenarios will be more realistic.

Additionally, for the architectural point of view, it has been pointed out that the re-configurability feature of nano crossbar arrays can be used to optimize variations [27]. While variations can be modeled as delay differences, the circuits will show different performance. In this manner, the devices with low performance can be bypassed and more suitable devices for that operation can be used.

For variation tolerance during the design phase, NASIC architecture is investigated [54]. Since NASIC uses nanowire based FETs for logic implementations, variations caused by the doping issues, channel length, etc. have significant effects and result in timing mismatches as well as exact critical path may not be detected during the design phase. Therefore, in this study, it is assumed that sources of error include permanent defects, process and environmental variation related errors, transient errors, as well as internal and external noise related errors for NASIC architecture. Then, they try to built defect tolerance for NASIC architecture. In the scope of defect tolerance, they add redundancy and interleaving (when the redundancy may not be enough). They further consider using Hamming Codes for error correction.

The approach in [55] uses a common CMOS based FPGA architecture, enhances it using CNFET and proposes Field Programmable Carbon Nanotube Array (FPGCNA). They first characterize the components considering variations by modifying well-known FPGA tools. Then they use statistical timing analysis and apply local/global routing considering variation effects.

A study in [34] presents modeling and a mapping algorithm to tolerate variations for NanoPLAs. In this study, they focus on mapping the logic functions for crossbars by implementing the slowest logical NAND-term to the fastest physical NAND-term. They explain timing for a NanoPLA as the switching time for the slowest NAND-term to switch. If the timing of the circuit goes beyond the predetermined

margins, the NanoPLA will not meet the constraints and considered as defective. To be able to model delay of a NAND term, they apply Elmore Delay models and variation is modeled as Gaussian distribution. They propose different methods for variation tolerance. First, the NAND terms not meeting constraints are considered as defective and mapping is applied by bypassing these resources. Therefore the constraints would be met with a successful mapping. They additionally propose another method that tries to reduce the delay by mapping the slowest logical NAND term to the fastest physical NAND term.

# Chapter 3

## Delay Modeling and Characterization Testing of Nano Crossbars

In this chapter, first, we give definitions for variation and defect tolerance techniques. Next, we explain the proposed delay models and their calculation for both crossbar structures (i.e. diode based and FET based nano crossbars). We further explain characterization testing methods for obtaining variation and delay values of each crosspoints (based on lumped delay modeling). Since defect tolerance is another major issue with variation tolerance, we extend our method for defect tolerance.

Crossbars are considered as the main building blocks for emerging nano technologies. However, for logic functions, instead of using one large crossbar, logics are divided for multiple crossbars where the crossbars are cascaded to each other composing crossbar arrays. Hence, we also extend our work for crossbar arrays.

### 3.1 Definitions

For the variation and defect tolerant logic mapping based on the lumped variation model, we define various matrices, as follows.

The binary *Function Matrix* (FM) of size  $n \times m$  indicates the logic function to be mapped into a crossbar with a size of  $n \times m$ . For both crossbar structures, each

row of FM corresponds to an input and each column corresponds to an output. The entries of FM are defined as below:

$$FM_{i,j} = \begin{cases} 1, & \text{if output } j \text{ depends on input } i \\ 0, & \text{otherwise} \end{cases}$$

Figure 3.1 shows a multi-output logic function “ $O_1 = I_1, O_2 = I_1.I_3, O_3 = I_1.I_3.I_4, O_4 = 0$ ” mapped into a diode based crossbar as well as a similar multi-output logic function “ $O_1 = \bar{I}_1, O_2 = \bar{I}_1.I_3, O_3 = \bar{I}_1.I_3.I_4, O_4 = 0$ ” into a FET based crossbar with a size of  $4 \times 4$ .

For the diode based crossbar, the diodes demonstrate the activated junctions that show diode behavior and where there is no diode at the crosspoint, the junction is deactivated (will not be effective for logic function). In addition, for the FET based crossbars, the FET crosspoints show that they are activated to show FET behavior. The corresponding FM for both crossbars is demonstrated in Table 3.1.

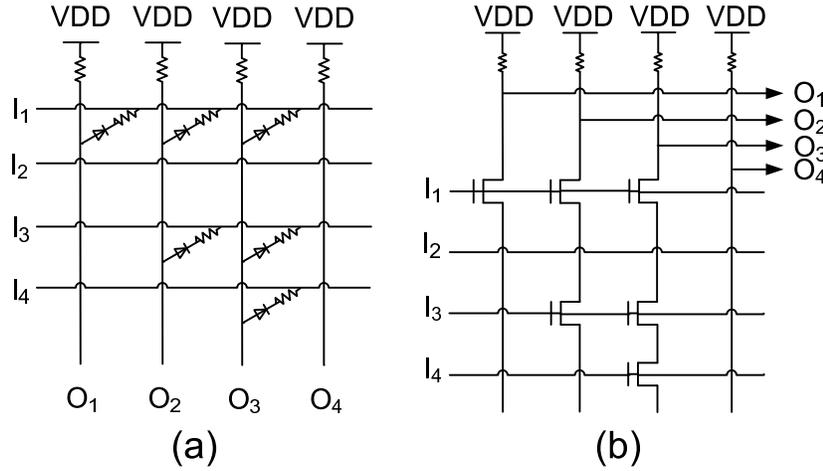


FIGURE 3.1: The implementations of two different multi output logic functions, which are based on the same FM, for diode based crossbar (A) and FET based crossbar (B).

TABLE 3.1: Function Matrix corresponding to the functions in Figure 3.1.

$I_1$	1	1	1	0
$I_2$	0	0	0	0
$I_3$	0	1	1	0
$I_4$	0	0	1	0
In/Out	$O_1$	$O_2$	$O_3$	$O_4$

It should be noted that the Function Matrix provided in Table 3.1 is just for pre-configuration where the mapping of the horizontal (vertical) nanowires of the crossbar to the inputs (outputs) of the function (as expressed in FM) has not been fixed yet (i.e. can be changed using the reconfigurability feature of crossbars).

The lumped variation model in a crossbar is represented by a real  $n \times m$  matrix called *Variation Matrix* (VM). VM has elements of real numbers where each entry indicates delay of individual crosspoints (either diode based or FET based) in the crossbar. As an example, a VM of a nano crossbar is provided in Table 3.2 in which numbers between 0 and 100 represent the normalized variation of each crosspoint.

While FM is the property of a logic function and it is fixed for different copies of crossbars (or nano chips) implementing that function, VM is the property of the individual crossbars and varies from one crossbar to another crossbar.

TABLE 3.2: An example of a VM for a  $4 \times 4$  crossbar.

90	50	10	95
80	25	35	55
40	45	75	10
10	85	20	35

It should be noted that VM depends on the electrical characteristics of a crossbar where each entry represents a crosspoint. Therefore, using VM, it is possible to represent any crossbar circuit (either diode based or FET based).

The important features of nano crossbars are reconfigurability, interchangeability, and abundance of programmable resources. For a given function (specified in an FM), there exist several different mappings of that particular function to the crossbar, i.e. how the inputs and outputs of the FM are assigned to the horizontal and vertical nanowires of the crossbar (and as a result, using different crosspoints in the mapping of the function). Therefore, we express the mapping of the function to a crossbar of size  $n \times m$  with two vectors:

- $n \times 1$  *Input Mapping Vector* (IMV):  $IMV[i] = j$  if input  $x_i$  is assigned to horizontal nanowire  $j$ .
- $1 \times m$  *Output Mapping Vector* (OMV):  $OMV[i] = j$  if output  $f_i$  is mapped to vertical nanowire  $j$ .

In other words,  $IMV$  ( $OMV$ ) represents a permutation of numbers from 1 to  $n$  ( $m$ ). As an example, the actual mapping (implementation) of the same logic functions (Table 3.1) using different crosspoints is shown in Table 3.3 with the implementation of the crossbar circuits in Figure 3.2. For this configuration, for both crossbar circuits,  $IMV = \{4, 3, 1, 2\}$  and  $OMV = \{1, 4, 3, 2\}$ .

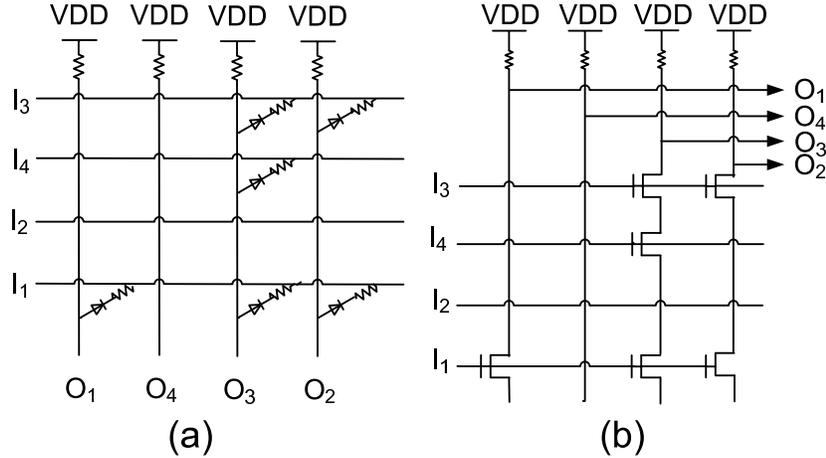


FIGURE 3.2: The implementation of the same logic functions using a different configuration.

TABLE 3.3: Different mapping the same multi-output logic function

$I_3$	0	0	1	1
$I_4$	0	0	1	0
$I_2$	0	0	0	0
$I_1$	1	0	1	1
In/Out	$O_1$	$O_4$	$O_3$	$O_2$

## 3.2 Delay Models for Nano Crossbars

Nano crossbars can be built either diode based or FET based. Therefore, delay models for each crossbar will be different than the other. Hence, in this section, we explain the delay models for both diode and FET based crossbars.

### 3.2.1 Delay Models for Diode Based Crossbars

In diode-based crossbars, the logic is implemented as Resistor Diode Logic (RDL). In RDL, programmable diodes are connected between outputs and inputs in a parallel

way, i.e. they are not cascaded. For both AND and OR logic implementations, all inputs are connected to corresponding crosspoint diodes which are all in parallel connected to the output nanowire as shown in Figure 3.1 and Figure 3.2.

Since all input connections are in parallel, for a non-controlling transition (e.g.  $0 \rightarrow 1$  for AND), the output will make a transition with the slowest input path makes the transition. Therefore, the maximum delay of an output is proportional to the maximum delay of the crosspoints connected to the inputs. This means that the delay of an output can be estimated by the maximum ‘used’ (activated) entry of VM for the corresponding column of FM.

