COMPUTING WITH NANOSCALE DEVICES – LOOKING AT ALTERNATE MODELS

Karthikeyan VijayaRamachandran
B.E., Sri Chandrasekhrendra Saraswati Viswa Maha Vidyalaya (2002)

A thesis presented to the faculty of the
OGI School of Science & Engineering
at Oregon Health & Science University
in partial fulfillment of the
requirements for the degree
Master of Science
In
Electrical and Computer Engineering
May 2005

The dissertation "Computing with Nanoscale Devices – Looking at Alternate Models" by Karthikeyan VijayaRamachandran has been examined and approved by the following Examination Committee:

Dr. Daniel Hammerstrom
Professor
Thesis Research Advisor

Dr. John Carruthers
Adjunct Professor
Intel Corporation

Dr. Raj Solanki
Professor

ACKNOWLEDGMENT

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I want to thank my thesis advisor Dr. Daniel Hammerstrom whose help, stimulating suggestions and encouragement helped me in my research and in writing this thesis. I am deeply indebted to Dr. John Carruthers for providing me with suggestions and ideas throughout my research. I thank Dr. Raj Solanki for his help and support.

Especially, I would like to give my special thanks to my parents whose patient love enabled me to complete this work.

ABSTRACT

With modern CMOS technology likely to reach physical limits in the next decade or so, researchers have been working on alternate approaches to continue semiconductor scaling. This next level would be at the nanoscale, that is, electronic devices with dimensions of a few nanometers, 10^{-9} of a meter. This thesis represents an early step in understanding one proposed implementation strategy for projected nanoscale devices. Although various architectures have been proposed, the most promising and interesting is that of a crossbar array with programmable cross points. A number of nanoscale crossbar devices ("nanoarrays") have been demonstrated. The use of such architecture is being researched at many levels and has yielded interesting results.

Our work aims at studying the electrical properties, efficiency and reliability of the crossbar arrays for a particular type of memory structure. In the first phase of the research we began by simulating the physical and electrical properties of the silicon nanowire and testing a simple 2x2 crossbar array made of silicon nanowires. Based on fabrication constraints obtained from real arrays, we have estimated a maximum array size that can be achieved. In the second phase of the research we calculated the "RC" delays for the crossbar array network. In this phase we studied two different crossbar models – the "Molecular" model and the "Leiber" model, for estimating the delays. In the final phase of this work we have done preliminary defects analysis of both the models and estimated the tolerance level of the models for various defect densities.

Contents

List of Figures

Chapter I

INTRODUCTION

With today's chip making technologies likely to reach physical limits in the next 10-15 years, researchers are studying alternative approaches device technologies that would allow the continued scaling of semiconductor circuits. Although current CMOS has nano-scale aspects to it, most of its dimensions are closer to the microscale. The next major level of scaling will be the use of nano-scale circuits with dimensions of only a few nanometers. However, nano-scale devices as currently envisioned have significantly more limited functionality than state of the art CMOS circuits.

Transistors are the electrical switches that form the core of CMOS circuit technology. Being able to create nanoscale switches that are connected by nanowires measuring a few nanometers in diameter would enable computer chips with over a trillion transistors per square centimeter, which is several orders of magnitude more than current CMOS technologies are likely to achieve.

Consequently there are various problems that researchers have to overcome in order to create fully functional nano-electronic circuits. These include:

- Developing a nanoscale device that is capable of acting as a non-linear switching action;
- Developing a technique to link the devices at both levels; and
- Organizing the nano-switches in an architecture that is efficient, defect tolerant and relative easy to build and program.

A promising architecture is a simple grid of rectangular nano-wires whose junctions form tiny densely packed switching devices. The molecular-scale wires can be arranged into interconnected, crossed arrays with non-volatile switching devices at their crosspoints. These crossed arrays can function as programmable-logic arrays, memories or programmable interconnects.

1.1 MOTIVATION

A standard MOSFET has three terminals: a source, drain and gate. Current flows from the source to the drain and is controlled by the gate. When a voltage is applied to the gate, its electric field creates a conductive channel between the source and drain that allows current to flow, creating an electrically controlled switch. An input signal opens the gate, switching the transistor on to create an output signal. Unfortunately, there are as yet no real equivalent three-terminal devices at the nano-scale. However, a number of "switches" have been proposed. Some switches use current shunting via variable resistance molecular connections, others a crude FET made from crossed nanowires, where one nanowire is the channel and the perpendicular nanowire becomes the gate. In a nano-grid, each input nanowire crosses every channel, or output, nanowire.

The disadvantage of the nanowire crossbar as described above is that one input nanowire affects all the output nanowires in the same way preventing selective addressing of elements. Changing the resistance of the specific crossbar junctions is one way to solve this problem. One approach is to change this resistance by chemically modifying the junction. Groups at UCLA and HP have demonstrated molecules that appear to exhibit several orders of magnitude changes in resistance in different states. Another group at Harvard has demonstrated a nano-grid where they can chemically modify specific cross point junctions in a nanowire grid.

In 2003, **Hewlett-Packard Laboratories** developed a method of addressing individual junctions in a nanowire array memory device. They have described the fabrication and testing of a molecular-electronic circuit that consists of a molecular monolayer of rotaxanes sandwiched between metal nanowires. Each crosspoint was used as a memory cell to create a dense crossbar memory. By varying the resistance of each crosspoint they configured the crossbar array to function as a multiplexer and demultiplexer which was used to read the memory elements formed by a separate crossbar array**[1]**.

The other grid architecture proposed by Charles Leiber and his group at Harvard is formed by crossed nanowire Field Effect Transistor (cNW-FET) junctions, where selective chemical modification of cross points in the arrays enables NW inputs to turn specific FET array elements ON and OFF. The chemically modified cNW-FET arrays function as decoder circuits, exhibit gain, and allow multiplexing and demultiplexing of information as described in "**Nanowire Crossbar Arrays as Address Decoders for Integrated Nanosystems**"[2].

The **Leiber** Architecture has one major advantage over the **Molecular** Architecture. The junctions in the HP device are made of resistors rather than transistors. Transistors provide signal gain, meaning the output voltage is higher than the input voltage. Because electrical signals fade, signal gain is necessary to allow signals to propagate through circuits.

These two grid architectures form the basis of the work performed here, where we evaluate both architectures as potential building block circuits in larger nano-grid based systems, measuring delay and defect tolerance for the arrays aiming towards creating memory and logic arrays.

1.2 BACKGROUND

In order to characterize these grid architectures we can divide the analysis into a two level hierarchy composed of: (1) devices, (2) architectures. Each level of the hierarchy consists of several devices. Although it is likely to undergo many revisions, this hierarchy has provided a good starting point for the nanoarrays studied here.

1.2.1 DEVICES & CONNECTORS

The devices themselves constitute the lowest level of the hierarchy. A device is an entity that can be defined as one which performs a useful action, which for the arrays being analyzed here is generally a switch. A number of research groups are searching for useful devices that can approximate the function of traditional 3-terminal Field Effect Transistors (FETs) but at the nanoscale, though at the moment there is no real CMOS transistor replacement. Likewise, nano-wires of some kind will be needed to carry the signal between the various nano-switches. Nanoscale wires, such as carbon nanotubes (CNTs) and silicon nanowires (SiNW) are being studied.

Carbon nanotubes. Carbon nanotubes consist of carbon atoms, rolled up to form molecular tubes or cylinders with diameters between 1 to 20nm and length ranging from 100 nm to several microns. CNT-FET device structures have been fabricated and tested with a standard source, drain and gate, with a CNT forming the channel.

Even though CNTs have much promise, there are many problems that need to be solved. The biggest concerns the fabrication of CNTs. CNT fabrication is difficult to control, and current technology produces a mixture of all types and sizes of CNTs. In order to fabricate a specific device, the CNTs have to be sorted based on size and type, which is not feasible for large numbers of CNTs.

Silicon nanowires. A typical SiNW typically has a diameter of 10-15 nm, and a length of a few micrometers. Silicon nanowires are a promising candidate for use in nano-scale circuits. One reason is that they have greater mobility compared to bulk silicon. This characteristic is related to the quantum-confined nature of the wire, which limits the number of available phonon states, thus reducing the number of electron phonon scattering events.

FET structures have been fabricated using SiNW. Several geometries have been developed including the back gate, top gate and coaxial gate, with each geometry having its own unique I-V characteristics.

Silicon Nano-wires are a promising building block for the **Cross-bar Arrays** studied in this research, and assumed for both the Molecular and Leiber array structures. A cross bar array (or nano-grid) can be formed by placing an array of SiNWs at right angles to another array of SiNWs, creating n^2 cross-points formed in the array. The cross-points of the arrays can be used to store information or to control the current flow in the array. There are several advantages to a crossbar, including ease of programming. However, the major advantage of the nano-scale cross-bar arrays is its density. And, fabrication costs may not be exorbitant, since these arrays can be self assembled.

However, there are also disadvantages to nano-grids. The two most serious disadvantages are unreliable operation (due to faults and defects) and speed of operation, which will probably be much slower than that of a scaled CMOS. It is most likely that these devices will be built on top of traditional CMOS, which can provide communication with the outside world and signal restoration. A number of researchers have proposed ways to connect CMOS to nano-grid structures, one example method is proposed by *Matthew M.Ziegler and Mircea R.Stan*[4]. Recent work at OHSU's OGI School of Science and Engineering has demonstrated a technique for growing silicon nanowires at precise locations under a controlled environment.

Molecular Switches. Molecular memory is a term that refers to memories that are built from a single molecule, atom or cell that stores one bit of information. The information storage occurs with a change in the molecular configuration of the molecules or atoms when voltage is applied, generally this can be quantified to two distinct conduction states, representing logic '1' and logic '0'. Information is read out of the molecular memory by measuring the resistance changes in the molecular connection. Information storage occurs when there is a hysteresis effect between voltage and resistance for larger voltage fluctuations. One such molecule that exhibits such a property is catenane, which opens at 2 V, closes at 1 V, and can be read at 0.1 V. This effect is sufficiently repeatable that the switch can be cycled open and closed many times. When used between a metal wire and n-type silicon nanowire, the junction acts as a programmable diode, making an addressable memory array. So far, conductance values varying 4x between the two states have been shown.

Molecular switches can be used to build molecular-based logic gates. A thin (one molecule) layer of rotaxane molecule, whose resistance can be significantly changed based on its oxidation state, can be used as a switch. The closed switch configuration exhibits a non-linear I-V characteristic. The switch can be programmed by applying voltage in a certain range to oxidize the layer of rotaxane. But rotaxane switches have their limitations. The switch operation is based on the principle of oxidation and since oxidation in this case is irreversible, the switch programming is an one-time operation. There is static current leakage in the devices fabricated using this process and only inherently molecule-sized in one dimension.

1.2.2 NANOARCHITECTURES

The term nanoarchitecture is used here to connote a structure of multiple devices that computes something useful, generally a logical function, or operates as a multi-element memory, and that is built using molecular scale components. We will use architecture to mean a nanoarchitecture in the context of the work being reported here.

According to this definition, architectures are the next structural level above devices and basic circuits. The ITRS has suggested that a number of new, "emerging" architectures, based on new characteristics may appear over the next two decades. Table 1.2.1 summarizes some of these emerging architectures.

Architecture	Implementations	Advantages	Challenges	Maturity
3D Integration	CMOS with dissimilar material systems	Less interconnect delay; enables mixed technology solutions.	Heat removal; no design tools difficult test and measurement	Demonstration
Quantum cellular automata	Arrays of quantum dots	High functional density; no interconnects in signal path	Limited fan out; dimensional control(low-temperature operation); sensitive to background charge	Demonstration
Defect-tolerant	Intelligently assembles nanodevices	Supports hardware with defect densities > 50 %	Requires precomputing testing	Demonstration
Molecular	Molecular switches and memories	Supports memory-based computing	Limited functionality	Concept
Cellular nonlinear networks	Single-electron array architectures	Supports memory-based computing	Subject to background noise; tight tolerances	Demonstration
Quantum computing	Spin resonance transistors, NMR devices, single-flux quantum devices	Exponential performance scaling, but can break current cryptography	Extreme application limitation; extreme technology	Concept

Table 1.2.1 Emerging research architectures

1.3 RESEARCH METHODOLOGY

The research plan is divided into three phases. The first phase is to understand the research area by studying the latest developments from other research. The second phase is to study various models in order to select an efficient model for simulation. And the third and final phase is to simulate the selected models and evaluate the models based on delay and defect tolerance.

