

Inherited Redundancy and Configurability Utilization for Repairing Nanowire Crossbars with Clustered Defects

Yadunandana Yellambalase¹, Minsu Choi¹ and Yong-Bin Kim²

¹Dept of ECE, University of Missouri-Rolla, Rolla, MO 65409-0040, USA
{ypymy9, choim}@umr.edu

²Dept of ECE, Northeastern University, Boston, MA 02115, USA
ybk@ece.neu.edu

Abstract

With the recent development of nanoscale materials and assembly techniques, it is envisioned to build high-density reconfigurable systems which have never been achieved by the photolithography. Various reconfigurable architectures have been proposed based on nanowire crossbar structure as the primitive building block. Unfortunately, high-density systems consisting of nanometer-scale elements are likely to have many imperfections and variations; thus, defect-tolerance is considered as one of the most exigent challenges. In this paper, we evaluate three different logic mapping algorithms with defect avoidance to circumvent clustered defective crosspoints in nanowire reconfigurable crossbar architectures. The effectiveness of inherited redundancy and configurability utilization is demonstrated through extensive parametric simulations.

1 Introduction

Recently, numerous nanoscale logic devices have been proposed based on nanoscale components such as carbon nanotubes (CNTs) and silicon nanowires (SiNWs); computing architectures are also being proposed using them as primitive building blocks. Unlike CMOS, chemically-assembled nanoscale components (such as CNTs and SiNWs) are unlikely to be used to construct complex aperiodic structures [1].

One of the most promising computational nanotechnologies is the crossbar-based architecture, a two-dimensional array (nanoarray) formed by the intersection of two orthogonal sets of parallel and uniformly-spaced nanometer-sized wires, such as carbon nanotubes (CNTs) and silicon nanowires (SiNWs). Experiments have shown that such nanoscale wires can be aligned to construct an array with nanometer-scale spacing using a form of directed self-assembly the formed crosspoints of nanoscale wires can be used as programmable diodes, memory cells or FETs (Field-Effect Transistors) [2, 3].

Nanoscale crossbar systems offer both an opportunity and a challenge. The opportunity is to achieve ultra-high density which has never been achieved by photolithography (a density of 10^{11} crosspoints per cm^2 has been achieved [2]). The challenge is to make them *defect tolerant*, since high-density systems consisting of nanometer-scale elements assembled in a bottom-up manner are likely to have many imperfections. A computing or storage system designed on conventional defect basis and top-down lithographic manufacturing would not

be practical [4]. Ultra-high density fabrication could potentially be very inexpensive if researchers can actualize a chemical self-assembly, but such a circuit would require laborious testing, repair and reconfiguration processes, implying significant overhead costs. So, finding the most cost-effective post-fabrication logic mapping solution is desired.

In this paper, nanoscale crossbar systems with clustered defects are considered to be repaired by defect avoidance logic mapping. Newly assembled nanowire crossbars are to be tested to locate defective crosspoints. Such defective crosspoints cannot be programmed to "closed" state; therefore, should be avoided when a netlist is mapped onto them. Three simple repair algorithms are considered and analyzed in terms of repair performance to demonstrate that proper utilization of inherited redundancy and configurability of the nanowire crossbar systems is very effective in improving the overall repairability.

2 Preliminaries & Review

2.1 Bottom-Up Paradigm for Nanowire Crossbar Assembly

CNTs and SiNWs are the most promising building blocks for nanoscale computing systems. Unfortunately, synthesis of such nanowires and high-density integration of devices and systems based on nanowires are fully different from conventional top-down lithographic fabrication techniques, because such nanowires must be synthesized first, then assembled into functional devices and systems in a bottom-up manner.