For a particular mapping (specified with IMV and OMV), the *cost* (maximum delay) of output  $f_i$  denoted by  $C(f_i)$ , is calculated as:

$$C(f_i) = \max_{k=1}^n (FM[k][i] \times VM[IMV[k]][OMV[i]]) \quad (3.1)$$

For example, considering the FM (Table 3.1) and corresponding VM (Table 3.2), with identity IMV and OMV ( $IMV[i] = i$ ,  $OMV[j] = j$ ) the costs of each output can be calculated as follows.  $Costs = \{\max(90), \max(50, 45), \max(10, 75, 20), 0\} = \{90, 50, 75, 0\}$ .

### 3.2.2 Delay Models for FET Based Crossbars

While for diode based crossbars, the diodes are connected in parallel and the output will make a transition with the slowest crosspoints, in FET based crossbars, all the FETs are cascaded. Therefore, instead of making a transition with the slowest crosspoint, the output will be dependent on the all FETs connected for this output. Hence, for a particular mapping of a function to a crossbar of size  $n \times m$  (specified with IMV and OMV), the cost of output  $f_i$  (in terms of delay or variance, depending on the interpretation of VM), denoted by  $C(f_i)$ , is calculated as:

$$C(f_i) = \sum_{k=1}^n FM[k][i] \times VM [IMV [k]] [OMV [i]] \quad (3.2)$$

For example, considering the FM (Table 3.1) and corresponding VM (Table 3.2) are demonstrating a FET based crossbar, the costs of each output should be calculated

based on the identity IMV and OMV ( $IMV[i] = i$ ,  $OMV[j] = j$ ) as follows.  $Costs = \{90, 50 + 45, 10 + 75 + 20, 0\} = \{90, 95, 105, 0\}$ .

### 3.2.3 Optimization Goals

During the optimization of the cost of a crossbar (either diode or FET based), the main criteria should be to minimize the critical path, i.e. minimizing the maximum cost over all outputs of the crossbar. This means that  $D_{max} = \max_i C(f_i)$  should be minimized.

$$Objective\ 1 = D_{max} = \max_i C(f_i) \quad (3.3)$$

Please note that while for diode based crossbars, the cost is the delay of the slowest ‘used’ crosspoint for an output, for the FET based crossbars the cost is the sum of the delay of all ‘used’ crosspoints for that output.

In addition to minimizing the maximum cost, other different optimization objectives can be considered. Another optimization goal could be balancing all the path delays. This corresponds to minimizing the maximum and the minimum cost differences of a crossbar as shown below.

$$Objective\ 2 = D_{max} - D_{min} = \max_i C(f_i) - \min_j C(f_j) \quad (3.4)$$

It should be noted that during the optimization of Objective 2, zero costs at the outputs should be ignored since zero costs correspond to unused outputs (all-zero column for an output in FM).

To give an example for a diode based crossbar, considering the FM (Table 3.1) and corresponding VM (Table 3.2), with identity IMV and OMV ( $IMV[i]=i, OMV[j]=j$ ) the objectives can be calculated as follows.

$$Costs = \{90, 50, 75, 0\},$$

$$Objective\ 1 = D_{max} = 90, \text{ and}$$

$$Objective\ 2 = D_{max} - D_{min} = 90 - 50 = 40.$$

However, for a FET based crossbar, since the cost for each output is the summation of each used VM entry in the same row,

*Objective 1* =  $D_{max} = 105$ , and

*Objective 2* =  $D_{max} - D_{min} = 105 - 90 = 15$ .

### 3.3 Characterization Testing for Crossbars

The presented mapping technique requires the variation (delay) of individual crosspoints (based on the crossbar structure) to be extracted prior to the mapping process in order to populate the Variation Matrix. This can be done in the characterization testing step, which in turn can be implemented as an extension of a delay testing procedure. Since both diode and FET based crossbars have a regular structure and they are also reprogrammable, an efficient delay testing technique can be devised for variation/delay characterization. We exploit these features for a fast and efficient delay characterization, as described next.

Our proposed delay testing (characterization) method is as follows. If the inputs are row-wise and the outputs are column-wise, all the outputs act as primary outputs and can be read simultaneously. We apply both falling and rising transitions through each crosspoint. Using path delay testing measurements, we infer the delay/variation values of individual FETs and create VM. The test process consist of one *test configuration* and a series of *test vectors*.

In the configuration of the crossbar, all the crosspoints are activated (all elements of FM are 1). In other words, each output line implements a function of all input lines (the implemented function depends on the structure of the crossbar, NAND or NOR logic for FET based crossbars and AND or OR logic for diode based crossbars). For an  $n \times m$  crossbar,  $n$  test vector triples are applied. For each input, two *controlling* transitions (falling and rising) are applied while all the other inputs are stable at the *non-controlling values*, i.e. only one input is making a transition at a time. For example, the non-controlling value for logic AND/NAND is 1 and for logic OR/NOR is 0. Since only one input ( $i$ ) is making a transition and all other inputs are stable at the non-controlling value, the transition delay observed at output  $j$  is due to the delay of only one crosspoint located at  $(i, j)$ . In other words, there is only one switch

on the transition path from input to each output. The entries of VM are populated as an average of rising and falling transition delays/variations.

Specifically, for a crossbar acting as an OR/NOR-plane, all the inputs are 0. Then, the last input make a transition to 1. By measuring the falling transition time at all  $m$  outputs, the rising delay of the last row of the crosspoints are obtained. If the transition does not arrive or arrives too late on a particular output, the corresponding crosspoint on that column is considered as defective. After all values are stabled, then that input transitions back to 0 and the rising transition delay on all outputs are sensed (a crosspoint is defective if it fails either of rising or falling transition tests). This process is repeated for every other input such that in  $n$  steps the delay of all crosspoints can be characterized. The process for a crossbar acting as a AND/NAND-plane is the dual of the above procedure.

This delay testing (characterization) procedure requires only *one* test configuration and  $O(n)$  test vectors for an  $n \times m$  crossbar. Therefore, even though we need to characterize the entire crossbar, we take the advantage of reprogrammability for the fastest possible testing procedure by using the minimum number of configurations and test vectors. This delay testing process can be extended to crossbar arrays in a straightforward way.

To consider the effect of crosstalk (which can occur during normal operation) in this characterization test, one extra test configuration and one input pair can be added. In the new test configuration, all crosspoints, except those identified as defective by the above procedure, are activated. In the test pattern, all inputs are making a transition from controlling to non-controlling values (e.g. in an AND-plane, all-0 vector followed by all-1). Since robust all-input path delay testing patterns are applied for all outputs simultaneously, the effect of crosstalk (worst case scenario) is sensitized. The difference of the delay observed at each output  $j$  (corresponding to the delay of slowest crosspoint in column  $j$  with the presence of crosstalk) and the maximum delay of column  $j$  of VM, characterized by the above procedure using one-input-transition test (no crosstalk), shows the contribution of crosstalk to the actual delay. All elements of column  $j$  of VM can be adjusted by this delay difference accordingly.

Please note that for changes in VM over runtime that result in timing failures, we can perform periodic characterization and apply re-mapping to avoid failures.

### 3.4 Modeling Defects in Crossbars

The number of defects in nanoscale devices fabricated using bottom-up manufacturing process is expected to be significantly higher than that for CMOS devices fabricated by conventional top-down lithography patterning. This is mainly because of inherent lack of control in self-assembly fabrication as well as atomic scale of nanodevices. Therefore, dealing with defects is a major aspect of design flow in crossbar nano architectures.

The defects in a crossbar can be generalized into two types for crosspoints: *stuck-open*, where the crosspoint cannot be activated, and *stuck-closed*, where the crosspoint cannot be deactivated. Defects in nanowires, such as opens and shorts, can be modeled by crosspoint faults, as described in [29]. After defective crosspoints are modeled with the infinite delay at the VM ( $V_\infty$ ), mapping algorithms can be modified to minimize variations by rejecting the infinite costs (i.e. configurations with defects).

An example of the representation of defects in VM for a  $4 \times 6$  crossbar is shown in Table 3.4. In this example, (1, 5) and (4, 1) contain stuck-open defects whereas (3, 3) contains a stuck-closed defect where all crosspoints of the wires that intersect at (3, 3) are unusable (all switches at row 3 and column 3 become unusable).

TABLE 3.4: Representing defects in VM

95	75	$\infty$	55	$\infty$	85
70	90	$\infty$	70	25	25
$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
$\infty$	55	$\infty$	80	60	85

Please note that the configurations that uses defective crosspoints will result in infinite cost. For diode based crossbars, since the cost is the maximum used crosspoint, the cost will be infinite and for FET based crossbars, since the output cost is the sum of the used crosspoints, the output cost will become infinite with the used defective crosspoint.

### 3.5 Modeling Crossbar Arrays

During the implementation of logic functions, instead of using a large crossbar (e.g. with a size of  $1000 \times 1000$ ), one should use smaller crossbars and cascade them considering the reliability issues, modularity, etc. Hence, usage of crossbar arrays (i.e. cascaded crossbars) are also an important problem. Here we explain the basic idea for the crossbar arrays and their optimization. One example of crossbar arrays is shown in the Figure 3.3.

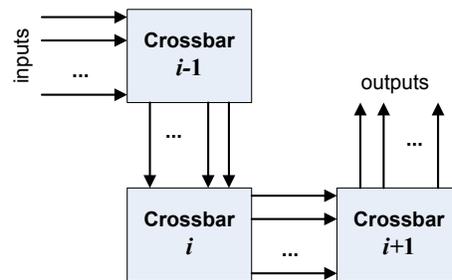


FIGURE 3.3: The basic idea of implementing a crossbar array using crossbars.

As shown in the Figure 3.3, a crossbar array is a set of cascaded crossbars. For this structure, the outputs of the previous crossbar (crossbar  $i - 1$ ) are set as the inputs of the next crossbar (crossbar  $i$ ). Briefly, one big logic function is divided into smaller logic function.

It should be noted that for the best efficiency in terms of logic mapping/routing for crossbar arrays, it is more efficient to have a large array of (relatively) small crossbars [2]. Additionally, the size of the crossbar array for delay optimization represents the combinational logic between two bistable stages (e.g. flip flops) and efficient combinational logic depth is not too large either [2].