1.3.1 NANOARRAY ANALYSIS

The first phase for a research project is to completely understand the research area. This work is focused on nanoscale devices and circuits specifically aimed at building a nanoscale memory. Although we use a memory as the basic computational model, the same grid structure can also be used to implement a programmable logic array. The study helped clarify the memory architecture and its functionality. The next concentration was in the area of nanoelectronics. The goal was to understand the problems faced by the current approaches to nano-electronics. At the nanoscale we will essentially be dealing with single molecules. Nanoscale circuits cannot be fabricated in the same way as traditional CMOS circuits and must be self-assembled. Since we cannot predict or control the arrangement of molecules, their function could be fully realized only after they are completely assembled.

There are a number of different methods for creating nanoscale logic devices, however, none of the current devices provide the capabilities of traditional CMOS. For example, most can only realize two-terminal-rather than three terminal-transistor devices.

1.3.2 MODEL DEVELOPMENT

The second phase was to study and analyze various candidate nanoscale circuit structures. A number of nano-electronic structures and their corresponding models have been proposed for memory and grid-like structures. A crossbar memory architecture seems to be the most promising and researchers are evaluating this architecture at many levels. A number of architectures based on crossbar circuits employing two-terminal devices have been recently suggested for memory and logic. Fig 1.3.1 shows an abstract representation of such a crossbar, consisting of two sets of parallel nanowires crossing perpendicularly.

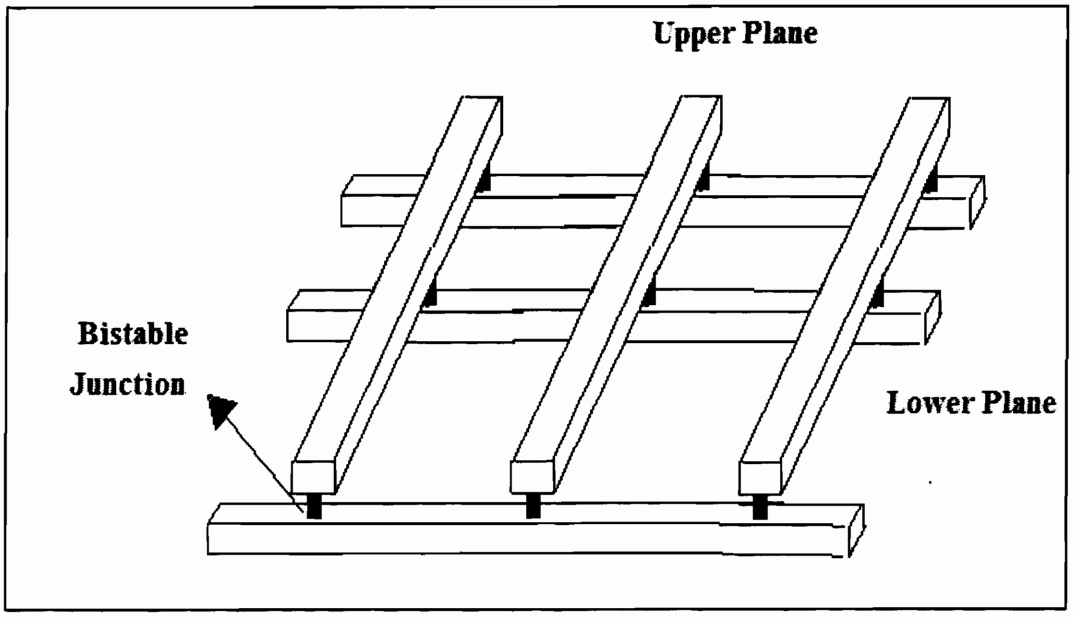


Fig 1.3.1 Crossbar Array

The crossbar structure seems to have several advantages compared to other circuit architectures, the wire dimensions can be scaled down to molecular sizes, while the number of wires can be scaled up arbitrarily to form large-scale generic circuits that can be configured for memory and/or logic applications. It has the potential for low-cost fabrication and high device densities. Also it appears as if most nanogrid designs have the potential to communicate efficiently with external circuits and systems. However, there are also several potential drawbacks to crossbar circuits. Most of the proposed models use two-terminal devices with no signal gain and inverting function. Another drawback is that since the crossbar architecture is regular with wires running long distances next to each other, there is significant interwire capacitance and even electron tunneling at the nano-scale. Manufacturing defects are another concern, but that is universal among all nano-electronic structures.

In this phase of the project, various architectures and models were evaluated based on their use as a memory. The next step was to study various models of the crossbar architecture in order to find a reasonable simulation model.

1.3.3 SIMULATION MODELS

In this third phase the model selected during the second phase was simulated. The model simulation consisted of four steps. The first step was to simulate the electrical characteristics of **Silicon Nanowire (SiNW)**. The crossbar array network is created by arrays of horizontal and vertical silicon nanowires. In order to measure the characteristics of the crossbar array we had to start with the measurement of electrical characteristics first of SiNW.

During this first simulation step, a model of a SiNW was created for measuring the resistance of the nanowire. The relationship between the resistance and the length of the SiNW and the resistance and diameter of the SiNW were studied. The range of numerical values for length and diameter were chosen based on the typical size of a SiNW. This simulation step gave a rough idea of the order of magnitude of SiNW resistances. The simulation was carried out for larger dimensions to demonstrate the range of values possible if current fabrication constraints could be overcome. The conductivity of the SiNW, which is fairly low at a nano-scale diameter has a significant effect on the total speed of the array. If there is a way to find a SiNW with better conductivity then that could lead to significant improvement in the size and speed of nanoarrays.

In the second simulation step a simple crossbar array model was created, using the nano-model developed in the first simulation step. The dimensions of the crossbar array was 1x1 i.e. two SiNW's perpendicular to each other. Through this model the crossbar intersection point, i.e., the crosspoint, was formed. The crosspoint is the region where the vertical and horizontal nanowires cross each other and in which the data can be written and read, allowing the crossbar array to be used as memory. The area occupied by the crossbar array was calculated by simulating the model. The area was then calculated for higher dimensions and for the largest array that could be reasonably fabricated, considering the current fabrication constraints. In calculating the area occupied by the array, certain assumptions were made on the pitch between SiNW's, the area of each junction and the end dimensions of the array. The number of junctions per unit area was calculated based on the area occupied by the crossbar array and the dimension of the array, which has given us some interesting results.

The third simulation step was the “RC" delay calculation. For any grid structure, the “RC” delay is one of the most import factors that determine grid performance. The “RC” delay calculations were done on two models – **“Leibers Model”** and **“Molecular Model”**. The resistance values used for the delay calculation were from previous simulations and the capacitance values used were from data obtained from Prof. Charles Leiber (Harvard University). Two separate delay models were created corresponding to the Leibers and Molecular models. Assumptions were made on the inductance of the SiNW’s and on the effect of coulomb blockage. The first assumption is that the inductance of the silicon nanowire is low enough that its effect can be neglected, second that there is no coulomb blockage at room temperature in the nanowires, and third that the interwire pitch is large enough that there is no electron tunneling from one wire to another. The worst case delays were calculated, i.e., the longest path for the current flow was found for each case and then the delay for that path was estimated. The delay of each model was studied with respect to the length and diameter of SiNW, the size of crossbar array and the area of the crossbar array.

The fourth simulation step was a preliminary defects analysis on the crossbar arrays. Two main defects were studied and the crossbar array area was compared before and after introducing redundancy in the arrays. The redundancy was calculated based on yield levels. Various array sizes were considered for the defects analysis.

During the final simulation step we have estimated memory density per cm^2 of crossbar array area. We used a standard cell containing a crossbar array with multiplexer interconnects that can function as a memory. We used this cell to estimate the memory densities for various memory capacities assuming a certain amount of defects rate and compared with the density with that of a defects free crossbar memory

Chapter II
CROSSBAR ARRAYS

This chapter describes crossbar array circuits in more detail. This chapter also describes the primary device models used for analysis presented here, **"Leibers Model"** and **"Molecular model"**. The crossbar architecture has several advantages. First, the dimensions of the wires can be scaled down to molecular sizes, and the array size can be scaled up in two dimensions to form larger circuits that can be used to implement logic and/or memory applications. Second, the addressing capacity of crossbar arrays is very high. This characteristic of the crossbar circuits allows it to communicate efficiently with external circuits such as CMOS. Third, the reconfigurable architecture makes programmability easier. And finally the physical structure is very simple easing the fabrication of crossbar circuits. In order to realize a completely functional nanoscale memory using a crossbar array architecture, we have to require certain characteristics. The first characteristic being reversibility, the programmable crosspoints should be able to store both logic values and there should be reversibility in storage i.e. if a certain crosspoint stores a logic value '0' we should be able to overwrite that with logic '1' and vice versa many times without losing the data. In some applications a Read-Only-Memory (ROM) is all that is required and so reversibility is not needed. To be useful, the memory should have a detectable state change, i.e. data stored at a crosspoint should not change until it is overwritten. Finally we should be able to integrate the memory structure with CMOS circuitry.

2.1 CROSSBAR CIRCUIT FABRICATION

A crossbar circuit consists of two orthogonal sets of parallel aligned wires. A one dimensional array of silicon nanowires can be fabricated using a wide variety of techniques such as e-beam lithography, imprinting lithography or chemical self-assembly. The nano-imprint lithography technique appears to be the most promising for fabricating crossbar array circuits. Imprint lithography is a nanoscale processing technique that can produce feature sizes less than 10 nm with high throughput and low cost. Another advantage of the nano-imprinting technique over other techniques is that the probability of fabrication defects/faults is lower than when using high-energy electrons under e-beam lithography, since imprinting reduces damage to circuit components. In imprinting lithography a prepatterned mold is brought into soft contact with a thin polymer film located on top of the substrate. By applying pressure and increasing the temperature, the mold pattern is transferred into the polymer. The residual polymer from the feature surface is removed and further processing can occur such as metal deposition, etching etc. Thus nanoimprint lithography is cost-efficient, since no sophisticated tools are required and has reasonably high throughput.

Fabricating a molecular electronic device or crossbar memory device using the imprinting method has been proposed by a research group at HP Labs (US Patent – 6579742 - Chen; Yong (Palo Alto, CA)). Fabrication begins with at least one bottom electrode on a substrate by forming a first layer on the substrate and then patterning the first layer to form the bottom electrode by an imprinting technique. In the second step the molecular switch film is deposited on top of the bottom electrode, optionally forming a protective layer on top of the molecular switch film to avoid damage during further processing. In the third step a polymer layer is deposited on top of the protective layer and patterned by the imprinting method to form openings that expose portions of the protective layer. In the final step, the top electrodes on the protective layer are connected through the openings in the polymer layer by first forming a second layer on the polymer layer and then patterning the second layer. Though it is still a long ways from volume manufacturing, the above described fabrication method is one of the methods that are being considered for fabrication of molecular crossbar array circuits.

Assuming that fabrication is possible, the next question concerns how to model the computation performed by a crossbar array architecture for nanoscale devices and circuits, especially for nanoscale memory. The next section summarizes the various circuit and device models used in this research.

2.2 CROSSBAR ARRAY MODELS

A general crossbar memory is shown in Fig 2.2.1. The memory consists of two components. The first is the actual crossbar array consisting of 16 vertical and 16 horizontal nanowires – forming a 256-bit memory circuit. At each intersection there is a bistable molecular switch capable of storing a single bit of data. The connection between the microscale and nanoscale is established at the binary tree multiplexer. The multiplexers adopt some interesting architectural variations that allow them to bridge from the micron or submicron scale of the larger sized wires to the nanowires. Each multiplexer consists of four sets of complementary wire pairs, designed to address 2^4 nanowires. The scaling is logarithmic: 2^{10} nanowires would require only 10 wire pairs for each multiplexer. One wire within each pair has an inverted input; for example a "0" input sends one wire low and its complement high. Along each of the microscale wires (the thicker wires in the figure) there are rectifying connections to the nanowires; each pair of wires has a complementary arrangement of connections. When a certain address is applied to the input, the multiplexer acts as a four-input AND gate so that only when all four inputs are asserted will the nanowire go high. In this way larger memories can be built using a minimal number of microscale wires.