There are two key bottom-up assembly techniques for NW building blocks: electrical field directed assembly and fluidic flow directed assembly [5]. In the electrical field directed assembly technique, applied electrical fields are used to attract and align NWs using their highly anisotropic structures and large polarizabilities. By changing the electrical field direction with sequential NW solutions, the alignment can be carried out in a layer-by-layer fashion to produce crossed NW junctions. Although this technique represents the first demonstration of actual assembly of 1D nano building blocks, it also has limitations. First, microelectrode arrays should be fabricated by conventional lithography to produce aligning electronic fields. Second, there is the deleterious effect of fringing electronic fields at the submicron length scales.

Lieber et al, also have proposed more effective ways to hierarchically assemble 1D nanostructures into integrated nanosystems based on fluidic flow and the well-established Langmuir-Blodgett (LB) method. In this method, SiNWs or CNTs can be aligned by passing a suspension of NWs through microfluidic channel structures. Ordered monolayers are formed over a large area and transferred to substrates by the LB method. This ordering and transfer processes can be repeated multiple times to yield more complex hierarchically-assembled nanosystems. It has been demonstrated that virtually all of the NWs are aligned along the flow direction. Alternating the flow in orthogonal directions in a two-phase assembly process results in crossbar structures in high yield. Experiments have demonstrated that crossbars extending over 100s of microns on a substrate with only 100s of nanometers pitch between individual crosspoints are obtained.

2.2 Nanowire Crossbar Architectures

In nanowire crossbars, one or more crosspoints can be grouped together to form a memory or logic device. Using the nanoscale switching, configurable OR planes can be assembled,

with connected wires acting as low-resistance p-n-junctions and distant wires isolated by high resistance. Similarly, configurable NOR planes can be assembled. Since {OR, NOR} is a complete logic set, any digital logic circuits can be implemented, if sufficiently interconnected OR and NOR planes are given.

In [6], Dehon et al. have proposed a method to build sublithographic PLAs (Programmable Logic Arrays) using nanowires (NWs) and to interconnect PLAs to form large arrays. In this architecture, the PLAs are built upon programmable crosspoint diodes and by using lithographic scale address decoder that can be used to address individual nanowires. Also, by using some semi-static structure and applying a sequence of timing control signals, the PLAs can perform buffer and inverter functions as well as global clock control.

There is another nanowire crossbar architecture called the NanoFabric [1]. The NanoFabric architecture has nanoscale crossbars and supporting microscale components and facilitates directed nanoscale self-assembly paradigm.

2.3 Advanced Lithography for Crossbar Architecture

There are considerable on-going research and development efforts to fabricate nanoscale crossbar-based circuits and systems using advanced lithography. For example, researchers at HP successfully fabricated 8×8 (i.e., 64 bits) crossbar memory arrays using nano-imprint lithography [7]. Non-volatile bistable Rotaxane molecules are sandwiched between two orthogonal metal wires to form a non-volatile memory cell. However, each junction area is $40\text{nm} \times 40\text{nm}$ and therefore it is not considered as "true nanometer-scale device". Also, about 75% of the memory cells are tested as functional and all the other cells are either stuck-open or stuck-closed: therefore, not functional. It is also easily predictable that even higher defect density will be induced as the size of such device scales down. Therefore, enhancement of lithographic resolution and defect tolerance are two key challenges that advanced lithographic fabrication methods, such as nano-imprint lithography, face.

3 Nanoscale Reconfigurable Crossbar Repair Problem

Aforementioned nanowire crossbar architectures share common characteristics - they support nanoscale manufacturing paradigm via simple homogeneous periodic structures and reconfigurability for post-fabrication design mapping. Due to imperfections and variations in nanoscale manufacturing, high defect densities are anticipated. Thus, such defects should be located when tested and avoided when the given design is mapped. In this section, a general model for nanoscale crossbar systems will be proposed and the defect avoidance logic mapping problem will be formally defined based on it.

The programmable diode crossbar structure (namely, logic block) supports flexible utilization of its crosspoints through reconfiguration, even though the defect rate is anticipated as high. The internal lines in the logic blocks are completely interchangeable and the switch block can provide the flexible connections between inputs and outputs signals of adjacent logic blocks. It is possible to utilize such flexibility and reconfigurability to get around defective crosspoints. The term "*repair*" used in this paper refers to the logic mapping procedure with defect avoidance.