For the extension of the delay calculation for *crossbar arrays* (in which the outputs of the previous crossbar are connected to the inputs of the next crossbar), the delay of the previous crossbar output (column)  $i$  will be added to the delay of input (row)  $i$  of the next crossbar. This is because the crossbars are cascaded (in series connection) and the delay from the primary input to the primary output is sum of the delays of the crosspoints in the path.

### 3.5.1 Diode Based Crossbar Arrays

Consider two diode based  $3 \times 3$  and  $3 \times 2$  crossbars are cascaded. Their corresponding VMs and mappings (the configuration mapped into the crossbar based on FM and IMV/OMV) are shown in Table 3.5, respectively. If the crossbars are considered independently, the delays for the outputs of crossbars  $i$  and  $i + 1$  are  $\{90, 55, 60\}$  and  $\{75, 45\}$ , respectively. Since the input  $j$  of the second crossbar is connected to output  $j$  of the first crossbar, the delay of first stage is added to the delay of second stage. The entire path delay can be represented by adjusted VM values of the second crossbar as shown in Table 3.6.

TABLE 3.5: VMs and configurations for the cascaded crossbars.

crossbar (i)						crossbar (i+1)				
VM			Conf.			VM			Conf.	
90	75	60	1	0	1	40	15	1	0	
70	95	20	1	0	1	35	45	0	1	
45	55	55	0	1	0	75	30	1	0	

TABLE 3.6: Adjusted VM for the diode based crossbar  $i + 1$  ( $3 \times 2$ ).

90+40	90+15
55+35	55+45
60+75	60+30

### 3.5.2 FET Based Crossbar Arrays

While for diode based crossbars arrays, the cost of each crossbar is calculated with the maximum used crosspoint, the cost of FET based crossbar is the sum of the used crosspoint VM entries. Therefore, the delay extension for FET based crossbar is different than the diode based. Based on the same example with the diode based crossbar arrays, assume that two FET based  $3 \times 3$  and  $3 \times 2$  crossbars are cascaded. Their corresponding VMs and mappings (the configuration mapped into the crossbar based on FM and IMV/OMV) are shown in Table 3.5, respectively.

When the crossbars are considered independently, the delays for each output will be as  $\{90 + 70, 55, 60 + 20\} = \{160, 55, 80\}$  for crossbar  $i$  and  $\{40 + 75, 45\} = \{115, 45\}$  for crossbar  $i + 1$ . However, when they are cascaded,  $i + 1$  crossbar VM should be recalculated since the output costs of the previous crossbar will be added to the

delay of second one. Therefore, the adjusted VM for the crossbar  $i + 1$  would be as in Table 3.7. And, the cost of the crossbar array (the delay from the first inputs to the last outputs) will be  $\{160 + 40 + 80 + 75, 55 + 45\} = \{355, 100\}$ . As can be seen the delay calculation for FET based crossbar arrays is different than the diode based crossbar and require appropriate adaptation of the optimization algorithms.

TABLE 3.7: Adjusted VM for the FET based crossbar  $i + 1$  ( $3 \times 2$ ).

160+40	160+15
55+35	55+45
80+75	80+30

### 3.5.3 2D Crossbar Arrays

While the previous structure, i.e. crossbar arrays, are extending the logical function implementation from one large crossbar to relatively smaller crossbars, the implementation is only one dimensional (1D) –  $i - 1^{th}$  crossbar output is directly connected to  $i^{th}$  crossbar. However, this approach can further be developed by using two dimensional (2D) crossbar arrays.

For a 2D crossbar array (Figure 3.4), the crossbars are not only connected to the previous ( $i - 1$ ) and the next crossbar ( $i + 1$ ); crossbar  $i$  is also connected to the crossbars at other rows. More specifically, using appropriate modeling, crossbar  $(i, j)$  in a 2D crossbar array is connected to crossbar  $(i, j - 1)$ ,  $(i, j + 1)$ ,  $(i - 1, j)$ , and  $(i + 1, j)$ . So, a crossbar will have connections with 4 crossbars and among these 4 crossbars, 2 of them will be supplying inputs and the other 2 of them are using the outputs.

2D crossbar arrays can be further developed to make it resemble FPGAs like architectures in order to have a better routing features. In this case, instead of using every single crossbar for logic operations, some of them can be used for connection (Figure 3.5). Hence, we categorize the crossbars as Logic Block (LB) and Connection Block (CB). It should be noted that in the structure of crossbars, there will not be any differences whereas the only difference is what they are used for. For example, a CB with an FM of identity function will be used to transfer the outputs of a crossbar to another crossbar as inputs.

It should be noted that while for 1D crossbar array, the VM is affected by only the previous crossbar output, for 2D crossbar array, the VM will not be only affected by

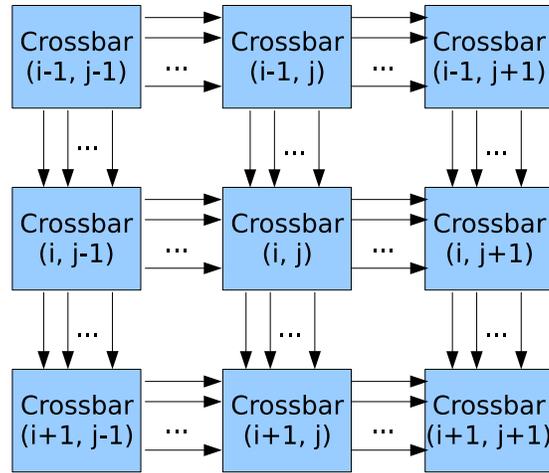


FIGURE 3.4: 2D crossbar arrays

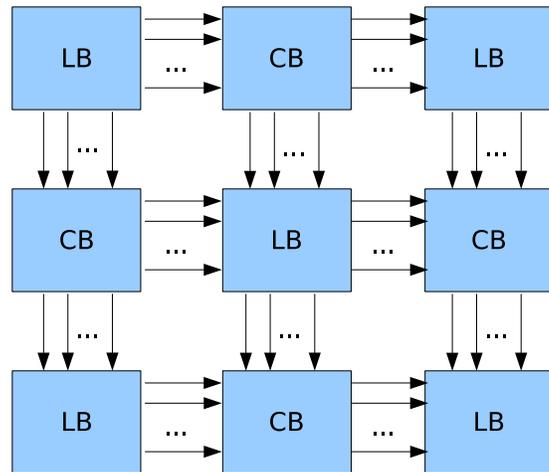


FIGURE 3.5: 2D crossbar arrays with logic blocks and connection blocks

the previous crossbar since a crossbar has multiple inputs. In this case, a crossbar  $(i, j)$  VM will be affected by crossbar  $(i - 1, j)$  and crossbar  $(i, j - 1)$  outputs. we can model the new VM for crossbar  $(i, j)$  with the following equation.

$$crossbar(i, j).VM[x, y] += crossbar(i - 1, j).cost(x) + crossbar(i, j - 1).cost(y) \tag{3.5}$$

# Chapter 4

## Algorithms

For nano crossbars, the important features are reconfigurability, interchangeability, and abundance of programmable resources. Using these features, for a crossbar, a given function can be implemented using several different mappings with different IMVs and OMVs (i.e. assigning different inputs and different outputs for the same FM) and using different crosspoints (diode or FET). Therefore, as shown by the Equations 3.1 and 3.2, different mappings (based on FM with IMV and OMV) will result in different costs. As a result, one can optimize the crossbars using interchangeability and reconfigurability feature for different optimization criteria.

In this chapter, we will explain the algorithms developed for variation and defect tolerance.

### 4.1 Exhaustive Search

While different IMV and OMV sets can be used for the search of an optimal crossbar mapping, one can try exhaustive search. For the exhaustive search, all permutations of IMV and OMV are tried and for each permutations of a mapping vector, the cost is calculated. If the new permutation results in less cost than the previous found cost, meaning that this configuration optimizes circuit more than the previous configuration, we keep the mapping vectors for further comparison. At the end of the search, only the best permutation(s) resulting in the least cost will be found.

For this method, we apply all permutations of IMV and OMV. For a crossbar with a size of  $n \times m$ , there are  $n!$  permutations for IMV and  $m!$  permutations for OMV.

```

begin
   $min\_cost = \infty$ 
   $best\_IMV = \emptyset$ 
   $best\_OMV = \emptyset$ 
  foreach permutation of IMV do
    foreach permutation of OMV do
       $cost = calculate\_cost(FM, IMV\_permutation, OMV\_permutation)$ 
      if ( $cost < min\_cost$ ) then
         $best\_IMV = IMV\_permutation$ 
         $best\_OMV = OMV\_permutation$ 
         $min\_cost = cost$ 
    end
  end

```

**Algorithm 1:** Exhaustive Search algorithm for nano crossbars

Therefore, for an exhaustive search of an  $n \times m$  crossbar, we need to try  $n! \times m!$  different mappings.

Please note that the presented exhaustive algorithm shows the overall idea and for FET based crossbars or diode based crossbars, the cost calculation should be implemented based on the Equation 3.1 and Equation 3.2.

Even though exhaustive search can be used to find out the best optimized mapping, the search space is exponential to the crossbar size. Therefore, it would be intractable to apply exhaustive algorithms for big crossbars. Hence, we need heuristic algorithms that will be used to find an acceptable sub-optimal mapping in reasonable time.

## 4.2 Simulated Annealing

### 4.2.1 Concept

During the optimization of the crossbars based on different optimization goals, we cannot apply exhaustive search for large crossbars. Therefore, we require combinatorial optimization methods for this optimization problem. In this section, we will explain the adaptation of Simulated Annealing algorithm for the optimization of crossbar and crossbar arrays for variation and defect tolerance. Simulated Annealing has been chosen for this optimization due to the fact that it has been widely used in VLSI design automation, especially for the placement problem [56].