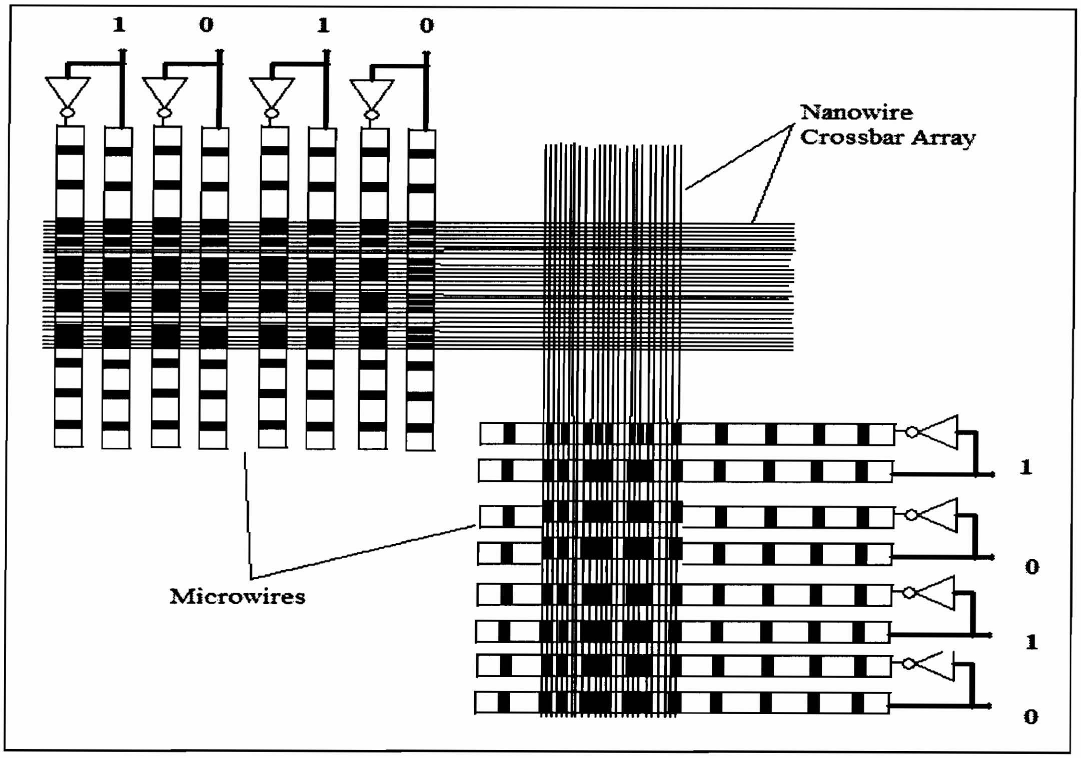


Fig 2.2.1 Crossbar Memory

The crossbar component of the memory architecture described above can be built using either carbon nanotubes or silicon nanowires and hence there are two types of crossbar arrays the Single walled carbon nanotube (SWNT) arrays and Silicon nanowire (SiNW) arrays. The bistable molecular switch at the crosspoints or junctions can be of two types either a single molecule that exhibits a bistable behavior or a Field Effect Transistor (FET) obtained as a result of chemically treating the junction.

We have presented three different crossbar array models, in the first model the crossbar array is made of carbon nanotubes and in the last two models the array is made of silicon nanowires. We simulate and study only the two models in which the crossbar array is made of silicon nanowires – "Leibers Model" and "Molecular Model".

2.2.1 SWNT-Based Nonvolatile Random Access Memory[10]

Nanometer-diameter single-walled carbon nanotubes exhibit unique electronic, mechanical, and chemical properties that make them attractive building blocks for molecular electronics. Depending on diameter and helicity, SWNTs behave as one-dimensional metals or as semiconductors, which, by virtue of their mechanical toughness and chemical inertness, they are ideal materials for creating reliable, high-density input/ouput (I/O) wire arrays. The crossbar array using SWNTs is formed using a set of parallel SWNTs on a substrate and a set of perpendicular SWNTs that are suspended on a periodic array of supports. Each cross point in this structure corresponds to a device element with a SWNT suspended above a perpendicular nanoscale wire.

The bistability of the crosspoint is realized from the interplay of the elastic energy, which produces a potential energy minimum at finite separation (when the upper nanotube is freely suspended), and the attractive van der Walls energy, which creates a second energy minimum when the suspended SWNT is deflected into contact with the lower nanotube. These two energy states correspond to well defined OFF and ON states. A device element can be switched between these well defined OFF and ON states by transiently charging the nanotubes to produce attractive or repulsive electrostatic forces as shown in Fig 2.2.2. At each cross point in the array the suspended (upper) SWNT can be either in the separated OFF state or the ON state in contact with the perpendicular nanotube on the substrate (lower SWNT). The ON/OFF information at the crosspoint can be easily read by measuring the resistance of the junction and, moreover, can be switched between the OFF and ON states by applying voltages pulses at the cross point. This crossbar array model yields a highly integrated, fast, and macroscopically addressable nonvolatile random access memory (RAM) structure that promises significant density and speed improvements.

One important limitation of arrays that use resistive crosspoint connections is the possibility of spurious paths. That is, current can go both ways through such a connection creating other, unintended paths, possibly having a vertical line turn on that should be off.

There are several other issues that need to be considered before we could realize a functional nonvolatile RAM using SWNT. Currently the most serious is that most SWNT manufacturing techniques generate a random distribution of metallic and semiconducting tubes.

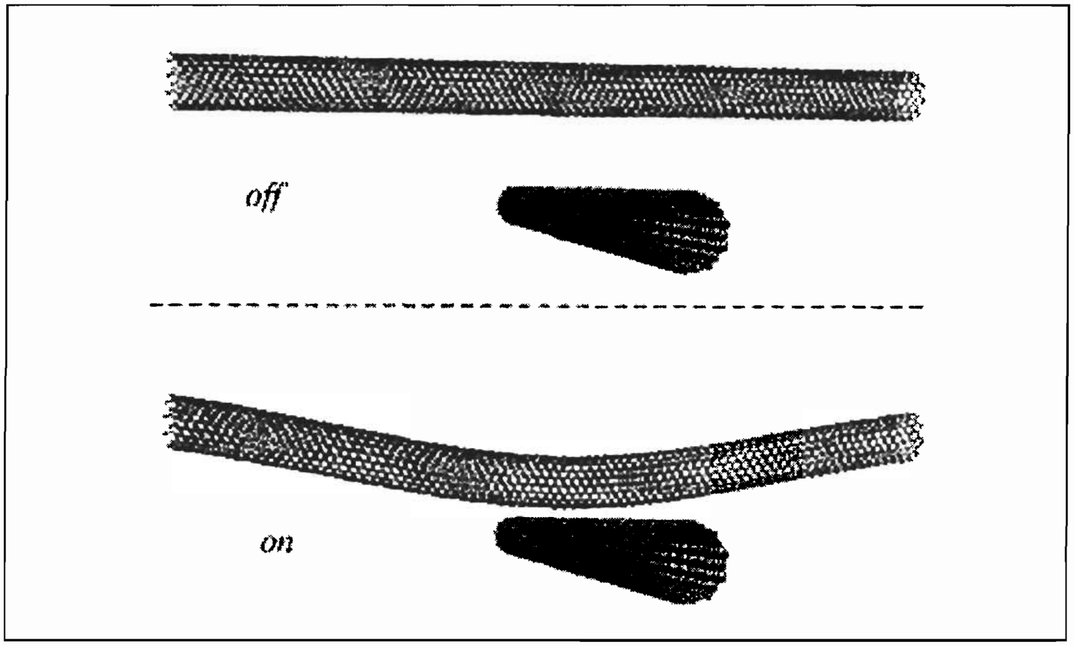


Fig 2.2.2 Carbon nanotube switch

2.2.2 Molecular Crossbar Array Architecture[9]. The crossbar architecture is a general approach for molecular circuits. A molecular crossbar consists of two parallel planes of molecular wire arrays separated by a thin chemical interlayer. The interlayer has specific electrochemical properties. Each plane is made up of many parallel molecular wires or nanowires and all the nanowires in each plane are of same type. The two planes are placed in such a way that the wires in one plane cross the wires in the other plane at right angles. At the junction points where the wires of two planes cross each other a junction is formed. These junctions or crosspoints can be configured via certain applied voltage levels to implement a number of switch configurations, or they can be left unconfigured so that the junction can be treated as always "off", since there is no electrical contact between the nanowires. These can be fabricated through chemical self-assembly or nanoimprint lithography, both fabrication techniques are inexpensive. This architecture is comparatively simple and can be connected to larger-scale external circuits (conventional circuits) to provide I/O to the molecular devices. The crossbar circuits can be used as general programmable devices, such as PLAs, or they can be used as large memories, where we store logic values '0' or '1' at the junctions.

2.2.3 LEIBERS MODEL – *Nanowire Crossbar Arrays As Address Decoders for Integrated Nanosystems*[2]

This model is based on the scalable crossed-nanowire field-effect transistor (cNW-FET) architecture, in which each crosspoint forms a field-effect transistor. By applying different voltage potentials to a horizontal line, each input nanowire line can turn on and off specific output lines, with the horizontal wires creating the "gate" of the FETs and vertical wires the "source/drain." This basic array structure functions as an address decoder. The problem with spurious current paths is solved in this model since there is no electrical connection between horizontal and vertical lines.

When a voltage is applied to a row nanowire in a crossbar array, it will affect each of the output nanowires in the same way, which makes it difficult to perform selective addressing of elements. Multiple effects are possible by differentiating the crosspoints so that each input affects only specific output crosspoints in the array. This is done by modifying the gate to create two states[1]. This allows the creation of an arbitrary logic device, known as a PLA (Programmable Logic Array). A memory can also be build, though in this case it would be Read-Only Memory, since the programming is done off-line before the circuit is used. For example, for an array of size NxN, when one output nanowire is turned on or off by a single input, differentiation of diagonal elements of a square array produces an addressing code where the row nanowire M will address the column nanowire M. This idea can be generalized to enable a small number of output NWs if two or more inputs are used to turn on or off a given output, or similarly, a small number of wires could address a much denser array of nanowires, as is required to bridge between micro and nanoscale features. The cNW-FET devices and arrays were assembled from silicon nanowire building blocks with fluid-directed assembly.

[1] Lieber is somewhat vague as to exactly how this modification is performed.

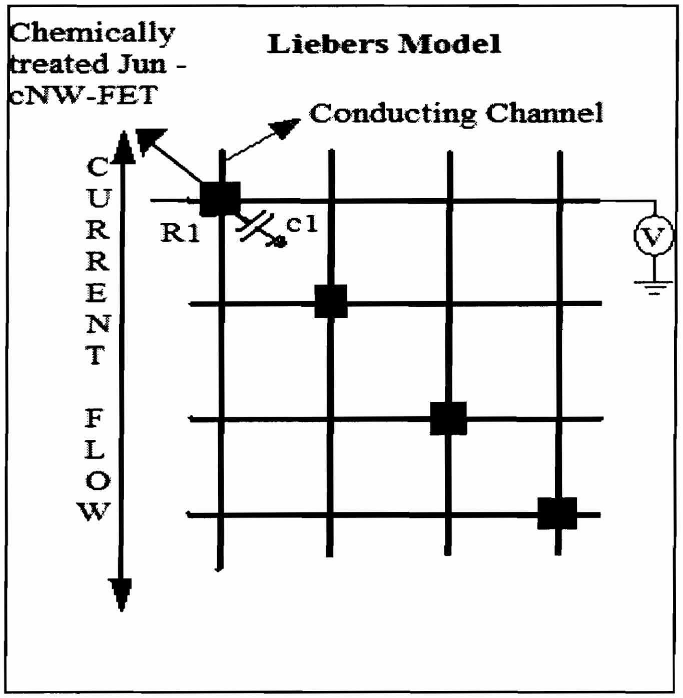


Fig 2.2.3 Leiber Model

The cNW-FET crosspoint array defines an address decoder architecture that has the desired functionality and is reasonably scalable. This decoder can serve as an approach for bridging between microscale wires and dense nanoscale arrays. In addition real functioning circuits have been built in Lieber's lab. Consequently, we have chosen this nano-grid architecture as one of the models used for simulation. The model is shown in above Fig.2.2.3.

2.2.4 MOLECULAR MODEL – *Nanoscale Molecular-Switch Crossbar Circuits*[1].

This model uses molecular cross-point devices incorporating an amphiphilic bistable rotaxane sandwiched between two metal nanowires forming a reversible, electrically toggled switch. Thus the basic element in the circuit is the NW/rotaxane/NW junction formed at each crosspoint, which acts as a reversible and nonvolatile switch. The rotaxane molecule consists of two mechanically interlocked components: a dumbbell encircled by a ring as shown in Fig 2.2.4. The large stoppers - one hydrophobic and the other hydrophilic - at either end of the central shaft makes the molecules amphiphilic, and also create a large area for each molecule in a monolayer film.

