

# Defect-Tolerant Gate Macro Mapping & Placement in Clock-Free Nanowire Crossbar Architecture

Ravi Bonam<sup>1</sup>, Yong-Bin Kim<sup>2</sup> and Minsu Choi<sup>1</sup>

<sup>1</sup>Dept of ECE, University of Missouri-Rolla, Rolla, MO 65409-0040, USA  
{rkbcdf, choim}@umr.edu

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

## Abstract

*Recently, we proposed a new clock-free nanowire crossbar architecture based on a delay-insensitive paradigm called Null Convention Logic (NCL). The proposed architecture has simple periodic structure that is suitable for non-deterministic nanoscale assembly and does not require a clock distribution network - so it is intrinsically free from timing-related failure modes. Even though the proposed architecture offers improved manufacturability, it is still not free from defects. This paper elaborates on the different programming techniques to map a given threshold gate macro on a random PGMB (Programmable Gate Macro Block) with predefined dimension. Defect-Aware and Defect Unaware approaches have been considered to map a given threshold gate onto a PGMB without affecting its functionality. Defect aware approach uses a defect map, gate table which help in efficient programming and also conservative use of resources. Defect unaware approach on the other hand is faster than defect aware approach, does not use defect maps and is not as efficient as defect aware approach. Parametric simulation results using MATLAB are used to show the programmability of these approaches under various circumstances.*

## 1 Introduction

Many of the nanoscale computing architectures proposed in recent years are clock driven. These architectures are mainly based on the two-dimensional nanowire crossbar architecture. In this architecture two sets of parallel, doped silicon nanowires or carbon nanotubes, are crossed over each other orthogonally to form a grid-like structure [1, 2]. The crossing over of these nanowires forms programmable junctions called crosspoints [1, 2, 3, 4]. The primary challenge in designing clocked architectures is to route the clock to all the components of the circuit. Due to imperfections in nanowires fabricated using current manufacturing processes, high defect densities are anticipated and realizing complex synchronous circuits on them is intricate. Hence, nanowire crossbars offers both an opportunity and a challenge.

The opportunity is to achieve ultra-high density which has never been achieved by photolithography. The challenge is to make them *simple enough to be manufactured and reliable enough to be used in everyday computing applications*, since high-density systems consisting of nanometer-scale elements assembled in a bottom-up manner are likely to have many

imperfections (much higher raw fabrication defect densities, as high as 10%, are expected [5, 6]) and parametric variations.

Asynchronous crossbar architecture efficiently helps utilize the opportunity and keep up to the challenge of fabricating reliable complex circuitry. The following conditions are required to be met to make nanowire crossbar to be a viable nanotechnology:

1. Structurally simple and scalable enough to be fabricated by bottom-up manufacturing technique,
2. Robust enough to tolerate extreme parametric variations,
3. Defect and fault-tolerant enough to overcome the extreme defect densities, aging factors and transient faults, and
4. Able to support at-speed verification and reconfiguration.

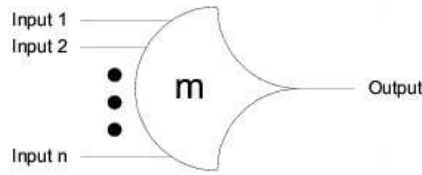
The proposed asynchronous nano-architecture is based on a delay-insensitive data encoding and self-timed logic - therefore, it is totally clock-free. Thus, no clock distribution network is needed and all failure modes related to the timing will be also eliminated. Potential benefits from the proposed asynchronous architecture includes enhanced manufacturability, scalability, robustness and defect and fault-tolerance.

## 2 Preliminaries and Review

The asynchronous crossbar architecture manages to reduce the complexity of circuits by eliminating the clock from the architecture. This architecture uses NCL (Null Conventional Logic) [7, 8, 9], a delay insensitive logic paradigm, which helps in eliminating the clock from the circuit. The use of NCL helps in realizing complex circuits on the nanowire crossbar architecture considering the high defective yield of the current manufacturing processes. NCL uses different logic gates called threshold gates to realize a particular function or logic equation which is quite analogous to its clocked counterparts. The operation of a circuit in NCL is synchronized using local handshakes between registers which act as data holding buffers between each computing circuit. Each of the NCL circuits has an input and output register to achieve the local handshaking phenomenon. Pipelining of NCL circuits i.e. having alternate recurrence of combinational logic blocks and register blocks would increase the throughput of the circuit.