Simulated Annealing is based on the minimization of energy probability distribution of metals by first heating them and then cooling gradually [56]. During the execution of the algorithm, a new perturbation of the solution is randomly generated (*Move*) and the cost of the new solution is compared with the old solution (*Cost Function*). Using this analogy with the physical model, Simulated Annealing accepts new solution when  $\Delta cost < 0$ . Also it accepts moves resulting in more cost (“bad moves”) with a probabilistic acceptant of  $random(0, 1) < e^{(-\Delta cost/temperature)}$ , in order to avoid local optimums. In high temperatures (early iterations), bad moves are accepted with higher probabilities; while toward the cool down of temperature (final iterations), the cost converges to the final values. A detailed algorithm for the Simulated Annealing can be seen in the Algorithm 2.

```

begin
  temperature  $\leftarrow$  high_temperature
  while (temperature  $\geq$  cold_temperature) do
    for ( $i = 1..N$ ) do
      [new_IMV, new_OMV]  $\leftarrow$  Move(IMV, OMV)
       $\Delta Cost \leftarrow$  calculate_delta_cost(new_configuration, old_configuration)
      if ( $\Delta Cost < 0$  or  $random(0, 1) < e^{-\Delta Cost/temperature}$ ) then
        IMV  $\leftarrow$  new_IMV
        OMV  $\leftarrow$  new_OMV
      temperature  $\leftarrow$  temperature *  $\alpha$   $\setminus\setminus$  ( $0 < \alpha < 1$ )
    end
  end

```

**Algorithm 2:** Simulated Annealing algorithm

It should be noted that the Simulated Annealing algorithm starts with a high temperature and then gradually cools down. So, at the high temperatures, even the bad moves will be accepted to avoid falling into local optimums at the very early stages. Furthermore, the temperature cools down at every stage by a constant  $\alpha$ . Even though there are other methods to cool down the temperature at every step, in this study we chose a cool down process of using  $\alpha$ . Additionally, in the inner loop (i.e.  $for(i = 1..N)$ ), we apply the perturbations (i.e. moves) and delta cost calculations without changing the temperature. Last but not least, for different crossbar structures (i.e. diode or FET based), initial temperature, terminating temperature, and the  $N$  values are obtained empirically.

#### 4.2.1.1 Simulated Annealing for Diode Based Crossbars

While Simulated Annealing is a general optimization algorithm, some modifications should be considered for diode based crossbars. Since the acceptance of a new configuration is based on the criteria ( $\Delta Cost < 0$  or  $random(0, 1) < e^{-\Delta Cost/temperature}$ ), temperature should be well defined for initial and the final values whereas the  $\Delta Cost$  is the result of every single moves in IMV and/or OMV and cannot be predefined.

During the fine tuning of Simulated Annealing for diode based crossbars, one should pay attention to the initial and the final temperature values as follows. Since the cost of an output is the maximum ‘used’ crosspoint delay, a change in the used crosspoints will give a  $\Delta Cost$  of the difference of the maximum used crosspoint of the previous configuration and the maximum used crosspoint of the next configuration. Therefore, the algorithm will converge soon (i.e. the swaps other than maximum used crosspoint cost, the cost for diode based crossbars will not change). Considering that the random acceptance rate increases when  $-\Delta Cost/temperature$  is high, the final temperature should not be high. Furthermore, the  $\alpha$  value should not be too low because with the low  $\alpha$  value, the algorithm will converge very quickly and most probably will result in a local minimum value with high cost.

#### 4.2.1.2 Simulated Annealing for FET Based Crossbars

For FET based crossbars, the output cost is the sum of the VM values of the used crosspoints for an output whereas for diode based it is the maximum used crosspoint. Therefore, unlike diode based crosspoints,  $\Delta Cost$  might be higher even by the end of optimization. than the maximum VM entry with a new configuration. Hence, the final temperature for FET based crossbars does not have to be as small as the final temperature for diode based crossbars. While the maximum crosspoint is used, the swaps with the crosspoints with less cost will still affect the output cost. Moreover,  $\alpha$  value still holds the same feature for FET based crossbars as well.

For both diode and FET based crossbars, the  $N$  value for the inner loop is determined as a function of the  $n$  and  $m$  (the size of the optimized crossbar) to have more iterations for larger crossbars so that the crossbar might be well optimized. For the  $N$  function, we determined it as  $N = 2 \times n \times m$ .

## 4.2.2 Moves

We consider different moves for the implementation of the Simulated Annealing mapping, namely:

1. Changing input assignment (permutation in IMV, OMV is fixed)
2. Changing output assignment (permutation in OMV, IMV is fixed), and
3. Changing both input and output assignments (permutations in both IMV and OMV)

The first two moves represent restricted mapping which will be important in the extension of this technique for the crossbar array (multi-stage crossbars), as will be explained in Sec. 4.3. To implement the random move function, two elements of the corresponding vector are randomly selected and their values are swapped. Based on Equation 3.1, 3.2, this swap will affect the cost function. When unrestricted move (Option 3) is implemented, first, one of the vectors is randomly selected and then the random swap within the selected vector (IMV or OMV) is performed.

For example, when we change only the inputs (IMV) of the initial mapping (Figure 3.1), we get a new IMV of  $IMV = \{1, 2, 4, 3\}$  as shown in Table 4.1. Then, if we change the output order, the new OMV becomes  $OMV = \{3, 2, 1, 4\}$  (Table 4.2).

TABLE 4.1: Perturbation only in input mapping vector (IMV)

$I_1$	1	1	1	0
$I_2$	0	0	0	0
$I_4$	0	0	1	0
$I_3$	0	1	1	0
In/Out	$O_1$	$O_2$	$O_3$	$O_4$

TABLE 4.2: Perturbation only in output mapping vector (OMV)

$I_1$	1	1	1	0
$I_2$	0	0	0	0
$I_4$	1	0	0	0
$I_3$	1	1	0	0
In/Out	$O_3$	$O_2$	$O_1$	$O_4$

After every move, the cost function for the new crossbar mapping based on *Objective 1* or *Objective 2* using Equation 3.1 or Equation 3.2 is recalculated and the delta cost as the difference of the new cost and the old cost is computed.

### 4.2.3 Efficient Delta Cost Calculation

After any moves, the new cost should be calculated to find out the  $\Delta cost = cost_{new} - cost_{old}$ . This can be done by calculating all the output costs. However, this process has a time complexity of  $O(n \times m)$ . Since this recomputation is done in the inner loop of Simulated Annealing, improvement in delta cost calculation can greatly affect the overall run time of the algorithm. In our implementation, instead of calculating all output costs at the end of every move, only the cost difference due to swapping of two inputs or two outputs, is recalculated which improves the overall performance.

In the followings, the methods for improving the efficiency in delta cost calculation will be explained for both crossbar structures.

#### 4.2.3.1 Efficient Delta Cost Calculation for Diode Based Crossbars

When two items  $i$  and  $j$  in IMV are swapped with a move during the optimization of the crossbar using Simulated Annealing, all output costs will be affected. Calculation of delta cost, in this case, needs the calculation of the whole crossbar to determine the maximum used crosspoint. Therefore, for delta cost calculation of diode based crossbars, there is no efficient method.

On the other hand, when the change is in the OMV ( $k$  and  $l$  are swapped), instead of recalculating every output costs, one could only look for the costs of the changed OMVs as follows.

$$\begin{aligned} \forall i, \Delta C(f_k) = & \max(FM[i][k] \times VM[IMV[i]][OMV[k]]) \\ & - \max(FM[i][k] \times VM[IMV[j]][OMV[l]]) \end{aligned} \quad (4.1)$$

#### 4.2.3.2 Efficient Delta Cost Calculation for FET Based Crossbars

With the similar approach of diode based crossbars, it is also possible to have an efficient delta cost calculation for FET based crossbars. Since for FET based crossbars, instead of using the maximum used VM entry, the sum of the used VM entries is used for cost calculation, the following methods can be used for reducing the complexity of delta cost calculation.

For a move in IMV (swapping  $i$  and  $j$ ), the following equation may be used.

$$\begin{aligned} \forall k, \Delta C(f_k) = & FM[i][k] \times VM[IMV[i]][OMV[k]] \\ & - FM[j][k] \times VM[IMV[j]][OMV[k]] \end{aligned} \quad (4.2)$$

For the move in the OMV ( $k$  and  $l$  are swapped), the efficient delta cost calculation is as follows.

$$\begin{aligned} \forall k, \Delta C(f_k) = & FM[i][k] \times VM[IMV[i]][OMV[k]] \\ & - FM[i][k] \times VM[IMV[j]][OMV[l]] \end{aligned} \quad (4.3)$$

#### 4.2.3.3 The Complexity of the Efficient Delta Cost Calculation

Without any efficient methods, the calculation would take  $O(m \times n)$  with calculating every output cost and then finding the difference. However, with the proposed method, the delta cost calculation is as follows for FET based crossbars. The time complexity of finding delta cost is now reduced to  $O(m)$ , when the move is in IMV since only the changed IMVs are considered in the calculation. When two items  $i$  and  $j$  in OMV are swapped, only the output costs of  $f_i$  and  $f_j$  will be affected and need to be recomputed. The time complexity of this operation is  $O(n)$ . Also, for diode based crossbars, when the move in OMV is considered, the time complexity will be  $O(n)$ . It should be noted that when the large crossbars are considered and when the iteration in Simulated Annealing is extremely high, the effect of the efficiency will be more important.

#### 4.2.4 Defect Tolerance

While defects in bottom-up self-assembled nanoscale devices is expected to be higher than CMOS top-down lithography based devices, methods dealing with defects is a major aspect of design flow.

In Section 3.4 an overview for the reasons of the defects, their representation are provided. To summarize, defects in a crossbar can be generalized into two types for

crosspoints: *stuck-open*, where the crosspoint cannot be activated, and *stuck-closed*, where the crosspoint cannot be deactivated. In addition, defects in nanowires can be modeled by crosspoint faults [29] and defective crosspoints can be modeled with the infinite delay at the VM ( $V_\infty$ ).

During the implementation of crossbars, infinity can be represented by a large enough number. Specifically, if the maximum cost for a defect free crossbar is represented by a number  $V_{max}$ , any VM entry greater than  $V_{max}$  can be used to represent infinite VM entry ( $V_\infty$ ). Therefore, any output cost greater than  $V_{max}$  can be treated as infinite (defective). Specifically, if the maximum non-defective entry in an  $n \times m$  VM is  $V_{max}$ , infinity,  $V_\infty$  can be represented by any number greater than  $n \times V_{max}$ . Which means that any cost function calculated as  $V_\infty$  or greater can be treated as infinity (defective).

While the configurations that use defective crosspoints will have infinite cost, algorithms can be modified to tolerate defects.

The modification of Simulated Annealing can be succeeded as follows. For the modification for defect tolerance, if the new perturbation (move) uses a defective crosspoint, its cost become infinite (as well as the delta cost) for both crossbar structures and it will automatically be rejected by the Simulated Annealing where  $\Delta Cost = \infty$  and  $e^{-\infty/Temp} = 0$ . In other words, the Simulated Annealing algorithm will select a defect-free mapping while trying to minimize variations by nature.