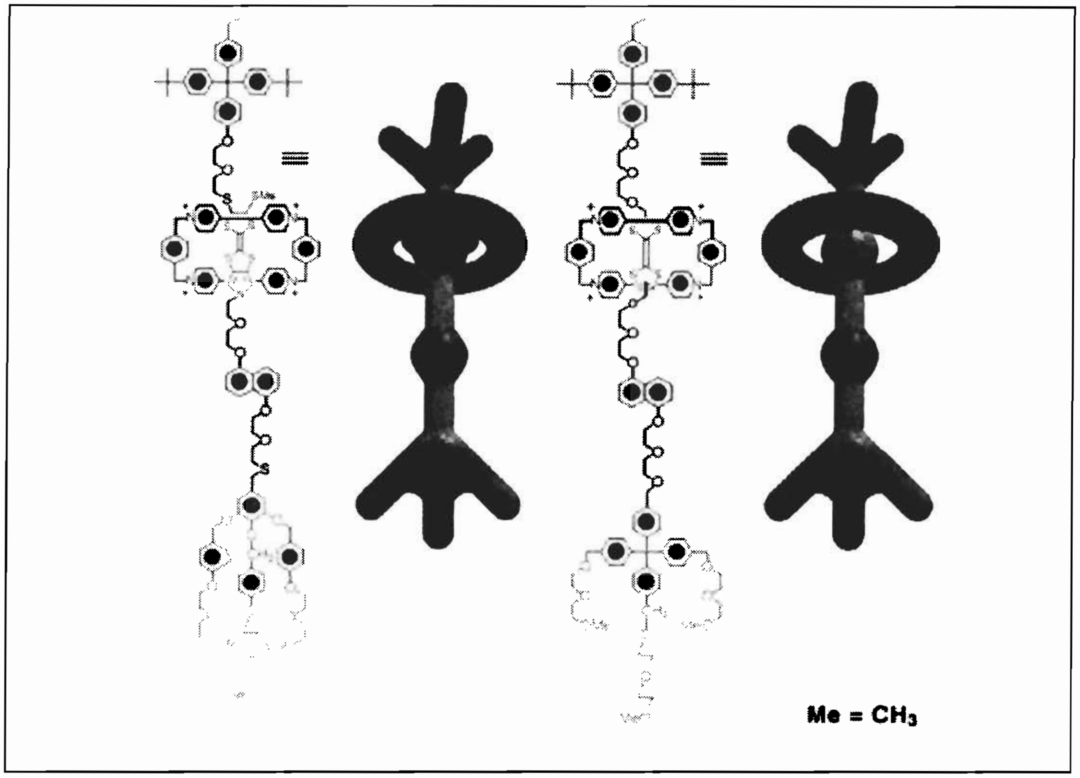


Fig 2.2.4 Rotaxane switch – "Open" & "Close" Configuration

In order for the current to flow from one nanowire to another the switch connecting the two nanowires must be "ON". In this model we calculate the total number of junctions in the current path as the total number of junctions that are "ON" during current flow. Thus for delay calculations we consider the resistance and capacitance of all the "ON" junctions in the current path. This model was used for simulation. The model is shown in Fig.2.2.5.

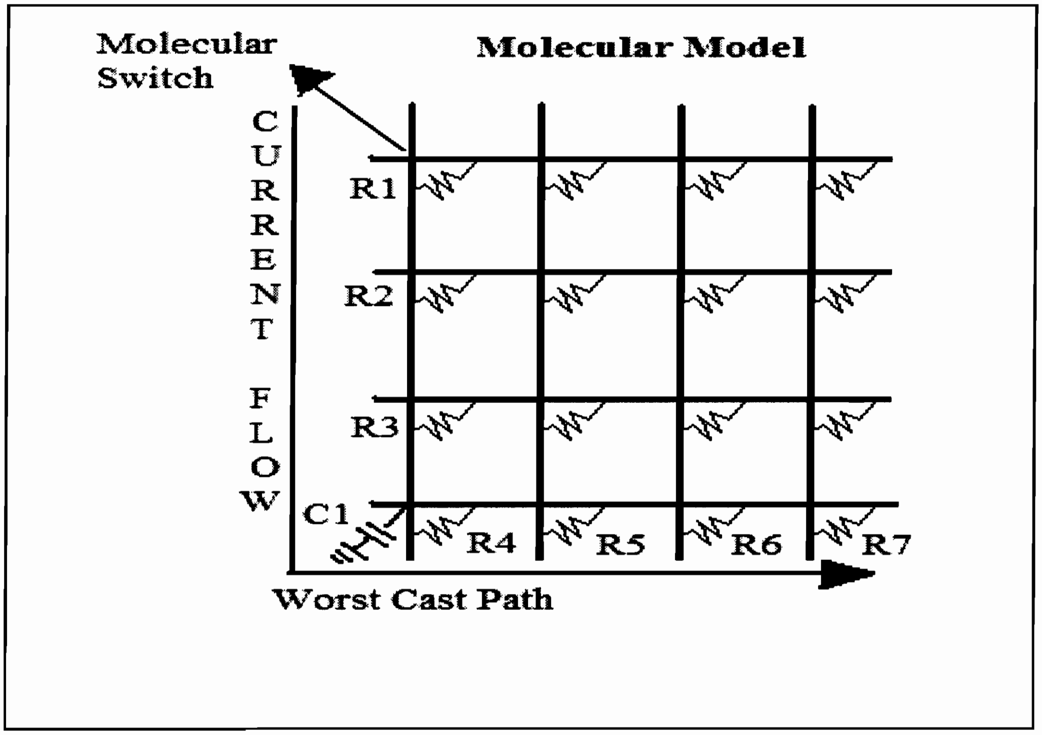


Fig 2.2.5 Molecular Model

2.3 CROSSBAR CIRCUITS AND DEVICES

2.3.1 Molecular Field Programmable Gate Arrays[11]. Field Programmable Gate Arrays are reconfigurable chips that can be programmed after fabrication to implement a desired circuit. In fact, most FPGAs today can be reprogrammed many times. Universal logic blocks, as used in an FPGA, normally consist of 4-bit look-up tables and can be programmed to implement any arbitrary logic function of 4 bits. The programmable switch block and interconnection network provide flexible routing channels to connect the logic blocks into different structures. The advantage of using FPGAs is that their flexibility allows arbitrary functions, in some cases, the ability to configure around defects. But FPGAs have increased area-delay product due to the overhead of the storage elements that hold the program state, as well as the “general” purpose, switchable interconnect. The basic nano-grid architecture with molecular switches at the crosspoints can allow the implementation of a simple “programmable logic” device. The molecular switches in their closed state act as a diode. Such a grid array can also be used as a memory. Fig 2.3.1 shows the implementation of an AND gate using the molecular model crossbar. One disadvantage of the diode-resistor logic is the lack of signal restoration. A molecular latch based on molecular resonant tunneling diode (RTDs) can be used for voltage restoration and I/O isolation. Fig 2.3.2 shows a molecular latch using two RTDs.

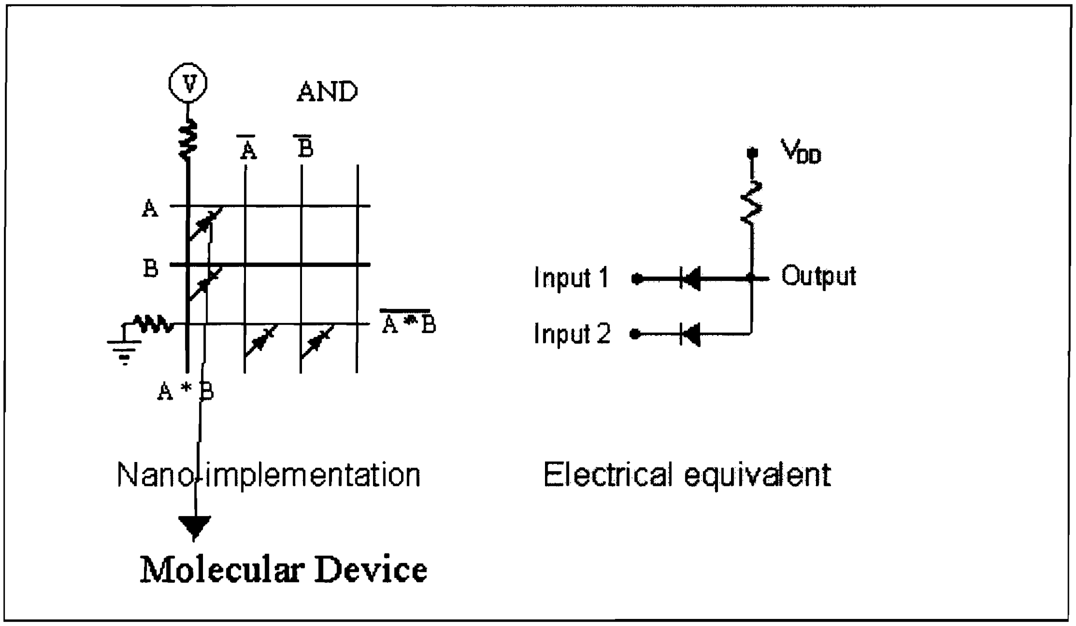


Fig 2.3.1 "AND" gate using Molecular Crossbar

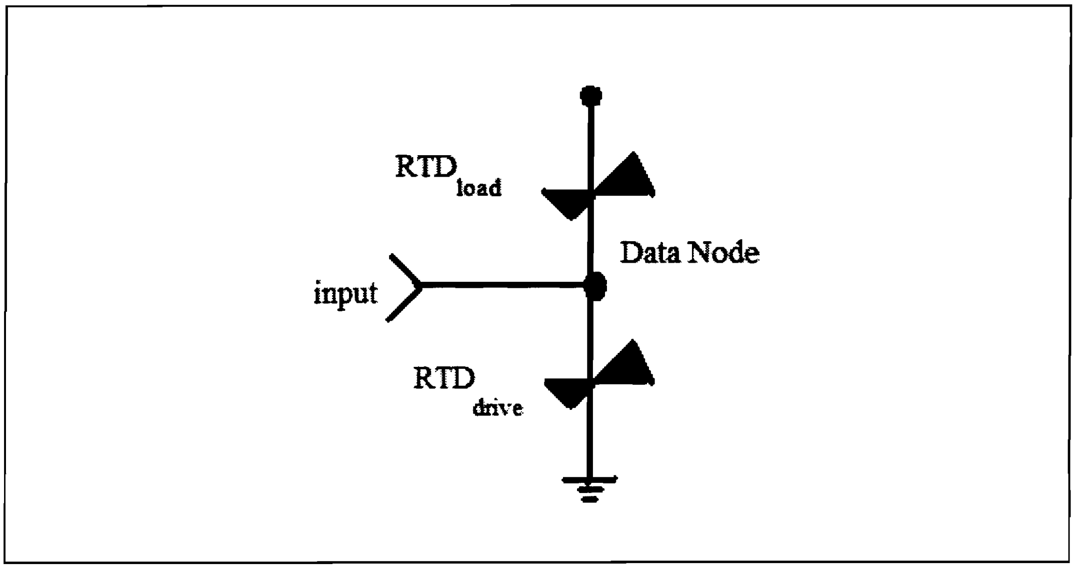


Fig 2.3.2 Molecular Latch using RTDs

The fundamental building block of a molecular FPGA is the 2-D crossbar array. These nanoblocks are can be viewed as universal logic blocks that can be programmed for any logical relation between input and output bits. They can also be configured as simple switch blocks for signal routing, though given the slower speed of molecular scale logic, it is not clear how much routing will actually be done at the molecular scale. Fig 2.3.3 shows the schematic of a nanoblock. It is composed of three sections: (1) A molecular logic array (MLA) is the crossbar array network with molecular switches at crosspoints used for logic implementation; (2) the latches (inline NDRs) that are used for signal restoration and latching at the outputs for sequential circuit implementation; and (3) the I/0 area for connecting the nanoblock to its neighboring blocks through switch blocks.

The molecular FPGA is fault tolerant because of its inherent redundancy and regularity, and its rich interconnect capabilities. The impact of a fault is limited to a small portion of the FPGA or even a small portion of a nanoblock. At this point, the molecular self-assembly process appears as if it will only be useful for implementing regular, periodic structures. The fundamental component of the molecular FPGA, the molecular switch, can be programmed without extra control signals and can hold its state without extra memory elements. Since the switch behaves like a diode in its ON state, it is suitable for resistor-diode type of logic. The molecular FPGA is promising for realizing functional circuits from molecular-scale devices.