NCL uses threshold gates with hysteresis for its composable logic elements. One type of threshold gate is the  $THmn$  gate, where  $1 \leq m < n$ . A  $THmn$  gate corresponds to an operator with at least  $m$  signals asserted as its set condition and all signals de-asserted as its reset condition.  $THmn$  gates have  $n$  inputs. At least  $m$  of the  $n$  inputs must be asserted before the output will become asserted. Because threshold gates are designed with hysteresis, all asserted inputs must be de-asserted before the output will be de-asserted. Hysteresis is used to provide a means for monotonic transitions and a complete transition of multi-rail inputs back to a *NULL* state before asserting the output associated with the next wavefront of input data. In a  $THmn$  gate as shown in Figure 1, each of the  $n$  inputs is connected to the rounded portion of the gate. The output emanates from the pointed end of the gate. The gate's threshold value,  $m$ , is written inside of the gate.

Threshold gates can be realized on a PGMB [10, 11], which are their logical counterparts in the asynchronous crossbar architecture. A PGMB is nanowire crossbar matrix whose dimensions are predefined prior to the manufacturing and programming. Its dimensions

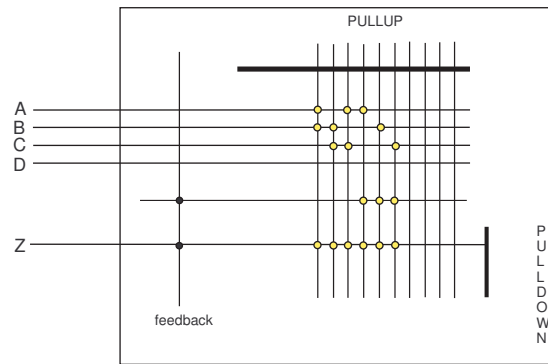


**Figure 1. THmn gate symbol.**

can be adjusted according to the manufacturers defect rate and efficiency of programming algorithm. A PGMB consists of AND and OR planes which are formed due to the pull-up and pull-down resistors.

We have demonstrated the feasibility of realizing a new architecture based on the existing technologies [10, 11]. It has also been demonstrated that each of the threshold gates can be programmed onto a defect free PGMB of 6 rows and 10 columns ( $6 \times 10$ ) [10, 11]. Unfortunately, current fabrication methods are so naive that we have not been able to manufacture a defect-free nanowire crossbar matrix. Hence, to successfully program crosspoints on to a given PGMB an efficient defect-tolerant mapping and placement strategy has to be adopted.

Every threshold gate that can be programmed on to a PGMB has a certain predefined pattern of crosspoint placement that would give the corresponding functionality of the gate. Misplacement of a single crosspoint would lead to malfunctioning of the gate. The Boolean functionality of the gate is represented in the sum of products notation. For instance, a TH23 gate can be expressed as  $F = AB + BC + AC + AF' + BF' + CF'$ , where  $A, B, C$  are the primary inputs and  $F'$  is the output feedback. The first three product terms in this Boolean equation are for the threshold behavior of the gate since the quorum of this gate is 2. Also, the last three product terms (which is also equivalent to  $(A + B + C)F'$ ) are for the hysteresis behavior. Once the output  $F$  is asserted, the only way to make it back to zero is reset all primary inputs. Figure 2 shows a TH23 gate configured on a PGMB.