For other algorithms to tolerate both variation and defects, the proposed framework can be used as an example and similar approaches can be used for variation optimization while defect tolerance.

### 4.3 Extension for the Crossbar Arrays

A crossbar array is a set of cascaded crossbars where the outputs of a crossbar are connected to the inputs of another crossbar. Therefore, for crossbar arrays, not only the optimization of a single crossbar, but also the optimization of the whole array is required. Since changing the output (input) mapping of a crossbar affects the input (output) mapping of the adjacent crossbars, individual crossbars cannot be mapped independently. Here, in this study, we also recommend a method to solve this problem.

Similar to Simulated Annealing algorithm, we can first determine a random crossbar. Then, we can try to optimize this crossbar based on IMV, OMV, or IMV/OMV. It should be noted that any move in a crossbar will also affect the other adjacent crossbars. This algorithm can be repeated for any arbitrary number.

```
begin  
   $rc = \text{random}(1..N)$   
   $rm = \text{random}(1..3)$   
  if ( $rm == 1$ ) then  
     $\lfloor \text{Optimize}(\text{crossbar}_{rc}, IMV)$   
  else if ( $rm == 2$ ) then  
     $\lfloor \text{Optimize}(\text{crossbar}_{rc}, OMV)$   
  else if ( $rm == 3$ ) then  
     $\lfloor \text{Optimize}(\text{crossbar}_{rc}, IMV, OMV)$   
end
```

**Algorithm 3:** Exhaustive Search algorithm for nano crossbars

# Chapter 5

## Experimental Studies

The proposed algorithms are implemented to show the effectiveness. In this chapter, the experimental setup and experimental results for both crossbar structures are explained.

### 5.1 Experimental Setup

For the comparison of the proposed framework for crossbars, we implemented the algorithm for both of the objectives proposed in Chapter 3. While different move (perturbation) choices will affect the reduction in the objectives, we applied different moves in the input assignment (IN), output assignment (OUT), and both of them (IN/OUT). Since it is infeasible to apply exhaustive search for large crossbars (requires  $n! \times m!$  different possible mappings), we first compared the proposed method (using Simulated Annealing) with exhaustive search only for small crossbars (with a size of  $6 \times 6$ ). Then, the effect of the constraints were investigated for larger crossbars. In addition, we extended the framework for both defect tolerance and crossbar arrays.

For the experimental studies, to represent the real case structure of function matrices, FMs were randomly generated based on an expected average value of *Crosspoint Usage Ratio* (CR) and *Output Usage Ratio* (OR). CR is the average percentage of crosspoints used in a crossbar in during typical logic mapping (e.g. 30% or 40%), and OR is the average percentage of vertical nanowires used for output mapping. During the experiments, only OR has been set as  $OR = 80\%$ . In addition to FMs,

VMs were generated with a Gaussian distribution with a mean of 50 ( $\mu = 50$ ) and a variance of 16 ( $\sigma = 16$ ) since delay variations typically follow normal distribution.

## 5.2 Experimental Study for FET Based Crossbars

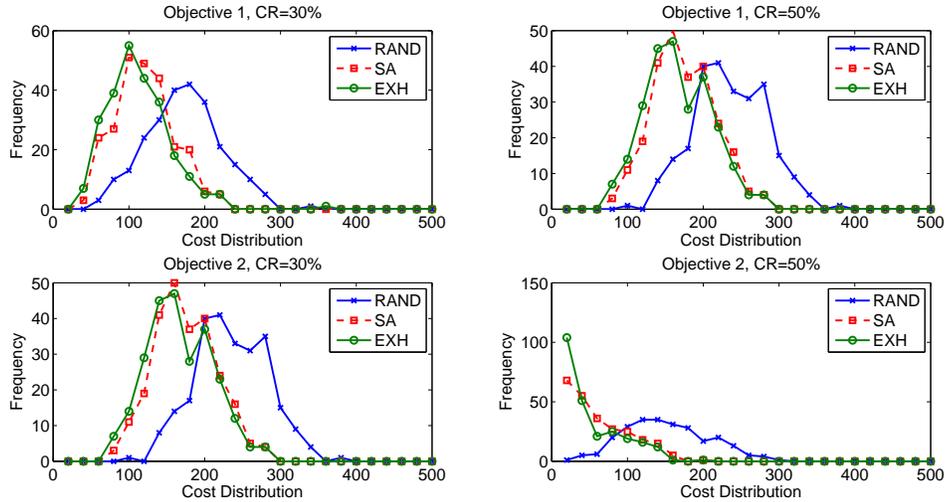
In order to evaluate the effectiveness for FET based crossbars, in terms of accuracy and runtime reduction, we have compared the proposed method against an exact method based on exhaustive search. The exhaustive method uses all the permutations to find the exact solution. Therefore, for an  $n \times m$  crossbar, exhaustive search requires  $n! \times m!$  iterations. Since exhaustive method is only tractable for very small crossbars, we have presented results for only  $6 \times 6$  crossbars.

While comparing the exhaustive search and Simulated Annealing, 250 pairs of FM and VM are generated based on the constraints mentioned in Section 5.1. Table 5.1 presents the results for *Variation Unaware Mapping* in which the first possible mapping is used (without consideration of VM) for comparison purposes (i.e. random mapping (RAND)), exhaustive search (EXH), and the proposed Simulated Annealing (SA). For this comparison, both of the objectives have been focused and during the experiments OR is chosen as  $OR = 80\%$  under different CR constraints. The cost objectives are presented as a percentage overhead with respect to the exact method as follows. The normalized average results presenting the misoptimization were gathered for SA as  $\frac{delay(SA) - delay(EXH)}{delay(EXH)}$  and for RAND,  $\frac{delay(RAND) - delay(EXH)}{delay(EXH)}$ . In addition, the runtime overheads were calculated using  $\frac{time(EXH)}{time(SA)}$ .

TABLE 5.1: Comparison of Variation unaware mapping (random, RAND), exact method (EXH), and Simulated Annealing (SA)

	Objective 1			Objective 2		
	RAND	EXH	SA	RAND	EXH	SA
Inaccuracy (CR=30%)	47.03%	–	8.63%	205.34%	–	37.34%
Inaccuracy (CR=50%)	36.60%	–	4.19%	170.67%	–	26.88%
Runtime Overhead	–	320x	1	–	320x	1

In addition to the average normalized results, the histogram for the costs are presented in Figure 5.1. In x-axis, the cost distribution is shown where for a FET based crossbar, cost is calculated by summing the used crosspoint VM values. In y-axis, the frequency (the number of occurrences) of each cost is shown.

FIGURE 5.1: The histogram of costs for a  $6 \times 6$  FET based crossbar.

As the average results (Table 5.1) present, the proposed framework is useful especially for Objective 1. The results obtained using SA and EXH are very close with respect to RAND, especially considering a performance increase of 320x. In addition, the histograms for these experiments (Figure 5.1) show that the cost distribution is getting closer to 0 cost by applying the proposed method. Also, the proposed method shows a close cost distribution to the cost distribution of EXH.

Additionally, we have studied the effect of constraints in moves (IN, OUT, IN/OUT) as well as the crosspoint usage ratio on the optimization results for larger FET based crossbars. Table 5.2 presents the average effect of constrained moves in the final cost functions for two optimization objectives, for various values of crosspoint usage ratio (CR) in  $16 \times 16$  crossbars and Figure 5.2 shows the histogram of the Objective 1 where Figure 5.3 shows the histogram of the Objective 2. In these experiments, 1000 random VM and FM are generated using the same idea expressed above. The average results show the reduction in unaware mapping using  $\frac{\text{delay}(SA) - \text{delay}(RAND)}{RAND}$  (i.e. negative values show that the cost is reduced).

TABLE 5.2: Constrained vs. unconstrained optimizations

CR	Objective 1			Objective 2		
	IN	OUT	IN/OUT	IN	OUT	IN/OUT
30%	-16.57%	-21.68%	-27.74%	-31.21%	-42.25%	-53.54%
50%	-10.85%	-18.62%	-22.25%	-32.05%	-50.39%	-61.71%

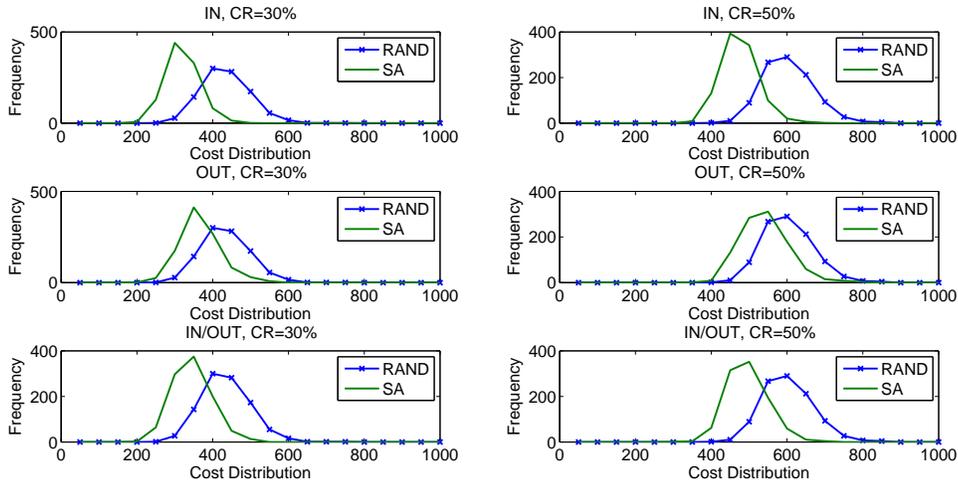


FIGURE 5.2: The histogram of costs (Objective 1) for a  $16 \times 16$  FET based crossbar.