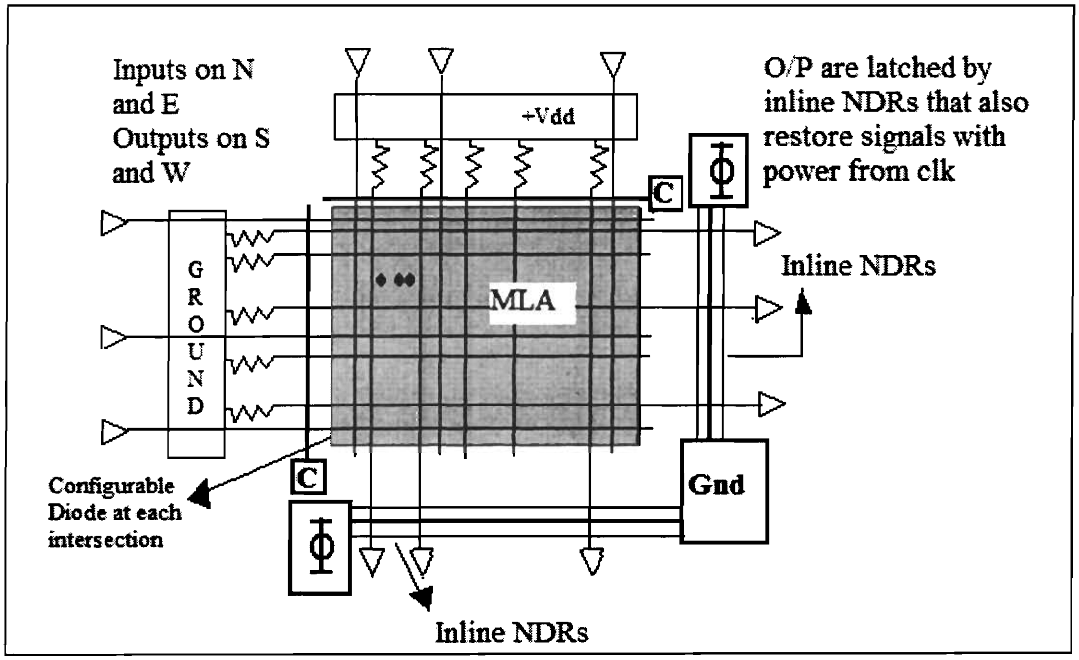

Inputs on N and E Outputs on S and W
+Vdd
C
Φ
O/P are latched by inline NDRs that also restore signals with power from clk
G R O U N D
MLA
Inline NDRs
Configurable Diode at each intersection
C
Φ
Gnd
Inline NDRs

Fig 2.3.3 Nanoblock

2.3.2 Logic Gates and Computation from Assembled Nanowire Building Blocks[12].

Semiconductor nanowires (NWs) can be used for assembling a range of nanodevices including FETs, p-n diodes, bipolar junction transistors and complementary inverters. NW devices can be assembled in a predictable manner because of the electronic properties and dimensions of the NWs can be precisely control during synthesis.

A nanoscale p-n junction and arrays of these junctions can be assembled using p-type silicon (p-Si) and n-type gallium nitride (n-GaN) NWs. The p-n junction and FET arrays can be configured as OR, AND, and NOR logic gates with gain. Current-voltage measurements show that these p-n junction devices exhibit the current rectification characteristics of p-n diodes with a typical turn-on voltage of about 1.0v. A nanoscale FET, specifically a p-channel FET with both a nanoscale conducting channel and a nanoscale gate, can be formed using n-type gallium nitride and p-type silicon crossed NW structure. And due to the small gate area, they will switch fairly quickly, though because of slow interconnect that may not be as useful as we would like.

These structures are called crossed NW FETs (cNW-FETs). The current-voltage data measurements on cNW-FET show a large decrease in conductance with increasing gate voltage. Thus logic devices can be made by use of the primary diode and FET devices. A two-input OR gate can be assembled using a 2(p) by 1(n) crossed p-n junction array with the two p-Si NWs as inputs and the n-GaN NW as the output. Similarly an AND gate can be assembled using 1 (p-Si) by 3 (n-GaN) multiple junction array. Also a logic NOR gate can be assembled by using a 1 (p-Si) by 3 (n-GaN) cNWFET array. The controllable electronic characteristics and reproducible properties of the assembled SiNW devices make it possible to realize various logic circuits at nanoscale.

2.3.3 CMOS/Nano Co-Design For Crossbar-Based Molecular Electronic Systems[13].

A crossbar junction device model is shown in Fig 2.3.4. The model consists of diodes representing the on-state behavior and the off-state behavior, respectively, in parallel with a parasitic junction capacitance. The model can be switched to the on state by applying a voltage bias (V_d) greater than the on-state threshold (V_{thresON}). In the on state, the device follows the behavior of the diode on the left branch of the circuit. Similarly by applying a reverse voltage bias greater than the threshold, causes the device to follow the behavior of the diode in the right branch of the circuit. This model can mimic variety of crossbar junction devices based on the threshold values and junction capacitances.

A circuit model of a crossbar network is show in Fig 2.3.5. This crosspoint circuit has an input vector that is applied to the rows and the output vector is generated by the columns. R_{wr} represents the lumped resistance of the row nanowire and the contact resistance at the input. The resistance R_{wc} represents the lumped column nanowire resistance and contact resistance. The resistance R_{pd} represents the pull down resistance added to the circuit. A single crosspoint can be selected in the circuit by applying a logic "1" to the selected row and a logic "0" to the nonselected rows and the column outputs are placed at logic "0". The state of the selected junction is determined by the output voltage or by the output current leaving the selected column.

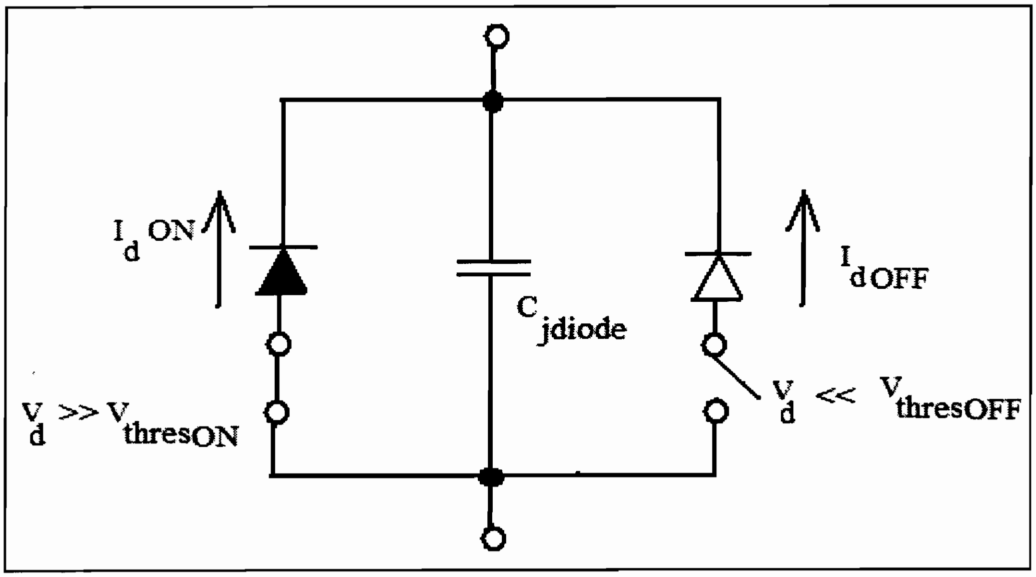


Fig 2.3.4 Crossbar Junction Device Model

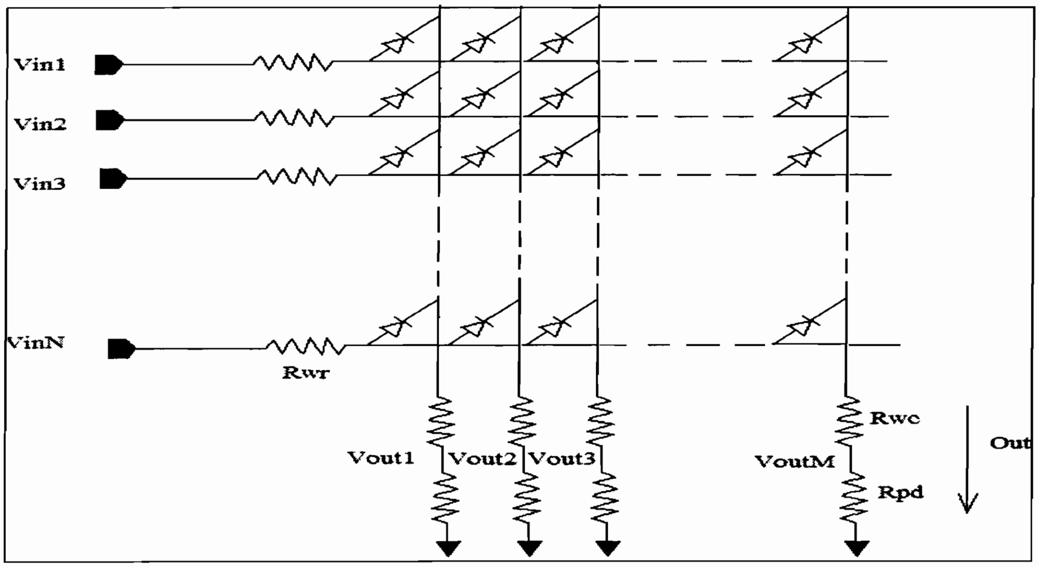


Fig 2.3.5 Crossbar Array Circuit model

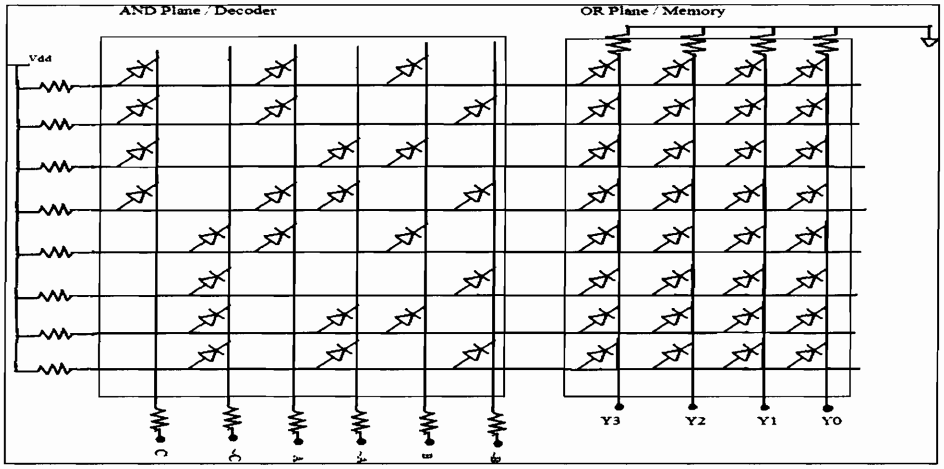


Fig 2.3.6 Decoder-Crossbar Array Memory

Since the crossbar array has poor signal restoration by itself it cannot be used for memory or logic. A decoder/crosspoint array combination provides the framework for addressing memory for data storage as well as memory based logic, decoders using stochastic computational elements or stochastic decoders were suggested for this purpose. Fig 2.3.6 shows a three-input decoder combined with a crossbar array for implementing a memory. The diode-resistor logic of the crossbar array does not include an inversion function hence both the address signals and their complements are needed. The circuit is a look-up table consisting of rows of diode-resistor AND gates forming the decoder and columns of diode-resistor OR gates forming the memory array. Interfacing between

CMOS and nanoscale circuitry is important and creates problems due to the difference in the signal levels of two circuits. The input signal swing for the crossbar technology will most likely be different than the operating voltage levels for the CMOS, hence we need to use CMOS level shifters to drive the decoder address lines. Also sense amplifiers are needed to restore the crossbar output signal to CMOS voltage levels.

There are numerous possibilities for assembling circuits and devices at nanoscale. The circuits and devices presented above are examples of circuits that can be built using crossbar arrays made of SiNWs. We have presented circuits and devices that are necessary for realizing a nanoscale memory.

Chapter III
SIMULATION

This chapter describes simulation of the crossbar array network. The simulations were carried out in a step-by-step manner with increasing level of detail at each step. The first step was the simulation of a silicon nanowire and the study of its electrical characteristics. The next step was the simulation of a simple crossbar array made of silicon nanowires with crosspoints. Complexity was then added by increasing the size of the crossbar array. The next step was to add the "RC" delay calculations to the crossbar array simulations. The delay calculations were done on two models "Leibers" model and "Molecular" model.

3.1 NANOWIRE SIMULATION

In this first simulation step the electrical characteristics of a silicon nanowire were modeled. The primary goal of this step was to get a rough order of magnitude estimate of silicon nanowire electrical resistance.