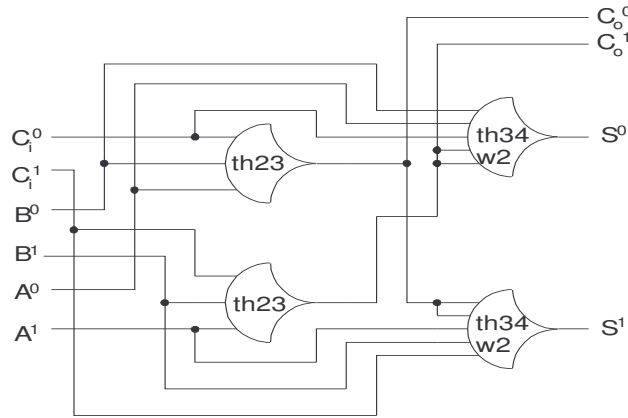
TH23 realized on PGMB

**Figure 2. TH23 gate realized on PGMB.**

To demonstrate the feasibility of the proposed clock-free nanowire crossbar architecture,

an asynchronous 1-bit full adder has been designed and reported in [10, 11]. The NCL 1-bit full adder uses two threshold gates as building blocks; TH23 and TH34W2. TH34W2 is expressed by the function  $F = AB + AC + AD + BCD + AF' + BF' + CF' + DF'$ , where  $A, B, C, D$  are the primary inputs and  $F'$  is the output feedback. Using the Dual rail encoding scheme a full adder can be realized using the following functions,  $Co^0 = A^0B^0 + Ci^0A^0 + Ci^0B^0$ ,  $Co^1 = A^1B^1 + Ci^1A^1 + Ci^1B^1$ ,  $S^0 = Co^1A^0 + Co^1B^0 + Co^1Ci^0 + A^0B^0Ci^0$ ,  $S^1 = Co^0A^1 + Co^0B^1 + Co^0Ci^1 + A^1B^1Ci^1$ , [12] where  $A, B$  represent primary inputs,  $Ci$  represents carry in,  $S$  and  $Co$  represent sum and carry out respectively. Each of these variables are represented by dual rail encoding scheme and rails are indicated by  $^0$  and  $^1$  preceded by the variable.  $Co, S$  can be represented by TH23 and TH34W2 gates respectively.

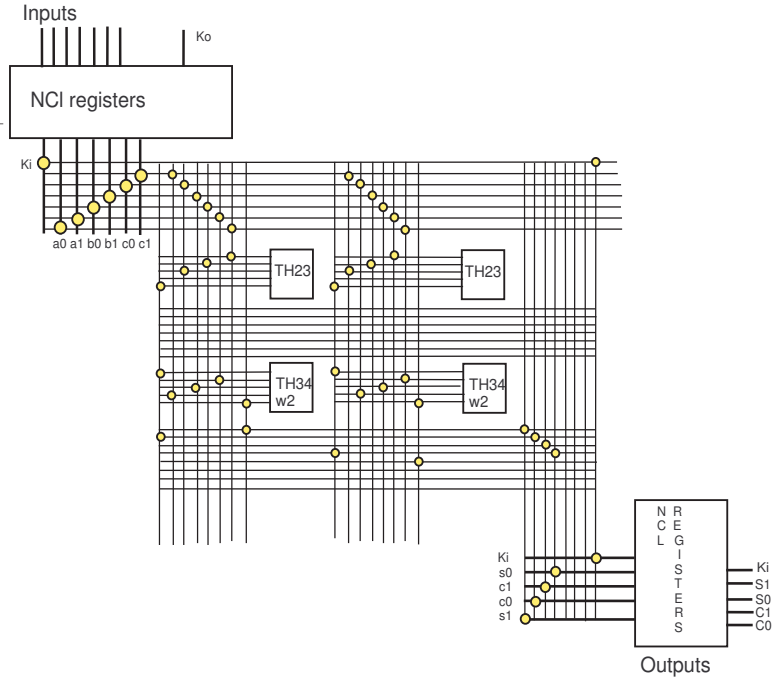
Two instances of TH23 and TH34W2 gates are used in the schematic shown in Figure 3. The same circuit can be realized with the proposed architecture and it is shown in Figure 4.