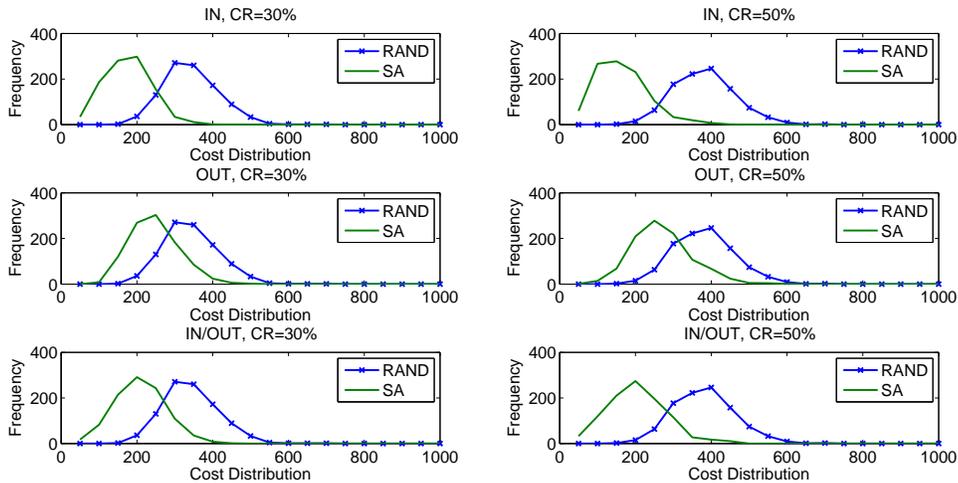


FIGURE 5.3: The histogram of costs (Objective 2) for a  $16 \times 16$  FET based crossbar.

It should be noted that the best optimization is gathered using the moves IN/OUT (moves in both input and output mapping vectors). Since the OR is not 100% meaning that not all outputs are used, there are some spare nanowires for output moves which gives us better optimization than input moves when output moves are used. The importance of this approach can be better understood with crossbar arrays. While for crossbar arrays, the outputs of crossbar (i) is the inputs of the next crossbar (i+1), moves cannot be decided independently. Therefore, not only the movements of IN/OUT, but also only IN and only OR should be effective. The experiments show that these methods are effective for the optimization of crossbars.

Moreover, the proposed method has also been investigated for various sizes of crossbars. The method has been compared for both average cost reduction ( $\frac{\text{delay}(SA) - \text{delay}(RAND)}{\text{delay}(RAND)}$ ) and time comparison (i.e. time for the proposed method in seconds). The results are shown in Figure 5.4. It should be noted that while crossbar size increases the cost reduction reduces since it gets more difficult to obtain a better mapping. In addition, the increase in runtime depends on the size of crossbars ( $n$  and  $m$ ) as well as the inner loop increase (as mentioned Section 4.2).

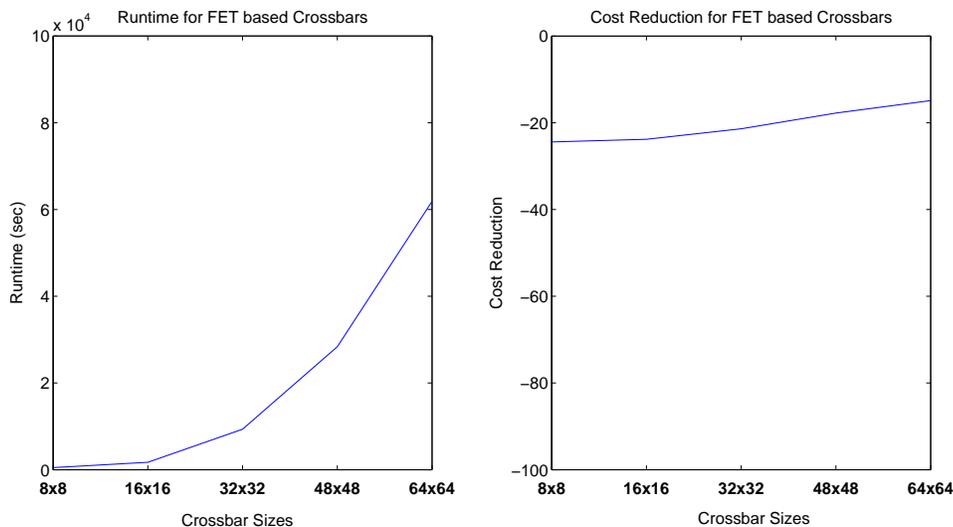


FIGURE 5.4: The comparison of runtime and cost reduction (for Optimization 1) for various sizes of crossbars.

Furthermore, we present experimental results for the augmentation of the proposed technique to handle defects. Since the Simulated Annealing is based on random perturbations, whenever the new mapping uses any defective crosspoints (resulting in  $\infty$  cost), it will automatically be rejected. However, when defect density and crosspoint usage ratio are high, all random perturbations may result in defective mapping. In that case, at the end of the execution of Simulated Annealing, no defect-free mapping can be obtained. To evaluate the success of this approach for defect and variation tolerance, we measure ratio of the number of cases where a defect-free mapping with reduced variation can successfully be generated at the end of the execution of Simulated Annealing, over the total number of simulations.

Different crossbar sizes of  $8 \times 8$  and  $16 \times 16$  have been considered. For each data point, 1000 pairs of FM and VM were generated and defects based on defect density of  $d = 5\%$  and  $d = 10\%$  were randomly injected in VM (as entries with  $\infty$  values). The results for various defect densities are provided in Table 5.3.

TABLE 5.3: Success rate in defect-free mapping for diode based crossbars

defect density = 5%		
Size	RAND	SA
8×8, CR = 30%	54.8%	100%
8×8, CR = 50%	28.8%	100%
16×16, CR = 30%	5.6%	100%
16×16, CR = 30%	0.8%	85%
defect density = 10%		
Size	RAND	SA
8×8, CR = 30%	23.8%	100%
8×8, CR = 50%	78.0%	99.3%
16×16, CR = 30%	0.4%	94.2%
16×16, CR = 30%	0.0%	30.2%

In this experiment, it is shown that the success rate drops with larger sizes of crossbars as well as higher crosspoint usage ratio. This is because in an  $n \times n$  crossbar with defect density  $d$  and crosspoint usage ratio  $c$ , the probability that a given mapping is defect free can be formulated as  $(\lceil n^2 c \rceil)^{1-d}$ . As this probability drops, the success ratio of Simulated Annealing reduces accordingly. However, when compared to variation unaware mapping (RAND), Simulated Annealing (SA) is still able to find a defect free mapping by using the moves both input and output mapping vectors.

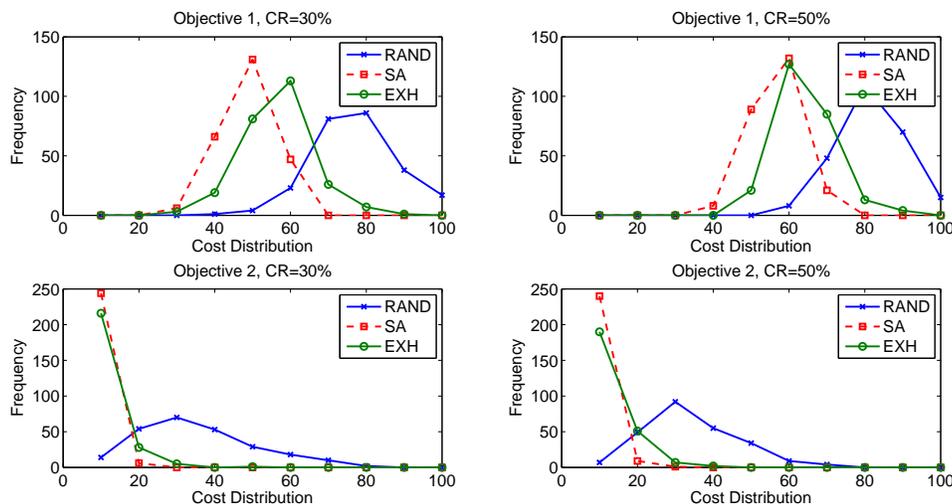
### 5.3 Experimental Study for Diode Based Crossbars

In addition to FET based, the effectiveness of the proposed method has been studied for diode based crossbars. While exhaustive search is only tractable for small crossbars, here, we present the results for only  $6 \times 6$  crossbars.

While comparing the exhaustive search and Simulated Annealing, 250 pairs of FM and VM are generated based on the constraints mentioned in Section 5.1. Table 5.4 presents the average results for variation unaware mapping, exhaustive search, and Simulated Annealing for cost minimization of both objectives. Furthermore, the cost histogram for diode based crossbar using RAND, SA, and EXH are shown in Figure 5.5. The results present that this technique is also applicable for diode based crossbars.

TABLE 5.4: Comparison of Variation unaware mapping (random, RAND), exact method (EXH), and Simulated Annealing (SA)

	Objective 1			Objective 2		
	RAND	EXH	SA	RAND	EXH	SA
Inaccuracy (CR=30%)	39.73%	–	17.64%	425.85%	–	104.48%
Inaccuracy (CR=50%)	29.32%	–	13.85%	283.53%	–	77.32%

FIGURE 5.5: The histogram of costs for a  $6 \times 6$  FET based crossbar.

In addition, the effect of the constraints in moves have been studied also for diode based crossbars. Table 5.5 presents the effect of constrained moves in the final cost functions, for  $16 \times 16$  crossbars using different crosspoint usage ratios. In the experiment, 1000 random VM and FM are generated using the same idea expressed above. The results are normalized to the cost of unconstrained (IN/OUT) moves as shown before. In addition to the normalized results, the cost distributions are shown in Figure 5.6 for Optimization 1 and Figure 5.7 for Optimization 2. The results present that the best optimization is gathered using the moves both in input and in output (IN/OUT). The histograms also show that the distribution of the cost is reduced to a better cost for both objectives (Objective 1 and Objective 2).

TABLE 5.5: Constrained vs. unconstrained optimizations

CR	Objective 1			Objective 2		
	IN	OUT	IN/OUT	IN	OUT	IN/OUT
30%	-19.81%	-10.43%	-29.29%	-40.85%	-55.82 %	-84.75%
50%	-15.68%	-7.73%	-23.81%	-78.15%	-35.68%	-78.15%

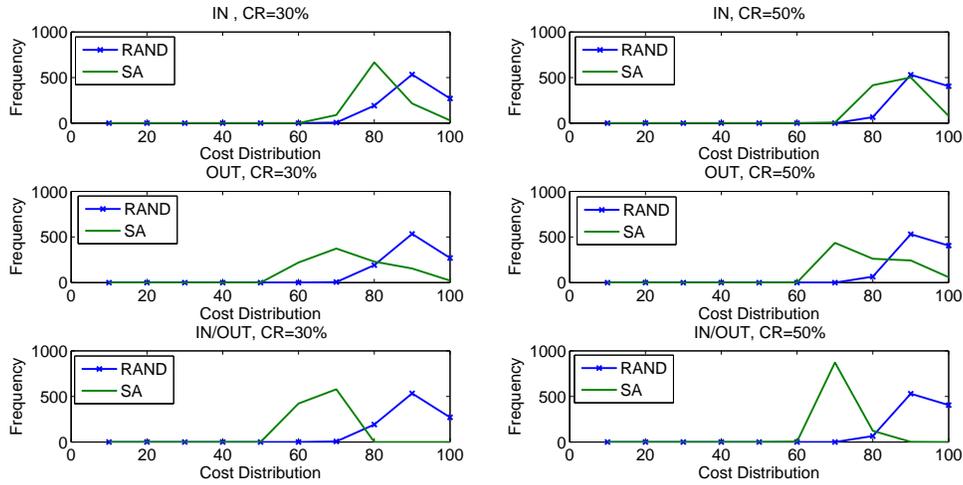


FIGURE 5.6: The histogram of costs (Objective 1) for a  $16 \times 16$  diode based crossbar.