3.1.1 Nanowire Resistance The resistance of the silicon nanowire was calculated using the general resistance formulae,

$$\mathbf{R = (\rho \times L) / A}$$

$$\mathbf{A = \Pi \times D^2 / 4}$$

Where,

R – Resistance of the nanowire

ρ – Resistivity of the nanowire

L – Length of the nanowire

A – Area of the nanowire

D – Diameter of the nanowire

We used conservative estimates of the parameters and calculated the highest possible silicon nanowire resistance. A simple model was created which computes nanowire resistance for a given length and diameter of wire. As with the entire simulation, this model was implemented in MATLAB ("Exp0_nanowire.m" is the MATLAB listing of the model). A typical nanowire diameter is of the order of 10 – 15 nm and the typical length of a nanowire is on the order of 15 μm. The resistivity of a typical silicon nanowire was found experimentally to be of the order of 10 ohm-cm. By using worse case parameters, the silicon nanowire resistance was determined to be $8.4848E^{+9}$ ohms.

3.1.2 Simple and Comprehensive Models. The next step during this simulation was to study the range of nanowire resistance for nanowires of various dimensions. The first case considered was determine the resistance of nanowires for varying nanowire diameters. In this case the length of the nanowire was kept constant at 10 μm, which is the average length of the fabricated nanowires, and the diameter of the nanowire was varied from 5 to 20 nm with 2 nm increments. The resistivity was maintained at a constant 10 ohm-cm. This model is described in "Exp1_nanowire.m", both textual and graphical outputs of the model have been generated.

The next case considered was the resistance of nanowires for various lengths. In this case the diameter of the nanowire was kept constant at 10 nm, the average diameter of real nanowires, and the length of the nanowire was varied from 1 to 20 μm with 2 μm increments. The resistivity was maintained constant at 10 ohm-cm. This model is described in "Exp1_1_nanowire.m.". The objective of the above study was to obtain the range of nanowire resistances for typical lengths and diameter of nanowires. The range for resistance for the first case was $50.93E^{+9}$ to $3.183E^{+9}$ ohms and for the second case were $1.273E^{+9}$ to $25.465E^{+9}$ ohms. A comprehensive model that calculates resistance for various combinations of length and diameter of nanowire is described in "Exp2_nanowire.m". The comprehensive model calculates nanowire resistance for nanowire lengths 1 – 20 μm and nanowire diameters 5 – 25nm. The difference between the simple model and the comprehensive model is that the latter calculates, for each length of the nanowire, the resistance of the nanowire for a range of nanowire diameters.

A similar model that calculates, for each diameter of the nanowire, the resistance of the nanowire for a range of nanowire lengths is described in "Exp2_1_nanowire.m'.

3.1.3 Simulations. Using the comprehensive models we were able to vary the resistivity, length and diameter of the nanowires simultaneously. The length was varied between 1 μm and 20 μm and the diameter of the nanowire was varied between 5 nm and 25 nm. A summary of the simulation results is presented in the results chapter.

3.2 SIMULATION OF NANOARRAY

The first simulation step gave us an approximate nanowire electrical resistance for a range of physical parameters. The next step was to simulate an array of multiple silicon nanowires. In this simulation step we estimated the length of the nanowires for various sized arrays, crossbar array area, and number of crossbar junctions per unit area. From these data we could then estimate the maximum crossbar array size and the corresponding array area.

3.2.1 Size of Nanoarray.

The size of the nanoarray can be calculated from the number of row and column nanowires. For example a nanoarray of size "2 x 3" has "2" row nanowires and "3" column nanowires. Nanoarrays can be either symmetric (the same number of inputs as outputs) or asymmetric (the numbers of input and outputs differ). For simplicity, we have only simulated symmetric nanoarrays. The length of the nanowire changes with the size of the nanoarray. We have created a simple model for calculating the length of the nanowire for various sized nanoarrays. The model takes into consideration the length of the crossbar array junction and the nanowire pitch, i.e., the distance between the centers of parallel nanowires. Also, the model includes extra wire length for creating contacts to external, CMOS, circuitry. The crossbar array is shown in Fig 3.2.1. The model is described in "Exp3_nanoarray.m". This model calculates the length and resistance of the row and column nanowires. In the case of symmetric nanoarrays the length and resistance of row and column nanowires will be the same, but the model can also calculate the length and resistance of row and column nanowires for non-symmetric arrays.

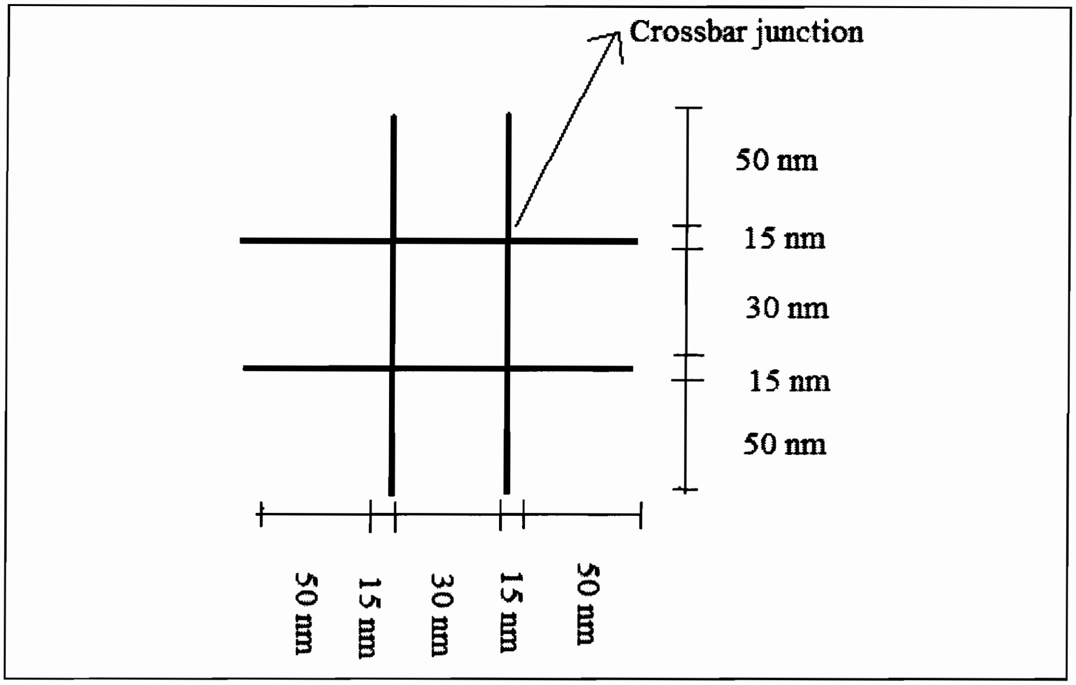


Fig 3.2.1 "2 x 2" Nanoarray

We have made certain assumptions on the lengths of the end contacts and the diameter. The length of the end contact was assumed to be 50 nm, the diameter of the nanowire was assumed to be 15 nm and the pitch was assumed to be, twice the diameter of the nanowire, 30 nm. Thus the length of the row and column nanowires can be calculated as follows,

$$\mathbf{L_{col} = 2 * L_{end} + i * D + ((i - 1) * NW_{pitch})}$$

$$\mathbf{L_{row} = 2 * L_{end} + j * D + ((j - 1) * NW_{pitch})}$$

Where,

L_{col} = Length of column nanowire

L_{row} = Length of row nanowire

L_{end} = Length of end contact

i = Number of rows

j = Number of columns

NW_{pitch} = Distance between two adjacent nanowires or nanowire pitch

Thus after calculating the lengths of row and column nanowires the resistance of the nanowires can be calculated using the formulae from the previous simulation step. We have assumed a nanowire pitch of twice the diameter of the nanowire, which should eliminate any possibility of electrons tunneling between adjacent nanowires.

3.2.2 Crossbar Junctions per unit area.

At each intersection of row and column nanowires a crossbar junction is formed. The crossbar junction of the nanoarray is where the data can be stored and read back through the nanowires. The number of crossbar junctions in a nanoarray can be computed from the size of the nanoarray.

$$\textbf{Number of crossbar junctions} = \mathbf{i \times j}$$

Where,

i = Number of rows

j = Number of columns

Thus for a nanoarray of size "3 x 3" the total number of crossbar junctions is "9". The number of crossbar junctions increases with the square of one dimension of the nanoarray. We have created a model that can calculate the total crossbar area and the number of crossbar junctions per unit area for a given nanoarray size. This model is described in "Exp4_ nanoarray.m". The model uses the following calculation,

$$\textbf{Number of junctions per cm}^2 = \mathbf{N_{juns} / A_{CB}}$$

$$\mathbf{A_{CB} = L_{row} * L_{col}}$$

Where,

N_{juns} = Number of crossbar junctions

A_{CB} = Area of crossbar array

As a next simulation step, a model was created that can compute the above thus showing the variation between the length and the resistance of the nanowires, the length of the nanowire and the crossbar area, and the number of junctions per unit area. This model is described in "Exp5_nanoarray.m". This model can compute the above for a range of nanoarray sizes, we computed data for the range of nanoarray sizes between "1 x 1" and "500 x 500". Based on the maximum length of the nanowire that can be fabricated assuming current techniques, we have found that the largest array that can be built is "450 x 450", for which the length of nanowire is approximately 20 µm. We have also estimated a limit on the array size based on the current fan-in and fan-out limitations. The limit was estimated based on the total voltage drop in the worst case path. The voltage drop was estimated by, for a given supply voltage and current, calculating the total resistance in the worst case current path.

3.2.3 Simulations. Three models were developed during this simulation step. Using the simple model, "Exp3_nanoarray.m," we did individual simulations on different nanoarray sizes to measure the length of the row and column nanowires. The simulated nanoarray sizes were mostly symmetrical; we did a few simulations on asymmetrical nanoarrays to study the variation in the row and column nanowires. The array size based on fan-in and fan-out constraints was estimated using the model "Exp3_1_nanoarray.m". We assumed a supply voltage of 2V and current of 100pa. We estimated the voltage drop across the worst case path, i.e. the current flow from the first row to the last column of the array. The maximum array size was found by having a 40% tolerance level. We ignored the effects due to transient currents in our simulations. We found the, with the above tolerance level, largest array size to be "205x205".

The model described in "Exp4_nanoarray.m" was simulated for the same set nanoarrray sizes, and the number of crossbar junctions per unit area was estimated. We studied the relation between the lengths of the nanowires and the nanoarray size using the comprehensive model "Exp5_nanoarray.m". In order to cover the maximum space of nanoarray sizes we performed simulations on the comprehensive model with the nanoarray size between "1 x 1" and "500 x 500"with increments of "15","20" and "25". The relationship between the length of the nanowires and the crossbar array area is exponential for symmetric nanoarrays.

Another interesting result obtained from the simulations is the behavior of the number of crossbar junctions per unit area with crossbar array area. The number of crossbar junctions per unit area becomes almost a constant for nanoarray sizes greater than "350 x 350". The increase in the number of junctions per unit area after this nanoarray size is very minimal that the increase can be neglected. This simulation step has yielded results on the crossbar array size relationship and length of the nanowires on the storage capacity of the arrays.

3.3 DELAY CALCULATIONS

As discussed in chapter 2, crossbar array architectures are most suited for implementing memories. When designing with crossbar arrays it is important to the signal delays of the arrays. We have assumed that all delay is RC and that all other types of electrical influence (e.g., inductance, inter-wire tunneling and coulomb blockage) are minimal. Our delay calculations are done for the two models: the "Leibers model" and the "Molecular model". The "Molecular" model has a steady state current that flows from one horizontal wire to another vertical wire or vice versa (depending on the "ON" junctions). The "Leiber" model, on the other hand, is electrically isolated. In the "Leiber" model the current flows only through the vertical wires, and there can be some transient current during switching but there is no steady state current from horizontal to vertical wires or vice versa.

3.3.1 Leibers Model Delay. This model is based on a scalable crossed-nanowire field-effect transistor (cNW-FET) architecture in which each crosspoint of the crossbar bar is modified to form a field-effect transistor, which allows specific input nanowire lines to turn on and off specific output lines hence an input row nanowire is made to address a single output column nanowire.