A few papers discuss the testing methods for crossbar architectures, including [8] and [9]. The basic idea is first use an external tester to test a certain area of the crossbar

chip under test. Then program that tested area to an internal tester which can be used to test the rest part of the chip. Except using the external tester, all the rest testing can be viewed as built-in self test. Also, this method could be done in parallel so that the test speed is fast. The defect models are different from CMOS system's. So, those defects are often categorized as: 1) defects in programmable crosspoints and 2) defects in nanowires [10]. Nanowires with short or break can be easily screened out and all crosspoints fall into those nanowires simply are not usable. Physically, defects in programmable crosspoints are due to the structure of the junctions, which are bistable molecules between two layers of nanowires. Reprogrammability of a crosspoint comes from the bistable property of the molecules located in the crosspoint area. If there are not enough molecules at a certain crosspoint then that junction may not be able to be programmed to a "closed" state, or the "closed" state may have higher resistance than the threshold from speculation which enable the whole NanoFabric system operate properly. If the crosspoint cannot be programmed to "open" status which means the two crossing nanowires are always connected, like a short occurred in those two nanowires, we should treat these as nanowire defects rather than junction defects. Those crosspoints which cannot be programmed into a "closed" state, but can be programmed into a "open" state is referred to as the *non-programmable crosspoints*. Although the non-programmable crosspoints are defective, they do not affect the other crosspoints in the rows and columns associated with them.

To map the given physical design onto the reconfigurable crossbar system, the logic synthesizer generates a netlist which allocates some of the nanowires as inputs, some as outputs and also indicate which crosspoints need to be set to "closed" state. In this paper, 50×50 matrix F is used to represent the set of functions that are needed to program the given reconfigurable crossbar system. In F , columns represent input terms and rows represent OR functions based on the input terms. If the node value is 1, this means the corresponding crosspoint is needed to be programmed to "closed" state and the crosspoint that should be programmed to "closed" state is called "on-input". If the node value is 0, the crosspoint should be left as "open" state. Because of the inherent reconfigurability of the crossbar architecture, the order of rows and columns can be rearranged if coupled switch blocks are reconfigured accordingly.

After testing, a defect map which indicate the locations of the defective crosspoints can be constructed. Another $N \times (N + 20)$ matrix D is to represent the defect map. For the location which represent a non-programmable crosspoint, 1 is allocated to indicate the defect. Otherwise, 0 is allocated to indicate the corresponding crosspoint location is programmable. An OR function from F (i.e., one row from F) can be assigned to a physical nanowire row if and only if each of the on-inputs of the OR function has a corresponding non-defective crosspoint on the physical nanowire. In summary:

1. Every node need to be programmed to "closed" state must fall into a non-defective crosspoint.
2. Every node that is "unused" could fall into either a non-defective crosspoint or non-programmable crosspoint.

Even though some nanowires have defective crosspoints, some OR functions can be successfully mapped to them, if on-inputs do not fall into non-programmable crosspoints. The challenge is to find a successful mapping while minimizing overhead costs induced by the defect avoidance mapping procedure.

4 Nanowire Crossbar with Clustered Defects

A defect layout of a nanowire crossbar was generated in the form of a matrix. To indicate a non-programmable crosspoint at a certain location, 1 was allocated and a programmable crosspoint was marked with 0. In fabricating nanowire crossbar structures, physical imperfections may result in defective crosspoints in clusters. Thus, cluster defect model was considered in this paper rather than the random defect model. The defect maps were randomly generated in the form of clusters with negative binomial distribution as described by Stapper [11]. The probability of introducing a fault into a given cross-point during a time interval Δt of the manufacturing process is given by: $p(\Delta t|k, l_1, l_2, \dots, l_n) = c(x, y) + bk + \sum_{l=0}^n b_l l_i$.