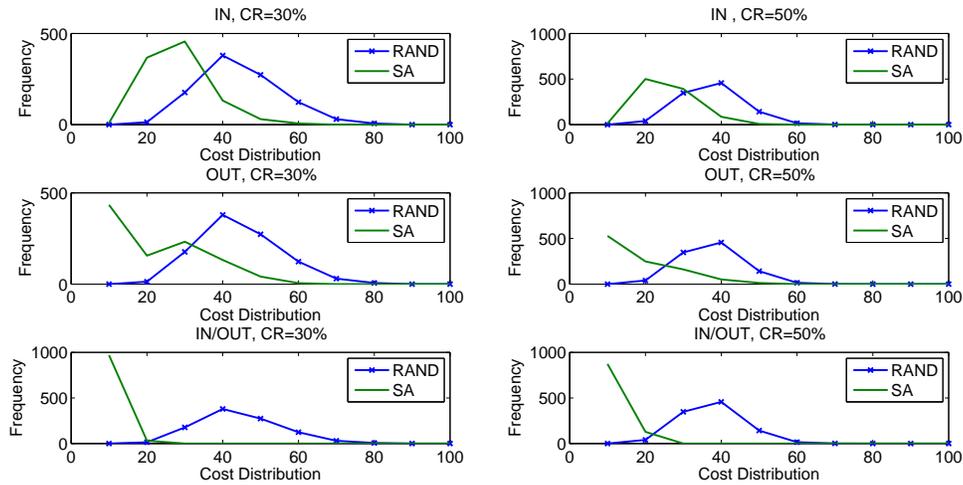


FIGURE 5.7: The histogram of costs (Objective 2) for a  $16 \times 16$  diode based crossbar.

Moreover, the runtime and the average cost reduction for various sizes of crossbars have also been investigated for diode based crossbars. The results provided in Figure 5.8 show the similar results with FET based crossbars: the runtime increases with the size of crossbars (based on  $n$  and  $m$  for  $n \times m$  crossbar and based on inner loop calculation) and the cost reduction decreases with the crossbar sizes.

In addition to the optimization of diode based crossbars, defect tolerance is investigated in Table 5.6. For the experiments 1000 FMs and VMs were created using the idea mentioned in Section 5.1.

It should be noted that the proposed method is suitable also for defect tolerance for

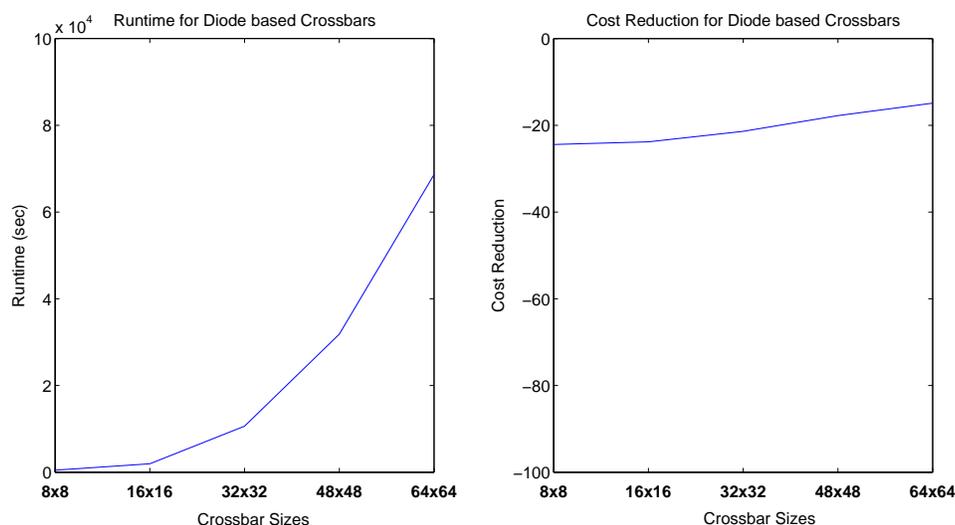


FIGURE 5.8: Runtime and cost reduction (for Optimization 1) comparison for various sizes of diode based crossbars.

TABLE 5.6: Success rate in defect-free mapping for diode based crossbars

defect density = 5%		
Size	RAND	SA
8×8, CR = 30%	54.8%	100%
8×8, CR = 50%	28.8%	100%
16×16, CR = 30%	5.6%	100%
16×16, CR = 30%	0.8%	97%
defect density = 10%		
Size	RAND	SA
8×8, CR = 30%	23.8%	100%
8×8, CR = 50%	78.0%	99.1%
16×16, CR = 30%	0.4%	95.2%
16×16, CR = 30%	0.0%	30.1%

diode based crossbars. The results in Table 5.6 show that while RAND mapping fails (0% of successful mapping), the proposed method is still able to find a suitable mapping for the diode based crossbars.

## 5.4 Experimental Study for Crossbar Arrays

In addition to the optimization of crossbars, crossbar arrays should be considered. In this section, the experimental results for crossbar arrays are presented. For the experiments, the proposed algorithm in Section 4.3 based on Simulated Annealing

was used. It should be noted that the algorithm first decides movement (IN, OUT or IN/OUT) and then tries to optimize the crossbar array.

The Table 5.7 shows the cost reduction of the proposed method with Simulated Annealing. For these experiments, 1000 were run for both FET and diode based crossbar arrays. VMs were generated based on a Gaussian distribution with a mean of 50 and variance of 16. FMs were generated randomly based on  $CR = 30$  and  $OR$  was chosen as  $OR = 80\%$ . The crossbar arrays were cascaded using 10 crossbars with a size of  $16 \times 16$ . The results show the comparison as the cost with respect to variation unaware mapping. The negative percentages show that the proposed technique is able to reduce the delay.

TABLE 5.7: Cost comparison for crossbar arrays (10 16x16 crossbars are cascaded)

	CR=30%		CR=50%	
	RAND	SA	RAND	SA
FET based	–	-19%	–	-19%
Diode based	–	-14%	–	-12%

In these experiments, comparison is shown as normalized values of SA to RAND values ( $\frac{\text{delay}(SA) - \text{delay}(RAND)}{\text{delay}(RAND)}$ ). The proposed method fits to the extension of crossbar arrays. However, while  $CR$  increases, the cost reduction decreases since it gets more difficult to obtain a more suitable configuration.

# Chapter 6

## Conclusions

With the extreme shrinking in feature size of CMOS based technologies, some issues are getting extremely important such as tremendous power dissipation, parasitic issues, direct tunneling, increase in manufacturing cost, etc. So far, these issues have been successfully handled using conventional top-down manufacturing strategy. However, for future systems, it seems that handling these issues will not be feasible. On the other hand, a new arising technology scheme, bottom-up approach, have been presented where nanowires, Carbon nanotubes, other devices can be manufactured beyond the top-down manufacturing limits.

Even though emerging nanoscale devices using bottom-up approach seems exciting for future systems, the benefits come with a cost. The inherent lack of control in stochastic bottom-up self-assembly nanofabrication as well as atomic device size makes high defect rates and increased variations as major challenges for systems built using emerging nanoscale devices. Even defect-free nanodevices (nanowires, crosspoints, and contacts), by nature, exhibit extreme variations in device characteristics.

In this study, we have presented a variation tolerant logic mapping technique for different crossbar structures, namely for FET based and diode based nano crossbars. We took advantage of programmability and interchangeability of nano architectures to be able to map the function while tolerating variations (delay). We formulated this optimization problem using Simulated Annealing. A path delay testing procedure to obtain delay/variation values was presented. Next, we extended the framework for defect tolerance. While logic function implementations require extremely large

crossbars, we defined crossbar arrays where crossbars are cascaded to each other. We visited the crossbar arrays using the proposed framework.

In order to compare the effectiveness of the proposed techniques, experimental results are presented. During the experiments, both FET and diode based crossbars were studied. The experimental results show that the proposed techniques are able to reduce the effect of variation under different constraints. When compared to the exhaustive search (the possible minimum cost) for small crossbars (with a size of  $6 \times 6$ ), Simulated Annealing succeeds to optimize the circuits as effective as exhaustive search. For larger crossbars, Simulated Annealing is able to optimize crossbars (both FET and diode based crossbars) as well. When defect tolerance is considered, the experimental studies show that the proposed method fits very good. It should be noted that when compared to exhaustive search methods, the proposed framework is 3-4 orders of magnitude faster for every cases. Therefore, it can be applied for both variation and defect tolerance. When the crossbar arrays are considered, the framework is applicable especially for low  $CR$  values.

# Bibliography

- [1] Y. Cui and C.M. Lieber. Functional nanoscale electronic devices assembled using silicon nanowire building blocks. *Science*, 291(5505):851–853, 2001.
- [2] M.M. Ziegler and M.R. Stan. Cmos/nano co-design for crossbar-based molecular electronic systems. *IEEE Transactions on Nanotechnology*, 2(4):217–230, Dec. 2003.
- [3] X. Ma, D.B. Strukov, J.H. Lee, and K.K. Likharev. Afterlife for silicon: Cmol circuit architectures. In *IEEE Conference on Nanotechnology*, pages 175–178 vol. 1, July 2005.
- [4] S.C. Goldstein and M. Budiu. Nanofabrics: Spatial computing using molecular electronics. In *Proc. Annual International Symposium on Computer Architecture*, pages 178–189, 2001.
- [5] A. DeHon and M.J. Wilson. Nanowire-based sublithographic programmable logic arrays. In *Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, pages 123–132, New York, NY, USA, 2004. ACM.
- [6] Teng Wang, P. Narayanan, M. Leuchtenburg, and C.A. Moritz. Nasics: A nanoscale fabric for nanoscale microprocessors. In *Nanoelectronics Conference, 2008. INEC 2008. 2nd IEEE International*, pages 989–994, March 2008.
- [7] Greg Snider, Philip Kuekes, and R Stanley Williams. Cmos-like logic in defective, nanoscale crossbars. *Nanotechnology*, 15(8):881–891, 2004.
- [8] W. Rao, A. Orailoglu, and R. Karri. Interactive presentation: Logic level fault tolerance approaches targeting nanoelectronics plas. In *Proc. Conference on Design, Automation and Test in Europe (DATE)*, pages 865–869, San Jose, CA, USA, 2007. EDA Consortium.