We have made certain assumptions in order to estimate the "RC" delay of this model. The first assumption is that the conducting channels are made of typical SiNW's with a resistivity of 10 Ohm-Cm and that the resistivity of the SiNW at the crosspoint is 0.1 Ohm-Cm. Thus the crosspoint, with low resistance, can be used to switch "ON" or "OFF" selected column nanowires by applying the lower threshold voltage. The "RC" delay involved can be calculated from the size of the nanoarray, worst case delay was calculated for all the cases. The model is described in "Exp6_nanoarray_leiber.m" and shown in Fig 3.3.1. The total "RC" delay was calculated as follows,

C_tot = (C_con * C_jun) / ((2 * C_jun) + (C_con))

R_tot = (2 * R_con) + R_jun + R_wire

$$\mathbf{RC\_delay = R\_tot * C\_tot}$$

Where,

R_con = Resistance of nanowire end contacts

R_jun = Resistance of crossbar junction

R_wire = Resistance of the nanowire in current path

C_con = Capacitance of nanowire end contacts

C_jun = Capacitance of crossbar junction

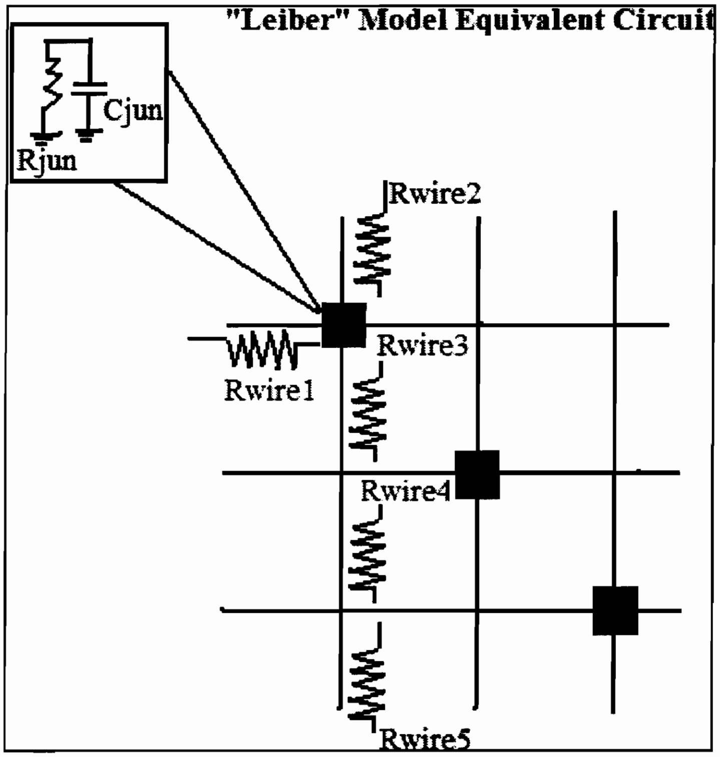


Fig 3.3.1 "Leiber" Model circuit

The resistance of the nanowire in the delay path was calculated in the same manner in which the resistance of the nanowire was calculated in first simulation step (Refer 4.1.1 Nanowire Resistance). The length of the nanowire in the current path for the resistance calculation was obtained as follows,

$$\mathbf{L\_wire = 2 * L_{end} + ((i - 1) * (D + NW_{pitch}))}$$

Using the above equations, the "RC" delay was calculated for different array sizes. The worst-case delay for each case was estimated by measuring the total resistance and capacitance in the longest delay path.

3.3.2 Molecular Model Delay. This model is based on crossbar arrays fabricated using molecular crossbar junctions. The crossbar junction is made up of rotaxane molecules sandwiched between Ti/Pt nanowires. Each junction acts as a reversible and nonvolatile switch. Thus the basic difference between the "Leiber" and "Molecular" models is the way in which the crosspoint is prepared. As described earlier there is no current flow from a horizontal nanowire to a vertical nanowire in "Leiber" model, but in the case of "Molecular" model the current flow is controlled by the resistance at the crosspoints with current flowing from a horizontal nanowire to a vertical nanowire.

In order to calculate the "RC" delay for this model, we have made certain assumptions. The resistance at a crosspoint is high when the molecular switch at the crosspoint is "OFF" and is low when the molecular switch is "ON". We assume that the current flows to the next closest crosspoint with a lower resistance or "ON" without any delay. The "ON" and "OFF" resistance of the crosspoint made of Pt/rotaxane/Ti junction is in the range of "$10^6 - 5x10^8$ " and "> $4x10^9$ ". The model is shown in Fig 3.3.2.

The "RC" delay for this model was calculated as follows,

$$\textbf{C\_tot = (C\_con * C\_jun) / ((2 * C\_jun) + (C\_con)}$$

$$\textbf{R\_tot = (2 * R\_con) + (R\_jun * N}_{\textbf{juns}}\textbf{) + R\_wire}$$

$$\textbf{RC\_delay = R\_tot * C\_tot}$$

The resistance of the nanowire in the current path was calculated in the same manner in which the resistance of the nanowire was calculated in first simulation step. The length of the nanowire in the current path for the resistance calculation was obtained as follows,

$$\textbf{L\_wire = (2 * L}_{\textbf{end}}\textbf{)+ ((N}_{\textbf{juns}}\textbf{ – 1) * NW}_{\textbf{pitch}}\textbf{)}$$

This model is simulated using "Exp6_nanoarray_molecular.m". The "RC" delay was calculated for various nanoarray sizes and the worst-case delay was calculated for all the cases.

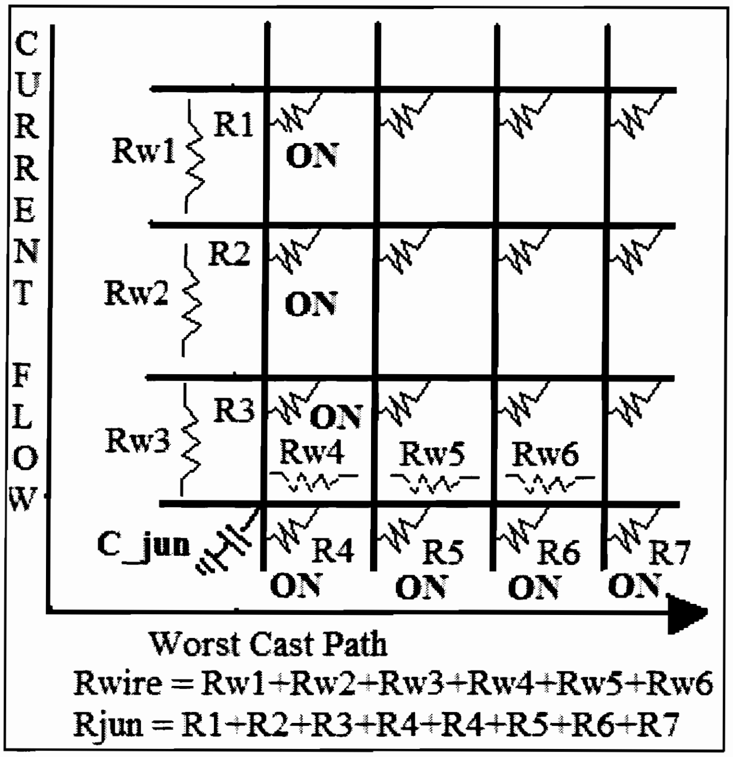


Fig 3.3.2 "Molecular" Model circuit

3.3.3 Simulations. The "Leiber" and "Molecular" models were simulated for the array sizes of "1x1" to "400x400". The total "RC" delay was calculated using the above equations for the two models. The worst-case delay was calculated by measuring the delay in the current path, which included the maximum number of crossbar junctions (only for the "Molecular" model). For example, for an array of size "4x4" the worst-case delay can be measured in the path – "11-12-13-14-24-34-44", where "11" indicates the (row 1, column 1) junction. Alternatively, the worst-case delay path can also be: "11-21-31-41-42-43-44". Both paths would give the same "RC" delay, since the total number of junctions is "7". Thus, in general, current entering the first row and leaving the last column of the nanoarray will include the maximum number of crossbar junctions in its path and hence will produce the largest "RC" delay. The "RC" delay of both models was plotted against the nanowire length and the total crossbar array area. The delays of both models were compared in the simulation “Exp7_newnanoarray_both.m”. The simulation results, delay comparisons and conclusion are presented in the results and conclusions chapter.

3.4 Defects Analysis

When crossbar arrays are assembled, there is a possibility that the nanoscale wires will have poor or nonexistent contacts and the switches at the crossbar junctions may not be functioning properly. In order for the crossbar array to be fully functional, even with the defects in the array, there must be a way to avoid the faulty wires and non-functional junctions. But in order to avoid the defects and still function properly there must be spares nanowires and switches which can be used. Thus a crossbar array can be designed to tolerate the defects by both local wire sparing and array sparing. The method is similar to the way one designs spare rows and columns in conventional DRAM memories.

3.4.1 Yield in Crossbar Array

The percentage of yield for a crossbar array can be calculated from the ratio between the total number of crossbar junctions or crosspoints (that acts as switches) and the number of functional crossbar junctions. For example for a crossbar array of size “3x3”,

Total Number of crossbar junctions	**= 3 x 3 = 9**
If total number of functional CB Juns	**= 5**
Yield	**= 5/9 * 100**
	~= 55%

We consider only two main causes of defects in a crossbar array, a non-functional crosspoint i.e. the switch is either broken or shorted and hence unusable, and broken nanowire i.e. the contact at one end of the nanowire is sufficiently poor as to be unusable. Current research suggests that non-functional crosspoint faults occur in single digit yield percentages and broken nanowire type faults are very unlikely. It also suggests a reliable growth of silicon nanowires which are over 10 um i.e. silicon nanowires of lengths up to 10 - 12 um can be grown without any breaks.

In this simulation step we have performed a preliminary defects analysis on various sized crossbar arrays based on yield percentages and crossbar area.

3.4.2 Defects Analysis with Redundancy

The term "Redundancy" can be defined as, in our case, the total number of spare wires and junctions. The redundant nanowires present in the system are not always used. These wires are used only used to avoid the defects, if any, in the system. The number of redundant wires in the system is determined by the yield percentage. Redundancy is inversely proportional to the yield percentage. But the as the number of redundant nanowires increases the total crossbar array area also increases. Thus we need to compensate on the crossbar array area in order to obtain a good defect tolerance.

For any crossbar array size of "m x n" and yield percentage "Y",

$$\textbf{Total number of crossbar juns} = N_{func} = m * n$$

$$\textbf{Total non-functional juns} = N_{non\text{-}func} = ((100 - Y) / 100) * N_{func})$$

$$\textbf{Number of Redundant nanowires} = N_{redun} = (N_{non\text{-}func})^{(1/2)}$$

Thus for any given crossbar array we can estimate the redundancy using the yield percentage. By finding the square root of the total non-functional junctions we obtain the number of rows and columns of the redundant crossbar array introduced into the system. We then calculate the redundant crossbar array area by estimating the length of row and column nanowire of the redundant array (refer 3.2.1 size of nanoarray).

$$\textbf{Redundant Array area} = A_{redun} = L_{row(redun)} * L_{col(redun)}$$

$$\textbf{Total Crossbar Area} = A + A_{redun}$$

Where,

$L_{row(redun)}$ = Length of row nanowire in redundant array

$L_{col(redun)}$ = Length of column nanowire in redundant array

A = Area of original crossbar array

3.4.3 Simulations

The model was simulated for array sizes between “5x5” and “600x600”. The redundancy, original crossbar array area, total crossbar array area was estimated for all the array sizes. The total crossbar area and original crossbar array area were plotted against the redundancy and crossbar array size. The length of the nanowire, for array sizes greater than “255 x 255”, is greater than 12 um. Thus for array sizes greater than “255x255” we have to take into account the broken nanowire defects. Thus we compensated for the defect by decreasing the total yield percentage. We have done four sets of simulations array sizes between “5x5” to “255x255” with yield 90% and 95% and array sizes between “5x5” to “505x505” with yield 80% and 85%. “Exp8_nanoarray.m” describes the model. A summary of all the simulation results is presented in the results and conclusions chapter.

3.5 Memory Density

Memory density can be obtained by calculating the number of bits that can be accommodated in a unit area. In the case of crossbar array memories the memory density is very high due to the very small occupied by the crossbar array. But to build a memory using crossbar arrays we need to have multiplexer along with the crossbar array in order to read and write data onto the array. Thus a crossbar memory cell can be considered to be a combination of a definite sized nanoscale crossbar array and a microscale multiplexer. The area occupied by the crossbar memory cell will be much higher when compared to the area of just the crossbar array. In this simulation step we have estimated the area of a crossbar memory cell containing a “250x250” crossbar array and a microscale multiplexer and have calculated the number such cells that are required to obtain memory capacities that we are using today and also have estimated the memory density for all the cases with and without redundancy.