**Figure 3. NCL 1-bit full adder using threshold gates.**

Notably, we can rearrange the product terms in the gate macro equation without affecting the functionality of the circuit. The rearranging of the product terms gives us a range of possibilities to place crosspoints. This helps in utilizing the inherent redundancy of a defective PGMB. According to researchers, current fabrication processes have defect rates of about 10% [13][14] in the nanowire crossbar, but scientists are yet to discover a standard fabrication technique which would have a consistent defect rate. The proposed algorithms aim at two different aspects, speed and utilizing the inherent redundancy of mapping and placement.

The subsequent sections elaborate on the proposed defect tolerance techniques for programming PGMB's, routing between them and results obtained by the use of different algorithms.



**Figure 4. 1-bit adder implemented on the proposed asynchronous architecture.**

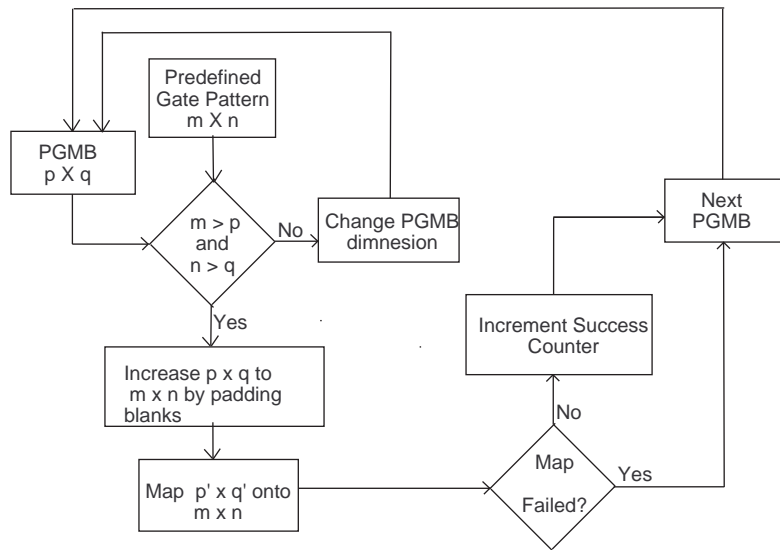
### 3 Proposed Mapping and Placement Techniques

#### 3.1 Defect-Unaware Approach

In this technique a predefined pattern of a gate is mapped on to a PGMB directly without the knowledge of any defects. We need not spend much time on generating the defect map of a given PGMB. The idea is to speed up the process of placement by compromising a few defective placements. The use of this technique is dependent on the consumer and manufacturer's capability to fabricate defect free PGMB's. The control flow for placement of crosspoints using the defect-unaware strategy is as illustrated in Figure 5.

The values of  $m$  and  $n$  for the predefined gate pattern are accessed from a gate table database which consists of all the required information (number of crosspoints, minimum rows and columns required to program the gate etc.) concerning a threshold gate. This table can be accessed by the algorithm while mapping and placement. The purpose of changing the dimensions of the predefined gate is to ensure precise placement of crosspoints.

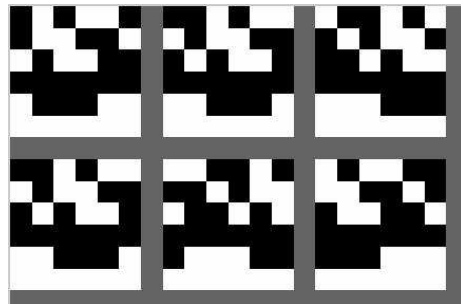
This simulation results for the given algorithm are shown in Figure 8 (the lower curve). On analyzing the simulation results we can infer that as defect rate increases the programmability of the gate decreases. We can clearly observe that this method is suitable only for defect rates ranging from 0-10%. Hence, if a manufacturer can guarantee that the defect rate of his fabrication process is under 10%, this process can be utilized.