where $c(x, y)$ is the susceptibility function, k is the number of defects already present in the chip, the index i pertains to the adjacent and other neighboring crosspoints, n indicates the number of neighboring crosspoints considered, b is the global cluster factor, b_i is the local cluster factor and l_i is the number of faults that occur on neighboring circuit area. In Figure 1, a defect map with clustered defects is shown. White dots represent non-programmable crosspoints (i.e., defects) while black dots represent programmable crosspoints. In the following section, three simple logic mapping algorithms with defect avoidance will be discussed.

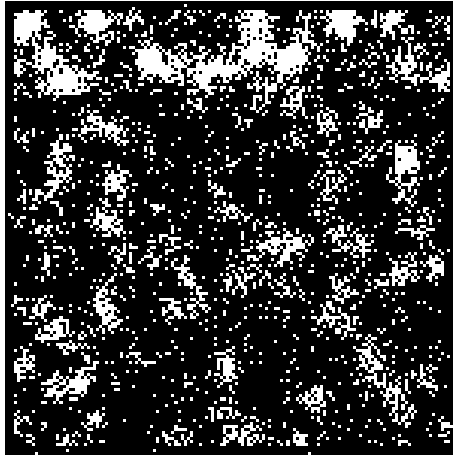


Figure 1. Defect map with clustered defects.

5 Repair Algorithms using Inherited Redundancy and Configurability

For the given repair problem, the row-wise repair algorithm is trivial, since the greedy algorithm always results in the best possible solution. However, for the two-dimensional repair problem which is similar to two-dimensional memory repair problem, only the brute-force algorithm can guarantee the optimal solution, simply because the given problem is NP-complete. So, faster algorithms that can be used to find sub-optimal solution at reasonable overhead are usually pursued. In this paper, two new techniques are considered

to improve the overall repair performance. The first one is to reorder the inputs to the crossbar columns to increase the probability of successful matching between OR functions and nanowire rows. The second one is to utilize unused columns as redundancy. For the given reconfigurable crossbar repair problem, the following three repair algorithms are extensively evaluated:

1. 1D Greedy Repair: For each OR function in F is selected from top to bottom. Then, sequential search on nanowire rows in D is performed to find a successful matching. This procedure is repeated until all OR functions in F are successfully mapped to nanowire rows while avoiding defective crosspoints that cannot be programmed to "closed" state.
2. 2D Sequential Shuffle Algorithm: In this algorithm, the order of nanowire columns are rearranged first so that the possibility of successful mapping could be improved. The idea is to arrange nanowire columns in D so that the function column with larger ON-inputs to the physical column with smaller number of defects. In this way, the possibility of successful mapping will be increased significantly. So, in this algorithm, nanowire columns are rearranged and mapped to function input columns. Then, the row-wise mapping algorithm is invoked to map individual OR functions.
3. 2D Repair with Redundant Inputs: If $M > N$, there should be unused nanowire columns can be used as redundancy. The switching block allocated to arrange input terms to the logic block can be rearranged to assign an input term to more than one nanowire columns. In that case, these unused columns can be utilized as redundancy and more than one nanowire columns can be configured to represent the same input term. So, if more than one of the crosspoint(s) that represent the same input term is programmable, then successful mapping is still possible. Thus, this redundancy utilization may further increase the probability of successful mapping.

The overall cost can be further reduced by using bounding technique [10]. Firstly, a normal algorithm is applied without bounding and unmatched rows are collected. Then these rows are split into multiple rows with smaller number of ON-inputs. These rows are then matched to the remaining unused rows in the crossbar. The switch block is reconfigured to recreate original function expression using rows which are reserved for rerouting. Usually this requires additional rows and switch block reconfiguration.