- 
- [9] S. Luryi, J. Xu, and A. Zaslavsky. *Future Trends in Microelectronics: Up the Nano Creek*. Wiley-Interscience Publication, Hoboken, NJ, 2007.
- [10] R.I. Bahar. Trends and future directions in nano structure based computing and fabrication. In *International Conference on Computer Design (ICCD)*, pages 522–527, Oct. 2006.
- [11] H. Iwai. Cmos scaling toward sub-10nm regime. In *IEEE International Symposium on Electron Devices for Microwave and Optoelectronic Applications (EDMO)*, pages 30–34, Nov. 2003.
- [12] S. C. Goldstein. The impact of the nanoscale on computing systems. In *ICCAD '05: Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, pages 655–661, Washington, DC, USA, 2005. IEEE Computer Society.
- [13] W. Lu and C.M. Lieber. Nanoelectronics from the bottom up. *Nat Mater*, 6(11):841–850, November 2007.
- [14] D. Whang, S. Jin, Y. Wu, and C.M. Lieber. Large-scale hierarchical organization of nanowire arrays for integrated nanosystems. *Nano Letters*, 3(9):1255–1259, 2003.
- [15] A. Bachtold, P. Hadley, T. Nakanishi, and C. Dekker. Logic Circuits with Carbon Nanotube Transistors. *Science*, 294(5545):1317–1320, 2001.
- [16] J.R. Heath, P.J. Kuekes, G.S. Snider, and R.S. Williams. A defect-tolerant computer architecture: Opportunities for nanotechnology. *Science*, 280(5370):1716–1721, 1998.
- [17] M. Butts, A. DeHon, and S.C. Goldstein. Molecular electronics: Devices, systems and tools for gigagate, gigabit chips. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 433–440, Nov. 2002.
- [18] T. Rueckes, K. Kim, E. Joselevich, G.Y. Tseng, C. Cheung, and C.M. Lieber. Carbon Nanotube-Based Nonvolatile Random Access Memory for Molecular Computing. *Science*, 289(5476):94–97, 2000.
- [19] A.M. Morales and C.M. Lieber. A laser ablation method for the synthesis of crystalline semiconductor nanowires. *Science*, 279(5348):208–211, 1998.

- [20] Y. Cui, L.J. Lauhon, M.S. Gudiksen, J. Wang, and C.M. Lieber. Diameter-controlled synthesis of single-crystal silicon nanowires. *Applied Physics Letters*.
- [21] Y. Wu and P. Yang. Germanium nanowire growth via simple vapor transport. *Chemistry of Materials*, 12(3):605–607, 2000.
- [22] Y. Huang, X. Duan, Y. Cui, and C.M. Lieber. Gallium nitride nanowire nanodevices. *Nano Letters*, 2(2):101–104, 2002.
- [23] A. DeHon. Array-based architecture for fet-based, nanoscale electronics. *IEEE Transactions on Nanotechnology*, 2(1):23–32, Mar 2003.
- [24] C.P. Collier, E.W. Wong, M. Belohradsky, F.M. Raymo, J.F. Stoddart, P.J. Kuekes, R.S. Williams, and J.R. Heath. Electronically Configurable Molecular-Based Logic Gates. *Science*, 285(5426):391–394, 1999.
- [25] A. Dehon. Nanowire-based programmable architectures. *Journal on Emerging Technologies in Computing Systems*, 1(2):109–162, 2005.
- [26] G. Snider, P. Kuekes, T. Hogg, and R.S. Williams. Nanoelectronic architectures. *Applied Physics A: Materials Science & Processing*, 80:1183–1195, Mar 2005.
- [27] A. DeHon. Architecture approaching the atomic scale. In *European Solid State Device Research Conference ESSDERC*, pages 11–20, Sept. 2007.
- [28] A. DeHon, P. Lincoln, and J.E. Savage. Stochastic assembly of sublithographic nanoscale interfaces. *IEEE Transactions on Nanotechnology*, 2(3):165–174, Sept. 2003.
- [29] M.B. Tahoori. Application-independent defect tolerance of reconfigurable nanoarchitectures. *Journal on Emerging Technologies in Computing Systems*, 2(3):197–218, 2006.
- [30] M.B. Tahoori, N. Jha, and I. Bahar. *Testing Aspects of Nanotechnology Trends*, chapter 17, pages 791–833. Morgan Kaufmann Publishers, 2008.
- [31] M. Mishra and S.C. Goldstein. Defect tolerance at the end of the roadmap. volume 1, pages 1201 – 1210, 30-oct. 2, 2003.
- [32] F. Hasani and N. Masoumi. Crosstalk and delay optimization techniques for nano scale interconnects. In *International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 159–163, Sept. 2007.

- [33] A. Raychowdhury and K. Roy. Performance estimation of molecular crossbar architecture considering capacitive and inductive coupling between interconnects. In *IEEE Conference on Nanotechnology (IEEE-NANO)*, volume 1, pages 445–448 vol.2, Aug. 2003.
- [34] B. Gojman and A. DeHon. Vmatch: Using logical variation to counteract physical variation in bottom-up, nanoscale systems. pages 78–87, dec. 2009.
- [35] M.M. Ziegler and M.R. Stan. The cmos/nano interface from a circuits perspective.
- [36] S. Eachempati, A. Nieuwoudt, A. Gayasen, N. Vijaykrishnan, and Y. Massoud. Assessing carbon nanotube bundle interconnect for future fpga architectures. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6, April 2007.
- [37] A.M. Zaitsev, A.M. Levine, and S.H. Zaidi. Temperature and chemical sensors based on fib-written carbon nanowires. *IEEE Sensors Journal*, 8(6):849–856, June 2008.
- [38] Yu Huang, Xiangfeng Duan, Yi Cui, Lincoln J. Lauhon, Kyoung-Ha Kim, and Charles M. Lieber. Logic Gates and Computation from Assembled Nanowire Building Blocks. *Science*, 294(5545):1313–1317, 2001.
- [39] J. Chen, M. A. Reed, A. M. Rawlett, and J. M. Tour. Large On-Off Ratios and Negative Differential Resistance in a Molecular Electronic Device. *Science*, 286(5444):1550–1552, 1999.
- [40] Thomas Rueckes, Kyoung-Ha Kim, Ernesto Joselevich, Greg Y. Tseng, Chin-Li Cheung, and Charles M. Lieber. Carbon Nanotube-Based Nonvolatile Random Access Memory for Molecular Computing. *Science*, 289(5476):94–97, 2000.
- [41] P.P. Sotiriadis. Information storage capacity of crossbar switching networks. In *Proc. ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pages 45–49, New York, NY, USA, 2003. ACM.
- [42] Y. Chen, G. Jung, D.A.A. Ohlberg, X. Li, D.R. Stewart, J.O. Jeppesen, K.A. Nielsen, J.F. Stoddart, and R.S. Williams. Nanoscale molecular-switch crossbar circuits. *Nanotechnology*, 14(4):462–468.
- [43] A. Dehon. Array-based architecture for molecular electronics. In *Proc. Workshop on Non-Silicon Computation (NSC)*, 2001.

- [44] Teng Wang, Zhenghua Qi, and Csaba Andras Moritz. Opportunities and challenges in application-tuned circuits and architectures based on nanodevices. In *CF '04: Proceedings of the 1st conference on Computing frontiers*, pages 503–511, New York, NY, USA, 2004. ACM.
- [45] J. Huang, M.B. Tahoori, and F. Lombardi. On the defect tolerance of nanoscale two-dimensional crossbars. In *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 96–104, Oct. 2004.
- [46] N. Seoane, A. Martinez, A.R. Brown, J.R. Barker, and A. Asenov. Current variability in si nanowire mosfets due to random dopants in the source/drain regions: A fully 3-d negf simulation study. *Electron Devices, IEEE Transactions on*, 56(7):1388 –1395, july 2009.
- [47] R. Amerson, R.J. Carter, W.B. Culbertson, P. Kuekes, and G. Snider. Teramac-configurable custom computing. *Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 0:0032, 1995.
- [48] H. Naeimi and A. DeHon. A greedy algorithm for tolerating defective crosspoints in nanopla design. In *Proc. IEEE International Conference on Field-Programmable Technology*, pages 49–56, Dec. 2004.
- [49] M.B. Tahoori. A mapping algorithm for defect-tolerance of reconfigurable nanoarchitectures. pages 668 – 672, nov. 2005.
- [50] W. Rao, A. Orailoglu, and R. Karri. Logic mapping in crossbar-based nanoarchitectures. *IEEE Design & Test of Computers*, 26(1):68–77, Jan.-Feb. 2009.
- [51] R.M.P. Rad and M. Tehranipoor. A reconfiguration-based defect tolerance method for nanoscale devices. pages 107 –118, oct. 2006.
- [52] M. Haykel Ben Jamaa, Yusuf Leblebici, and Giovanni De Micheli. Decoding nanowire arrays fabricated with the multi-spacer patterning technique. In *DAC '09: Proceedings of the 46th Annual Design Automation Conference*, pages 77–82, New York, NY, USA, 2009. ACM.
- [53] Jie Deng, Albert Lin, Gordon C. Wan, and H.-S. Philip Wong. Carbon nanotube transistor compact model for circuit design and performance optimization. *J. Emerg. Technol. Comput. Syst.*, 4(2):1–20, 2008.
- [54] P. Narayanan M. Leuchtenburg Yao Guo C. Dezan C.A. Moritz, Teng Wang and M. Bennaser. Fault-tolerant nanoscale processors on semiconductor

- nanowire grids. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 54(11):2422–2437, Nov. 2007.
- [55] Chen Dong, Scott Chilstedt, and Deming Chen. Fpcna: a field programmable carbon nanotube array. In *FPGA '09: Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays*, pages 161–170, New York, NY, USA, 2009. ACM.
- [56] S. Gerez. *Algorithms for VLSI Design Automation*. John Wiley & Sons, Inc., Chichester, UK, 1999.