**Figure 5. Defect-Unaware Control Flow**

### 3.2 Defect-Aware Technique

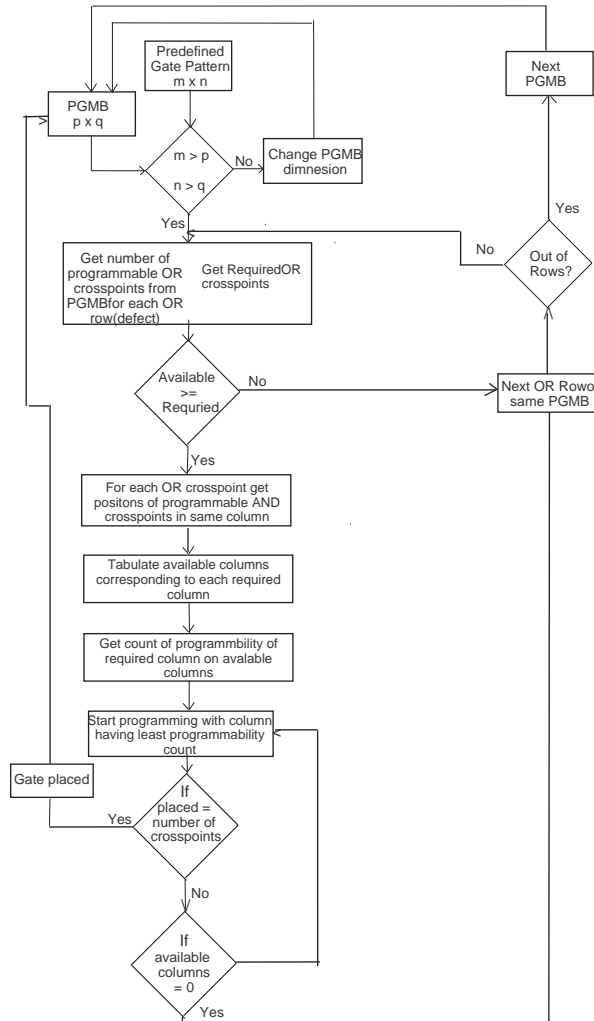
As most of the fabrication techniques cannot guarantee on their defect rates, the defect unaware process of placement is not as efficient in utilizing the inherent redundancy of a given PGMB. The defect aware approach on the other hand makes use of the available redundant crosspoints to program a given gate onto a PGMB. The main challenge associated with inherent redundancy is to use it without affecting the functionality of the gate. On observing the pattern of crosspoints of a gate we can infer that columns represent the intermediate product outputs and they can be interchanged in any fashion without affecting the functionality. A TH23 gate is represented in 6 ways with different patterns of crosspoint placement without affecting the functionality of the gate in Figure 6.



**Figure 6. Different mapping configurations of a TH23 gate macro.**

The defect aware technique utilizes the fact that any given gate can be represented in different ways without affecting the gate's functionality. This coupled with the inherent redundancy would give us a good scope of being able to map and place crosspoints on highly defective PGMB's.

The Defect aware approach generates a defect map of a given PGMB and compares the pattern of defect free crosspoints with the required crosspoint pattern. The mapping and placement flow of the technique is illustrated in the Figure 7.



**Figure 7. Placement flow for defect-aware approach.**

The algorithm for the defect aware approach is as follows:

1. Get the dimensions of PGMB (i.e.,  $p \times q$ ) and minimum required rows  $\times$  columns for programming the required gate (i.e.,  $m \times n$ ).
2. If  $p > m$  and  $q > n$  proceed to step 3, else roll back to step 1 and get next PGMB.
3. Get the count of defect-free crosspoints corresponding to each row on the PGMB, and also the required count of OR crosspoints for programming the Gate.
4. Consider a row from the PGMB, If available crosspoints in the row are greater than the required crosspoints then proceed to step 5. Else rollback get next row. If there are no more rows with required crosspoints, then go to step 1 and get next PGMB.
5. Tabulate available crosspoints of each column corresponding to the selected row.

6. Get the programmability count of each of the required columns on each of the available columns for the selected row.
7. Starting with least programmability count, place the crosspoints.
8. If placed crosspoints is equal to the number of crosspoints to be placed then Go to step 1 and start with next PGMB Else if columns are available go to step 7 and the next column. Else Go to step 4 and select next row.