The example shown in Figure 2 illustrates this technique: Consider a function with 32 ON inputs and it has to be matched on a crossbar block with 37 cross points in a row. Let the OR function to be programmed represented as I . Due to the excessive number of ON inputs, there is a possibility that this particular function will not find a matching row in the nano block even though the nano block has many unused rows (i.e., D_1 to D_6). In this case, instead of utilizing a new nano block, the function is broken into 3 smaller functions (i.e., I_1 , I_2 and I_3) as shown in the figure. These functions have better chance of finding a match with the existing free rows in the nano block. In the example shown (I_1, D_6), (I_2, D_2) and (I_3, D_4) are matched and their effective result is obtained using row D_4 and output routing which is represented with dotted lines.

There are some important points to be considered: 1) The bounding technique is more effective when we have higher number of additional rows, 2) The probability of matching increases with the number of smaller functions generated from the original function. In the given example, to implement a function with large number of ON inputs we have utilized 3

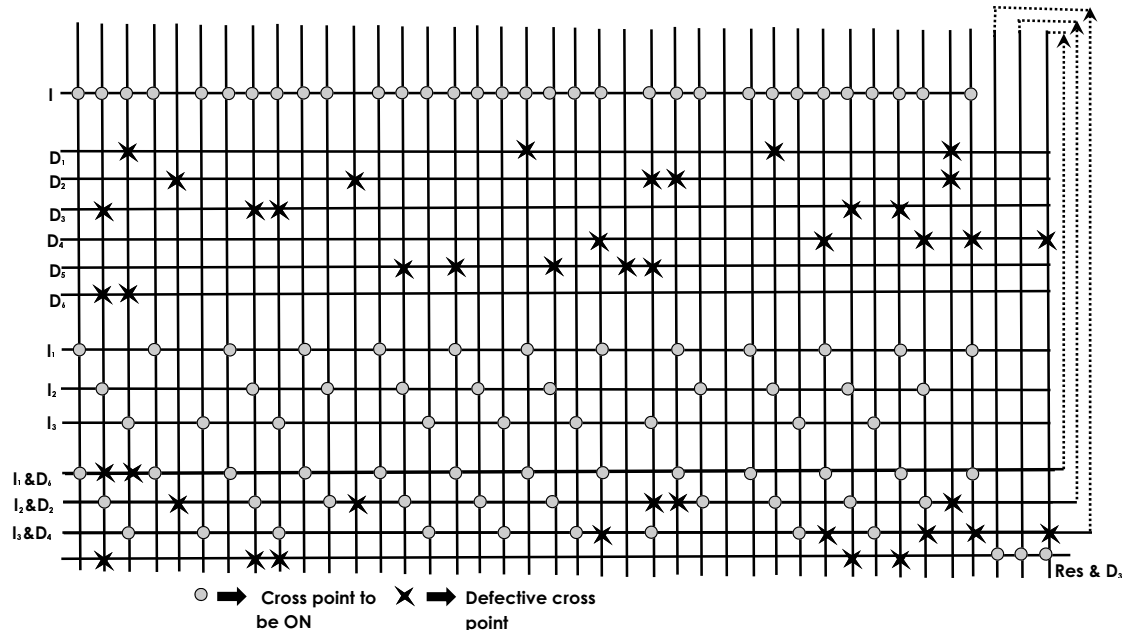


Figure 2. Splitting one OR function with excessive ON-inputs into three parts.

additional rows instead of going for a new nano block. If the number of free rows increases (3 to 4 times original rows) the efficiency of this algorithm will be higher, and 3) The time and switch block reconfiguration overhead increases by a small amount. However, this small increase can be tolerated if the number of logic blocks required to implement a function set reduces.