The significance of using this defect aware strategy is that it starts programming with the column(of the gate to be programmed) having least programming capability on a given PGMB which will help in maximum utilization of the inherent redundancy.

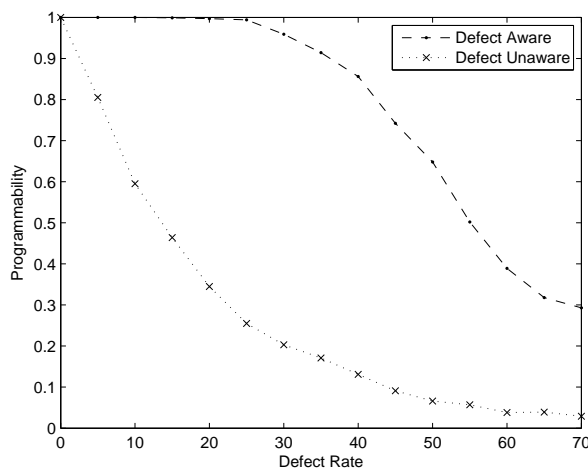
The result of using the defect-aware approach for programming is illustrated in Figure 8 (the higher curve).

On analyzing the graph we can observe that almost all the gates are programmed even at 30% defect rate and then programmability (ratio of successfully programmed PGMB's to the total number of PGMB's) reduced which is inevitable. This is much better in terms of programmability compared to the defect aware technique but it takes time to generate defect map and then perform mapping and placement, but is very good at utilizing the inherent redundancy. This helps when we have a highly defective fabrication process, the algorithm bypasses the defective crosspoints and places the crosspoints without affecting the functionality of the gate.

For both the techniques 20% of the rows have been dedicated to the OR crossbar logic.

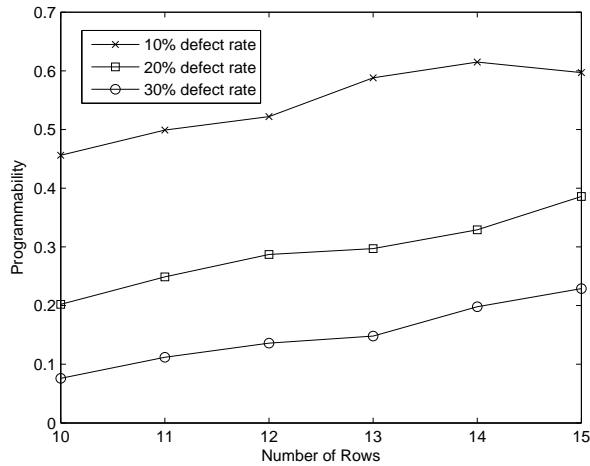
## 4 Parametric Simulation Results

Figure 8 shows an illustration of the behavior of programmability with varying defect rates for both the techniques. The defect-aware approach surely outperforms the defect-unaware approach, but the defect-unaware approach still shows good programmability when the defect rate is considerably lower (e.g.,  $\leq 5\%$ ).

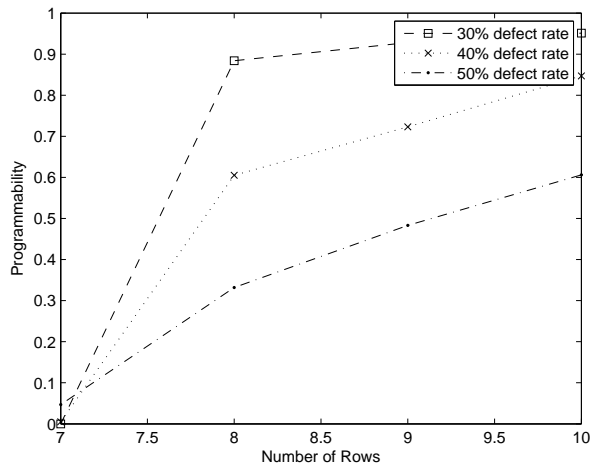


**Figure 8. Programmability comparison for defect-aware and -unaware approaches.**

On analyzing the figure we can clearly observe that the defect aware technique has a better programmability than the defect unaware technique. Analysis is also done for the two techniques with different dimensions of PGMB's.