6 Parametric Simulation and Results

Parametric simulators of the described algorithms were implemented using Matlab. The following assumptions were used throughout the simulation:

- A defect rate of 20% is used with global cluster factor of 0.00001, local cluster factor of 0.02, susceptibility of 0.005 and local cluster size of 3. When a $N \times (N + 20)$ matrix D is constructed, clustered defects are generated using these parameters.
- The function set was generated to simulate a high number of ON-inputs in the set. The size of the function block was chosen as 50×50 . Each row in the function matrix represents an OR function whose inputs are given by columns. The number of ON inputs/per row was controlled as follows: 1) 10% of rows had number of inputs between 1-10, 2) 10% of rows had number of inputs between 11-20, 3) 18% of rows had number of inputs between 21-35, and 4) 62% of rows had number of inputs between 36-50. The ON cross points for each row were generated using uniform random distribution.

Different crossbar sizes of $N \times (N + 20)$ where $N = 50$ to 150 in steps of 5 were considered in the parametric simulation of the proposed repair algorithms. In Figures 3 and 4, simulation results (i.e., the average number of crossbar arrays required to fully implement

the given function set as a function of N) are shown for the proposed algorithms without and with function splitting technique, respectively.

The overall repair performance of the algorithm 3 is superior to the others in both figures. So, it can be concluded that the redundant column utilization technique is very effective. Also, in most of cases, algorithms with the function splitting technique outperform ones without it. There is almost 1 crossbar array reduction, if the data shown in these two figures are compared.

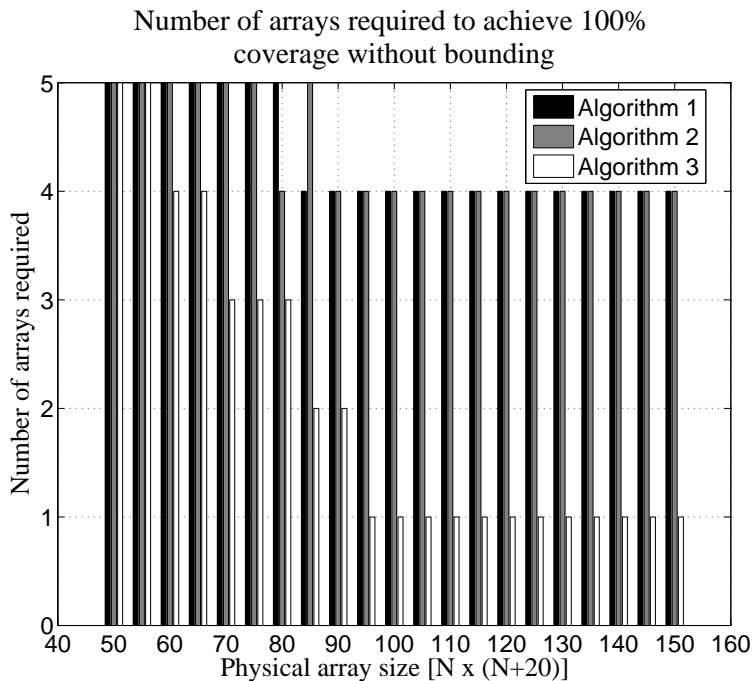


Figure 3. Repair performance of the proposed algorithms without function splitting technique.

7 Conclusion

In this paper, the state-of-the-art in nanowire crossbar architectures and bottom-up assembly paradigm are briefly introduced. Then, defect-tolerance issues are also discussed. For the emerging nanoscale crossbar-based systems, higher defect densities are anticipated due to nondeterministic nature of nanoscale bottom-up assembly paradigm. This indicates that effective and efficient methods are needed to tolerate such defects. Considering the defects in nanoscale wires can be screened out by testing, this paper focuses on avoiding the defective crosspoints in effective manner. Three different repair algorithms have been evaluated to tolerate clustered defects in nanowire crossbars. Input column shuffling and redundant column utilization techniques have been considered and compared with the 1D greedy repair algorithm. Also, the effectiveness of the function splitting technique has been demonstrated through extensive parametric simulations.

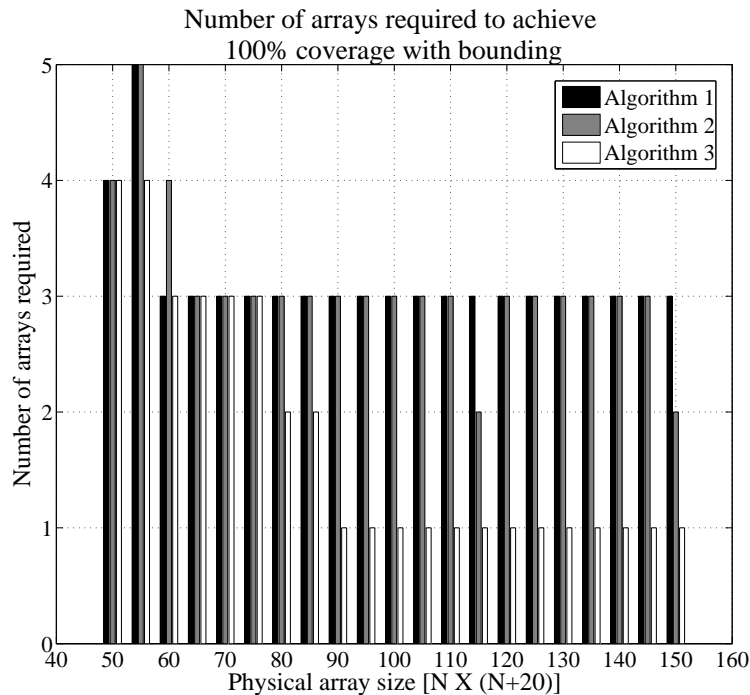


Figure 4. Repair performance of the proposed algorithms with function splitting technique.

References

- [1] S. C. Goldstein and M. Budiu, "Nanofabrics: spatial computing using molecular nanoelectronics", in Proc. 28th Int. Symp. Computer Architecture, 2001, pp. 178-189, 2001.
- [2] N. Melosh, A. Boukai, F. Diana, B. Gerardot, A. Badolato, P. Petroff and J. Heath, "Ultrahigh-density nanowire lattices and circuits", Science, Vol. 300, pp. 112 - 115, April 2003.
- [3] Y. Cui and C. M. Lieber, "Functional nanoscale electronic devices assembled using silicon nanowire building blocks", Science, vol. 291, pp. 851 - 853, February 2001.
- [4] J. R. Heath, P. J. Kuekes, G. S. Snider, and R. S. Williams, "A defect-tolerant computer architecture: Opportunities for nanotechnology", Science, Vol. 280, pp. 1716 - 1721, 1998.
- [5] D. Whang, S. Jin and C. M. Lieber, "Large-Scale Hierarchical Organization of Nanowires for Functional Nanosystems", Japanese Journal of Applied Physics, Vol. 43, No. 7B, 2004.
- [6] Andre Dehon, Michael J. Wilson "Nanowire-Based Sublithographic Programmable Logic Arrays," *FPGA'04*, , Monterey, CA, February, 2004.
- [7] Y. Chen, D. Ohlberg, X. Li et al, "Nanoscale molecular-switch devices fabricated by imprint lithography", Applied Physics Letters, Vol.82, No. 10, pp. 1610-1612, Mar 2003.
- [8] M. Mishra and S. Goldstein, "Scalable defect tolerance for molecular electronics", Workshop Non-Silicon Computation (NSC-1), pp. 78, 2002.
- [9] M. Tehranipoor, "Defect Tolerance for Molecular Electronics-Based NanoFabrics Using Built-In Self-Test Procedure", IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2005.
- [10] H. Naeimi and A. DeHon, "A greedy algorithm for tolerating defective crosspoints in nanoPLA design", IEEE International Conference on Field-Programmable Technology, pp. 49 - 56, 2004.
- [11] C.H. Stapper, "Simulation of spatial fault distributions for integrated circuit yield estimations," IEEE Transaction on Computer-Aided Design, Vol. 8, No. 12, pp. 1314-1318, 1989.