**Figure 9. Programmability vs # of rows for defect-unaware approach.**



**Figure 10. Programmability vs # of rows for defect-aware approach.**

Figures 10 and 9 provide useful analysis for the behavior of programmability with change in dimension of the PGMB at different defect rates. These graphs also indicate the superiority of defect aware approach over the unaware approach.

## 5 Future Work

Even though the defect-aware approach is better than the defect-unaware approach especially when the defect rate is higher, it requires much more laborious testing (i.e., each PGMB and switch block should be tested to locate all defective crosspoints) and reconfiguration (i.e., all defective crosspoints should be avoided when the netlist is actually placed and routed) tasks. However, the defect-unaware approach can be simpler since the netlist is directly mapped without considering any defects. After that, PGMBs and switch blocks can be functionally tested to locate ones with faults. These faulty ones then can be tested and reconfigured to avoid defects.

## References

- [1] Matthew M. Ziegler and Mircea R. Stan, "Design and analysis of Crossbar Circuits for Molecular Nanoelectronics," IEEE Nanotechnology Conference, pp. 323-327, 2002.
- [2] 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.
- [3] Y. Cui and C. M. Lieber, "Functional nanoscale electronic devices assembled using silicon nanowire building blocks," Science, Vol. 291, pp. 851-853, 2001.
- [4] Nicolas A. Melosh, Akram Boukai, Frederic Diana, Brian Geradot, Antonio Badolato, Pierre M. Petroff, James R. Health, "Ultrahigh-Density Nanowire Lattices and Circuits," Science, Vol. 300, pp. 112-115, 2003.
- [5] J. Huang, M. B. Tahoori and F. Lombardi, "On the defect tolerance of nano-scale two-dimensional crossbars", IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 96-104, Oct 2004.
- [6] M. Jacome, C. He, G. de Veciana, and S. Bijansky, "Defect tolerant probabilistic design paradigm for nanotechnologies", IEEE/ACM Design Automation Conference (DAC), pp. 1-6, 2004.
- [7] Karl M. Fant, Scott A. Brandt, "NULL Convention Logic : A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," IEEE International Conference on Application-Specific Systems, Architectures and Processors, pp. 261-273, 1996.
- [8] S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb, "Optimization of NULL Convention Self-Timed Circuits," Integration, The VLSI Journal, Vol. 37, No. 3, pp. 135-165, 2004.
- [9] S. Smith, R. DeMara, J. Yuan, M. Hagedorn and D. Ferguson, "Delay-Insensitive gate-level pipelining," Integration, the VLSI journal, Vol. 30, pp. 103-131, 2000.
- [10] R. Bonam, S. Chandhary, Y. Yellambalase and M. Choi, "Clock-Free Nanowire Crossbar Architecture based on Null Convention Logic (NCL)", Accepted to appear in the 7th IEEE International Conference on Nanotechnology (IEEE-Nano), Apr 2007.
- [11] R. Bonam, Y. Yellambalase and M. Choi, "Redundancy Optimization for Clock-Free Nanowire Crossbar Architecture", Accepted to appear in the 7th IEEE International Conference on Nanotechnology (IEEE-Nano), Apr 2007.
- [12] S. C. Smith, "Gate and Throughput Optimizations for NULL Convention Self-Timed Digital Circuits", Ph.D. Dissertation, School of Electrical Engineering and Computer Science, University of Central Florida, May 2001.
- [13] M. Mishra and S. Goldstein, Scalable defect tolerance for molecular electronics, Workshop Non-Silicon Computation (NSC-1), pp. 78, 2002.
- [14] M. Tehranipoor "Defect tolerance for molecular electronics-based nanofabrics using built-in self-test procedure", 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2005, Oct. 2005 pp 305 - 313