3.5.1 Memory density using crossbar Cell

The memory capacity of a crossbar memory cell containing a “250x250” can be calculated as follows,

Total number of bits = 250 * 250 = 62500

Memory Capacity = 62500 / (8*1024)

= 7.62 KB or 7.45e-3 MB

Thus one memory cell has a capacity of around 7.6 KB. Therefore the total number of such cells required to obtain a memory capacity of 256, 512, 1024 and 2048 MB was calculated. Assuming a 90% yield in the crossbar array the array area with redundancy was calculated similar to the previous simulation step (refer 3.4.1). Assuming that the multiplexer occupies around 50 times the total crossbar area (including redundancy) the total area occupied by the cells was estimated for each case, 256, 512, 1024 and 2048MB. The memory density was estimated as follows,

For the case of 256 MB,

$$\textbf{Memory density} = \frac{\textbf{Total number of bits}}{\textbf{Area occupied by the cells to obtain256MB of memory}}$$

3.5.2 Simulations

The simulation model is described in "Exp9_nanoarray.m". The model was simulated to calculate the area of individual crossbar array cell, total area with redundancy (with an assumed yield of 90%) and the final area including the multiplexer area. It was also simulated to obtain the memory densities for 256, 512, 1024 and 2048 MB. The memory densities for all the cases without redundancy was found and compared. A summary of all the simulation results is presented in the results and conclusions chapter.

Chapter IV
RESULTS & CONCLUSIONS

This chapter summarizes the simulation results and draws some conclusions. The Matlab models (.m files) are provided in the appendix. In total, eleven models were developed. Each simulation made specific assumptions for the simulation parameters. Only the more important results are presented in this chapter, other results are presented graphically in the appendix.

Simulation # 1.1 – "R" of nanowire:

Model	**Input Parameters** *(Assumed)*	**Simulation Result** *(Nanowire Resistance)*

Exp0_nanowire.m	D = 15E^{-7} cm,
	L = 15E^{-4} cm,
	ρ = 10 ohm-cm
		8E^{+9} ohms

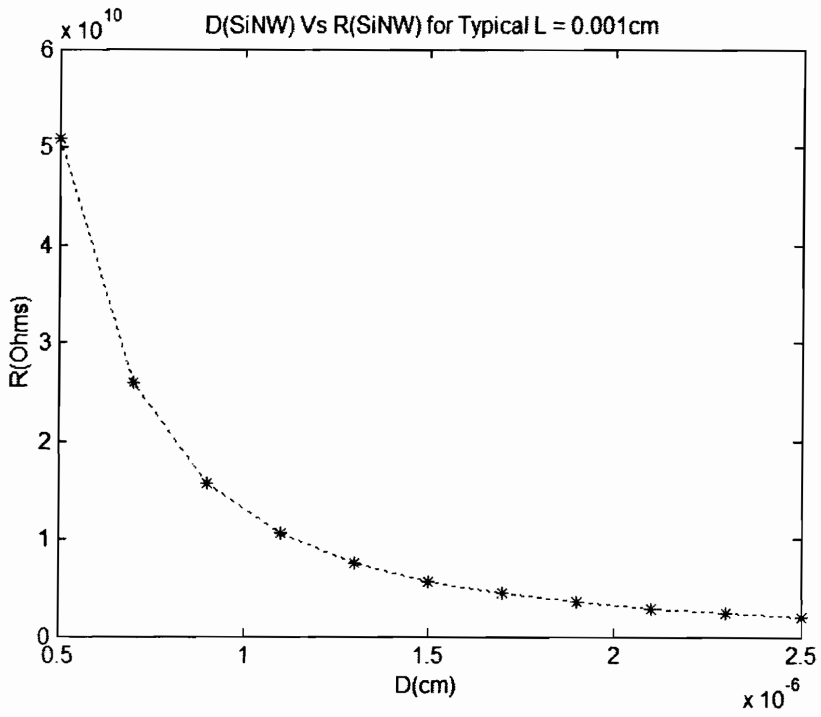
Conclusion # 1.1.1 – Typical resistance of a silicon nanowire is of the order 8.4848E^{+9} ohms.

Simulation # 1.2 – "R" of nanowire for varying "D":

Model: "Exp1_nanowire.m"

Input Parameters (Assumed): L = $10E^{-4}$ cm, ρ = 10 ohm-cm

Diameter (cm)	Resistance (ohms)
$5E^{-7}$	$5E^{+10}$
$7E^{-7}$	$2E^{+10}$
$9E^{-7}$	$1E^{+10}$
$11E^{-7}$	$1E^{+10}$
$13E^{-7}$	$7E^{+9}$
$15E^{-7}$	$6E^{+9}$
$17E^{-7}$	$4E^{+9}$
$19E^{-7}$	$4E^{+9}$



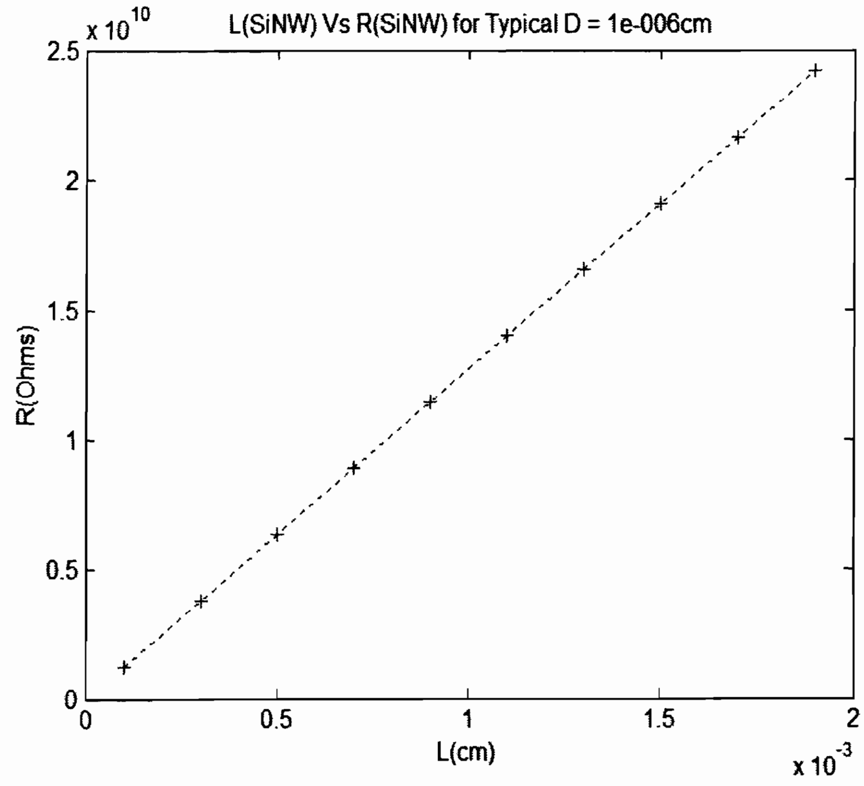
Observation # 1.2.1 – The nanowire resistance decreases exponentially with diameter.

Simulation # 1.3 – "R" of nanowire for varying "L":

Model: "Exp1_1_nanowire.m"

Input Parameters (Assumed): D = 10E^{-7} cm, ρ = 10 ohm-cm

Length **(cm)**	**Resistance** **(ohms)**
1E^{-4}	1E^{+9}
3E^{-4}	4E^{+9}
5E^{-4}	6E^{+9}
7E^{-4}	9E^{+9}
9E^{-4}	11.5E^{+9}
11E^{-4}	14E^{+9}
13E^{-4}	16.5E^{+9}
15E^{-4}	19E^{+9}
17E^{-4}	21E^{+9}
19E^{-4}	24^{+9}



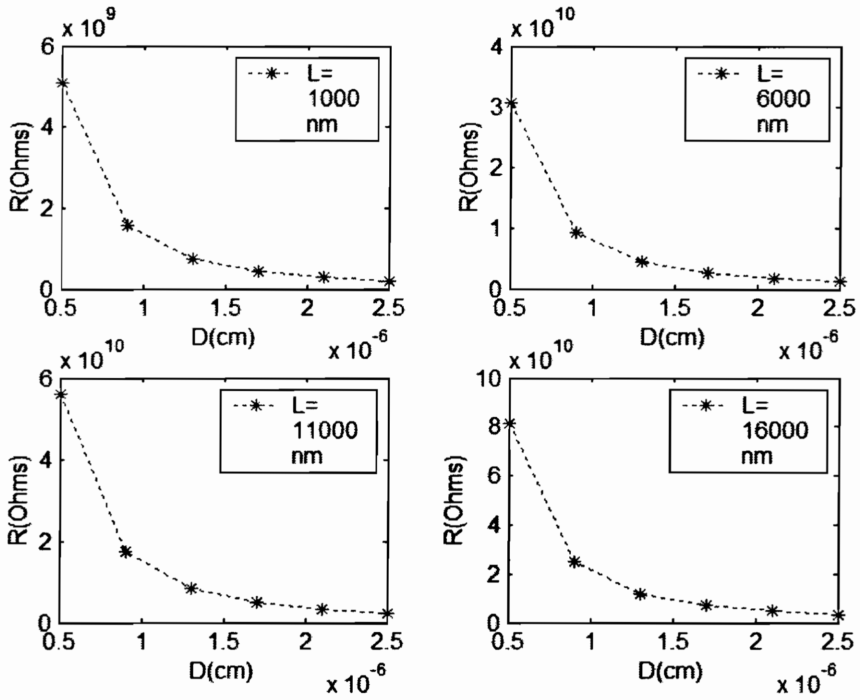
Observation # 1.3.1 – The nanowire resistance increases linearly with length

Simulation # 1.4 – Nanowire Simulation (Comprehensive model)

Model: “Exp2_nanowire.m”

Input Parameters (Assumed): ρ = 10 ohm-cm

L	**Diameter**	**Resistance**
(cm)	**(cm)**	**(ohms)**
$1E^{-4}$	$5E^{-7}$	$5E^{+9}$
	$9E^{-7}$	$1.5E^{+9}$
	$13E^{-7}$	$7.5E^{+8}$
	$17E^{-7}$	$4.4E^{+8}$
	$21E^{-7}$	$2.8E^{+8}$
	$25E^{-7}$	$2E^{+8}$
$6E^{-4}$	$5E^{-7}$	$3E^{+10}$
	$9E^{-7}$	$1E^{+10}$
	$13E^{-7}$	$4.5E^{+9}$
	$17E^{-7}$	$2.6E^{+9}$
	$21E^{-7}$	$1.7E^{+9}$
	$25E^{-7}$	$1.2E^{+9}$
$11E^{-4}$	$5E^{-7}$	$5.6E^{+10}$
	$9E^{-7}$	$1.7E^{+10}$
	$13E^{-7}$	$8.2E^{+9}$
	$17E^{-7}$	$4.8E^{+9}$
	$21E^{-7}$	$3.1E^{+9}$
	$25E^{-7}$	$2.2E^{+9}$
$16E^{-4}$	$5E^{-7}$	$8.E^{+10}$
	$9E^{-7}$	$2.5E^{+10}$
	$13E^{-7}$	$1.2E^{+10}$
	$17E^{-7}$	$7E^{+9}$
	$21E^{-7}$	$4.6E^{+9}$
	$25E^{-7}$	$3.2E^{+9}$



Observation # 1.4.1 – From above simulations the nanowire resistance, for diameters 10 – 20 nm and lengths 10-15 μm, ranges between $0.2037E^{+9}$ and $8.1487\ E^{+10}$.

Simulation # 1.5 – Size of Nanoarray

Model	**Input Parameters**	**Simulation Result**	
	(Assumed)	*"L"*	*Array Size*
Exp3_nanoarray.m	$D = 15E^{-7}$ cm $\rho = 10$ ohm-cm $L_{end} = 50E^{-7}$ cm $NW_{pitch} = 30E^{-7}$ cm	20 μm	450x450

Conclusion # 1.5.1 – The maximum size of nanoarray, based on a typical maximum nanowire length of 20 μm, is "450x450".

Simulation # 1.6 – Size of Nanoarray Based on Fan-in and Fan-out

Model: Exp3_1_nanoarray.m

Input Parameters(Assumed): D = 15E^{-7} cm, ρ = 10 ohm-cm

L_{end} = 50E^{-7} cm, NW_{pitch} = 30E^{-7} cm

V = 2V , I = 100 pa

Array Size	**Resistance** (ohms)	**Voltage Drop** (V)
5x5	1.95E^{+8}	1.98
55x55	1.9E^{+9}	1.81
105x105	3.6E^{+9}	1.64
155x155	5.3E^{+9}	1.47
205x205	7E^{+9}	1.30
255x255	8.7E^{+9}	1.12
305x305	1E^{+10}	0.95
355x355	1.21E^{+10}	0.8

Conclusion # 1.6.1 – The maximum array size after accounting for the fan-in and fan-out constraints is “205x205”

Simulation # 1.7 – Number of Crossbar Junctions per unit area

Model	**Input Parameters** *(Assumed)*	**Simulation Result** *“CB_{area}”* *“CB_{juns} /cm^2”*
Exp4_nanoarray.m	Array size=“450 x450”,L_{end}=50E^{-7} cm, NW_{pitch}=30E^{-7} cm	4.1351E^{-6} cm^2 4.8971E^{+10}

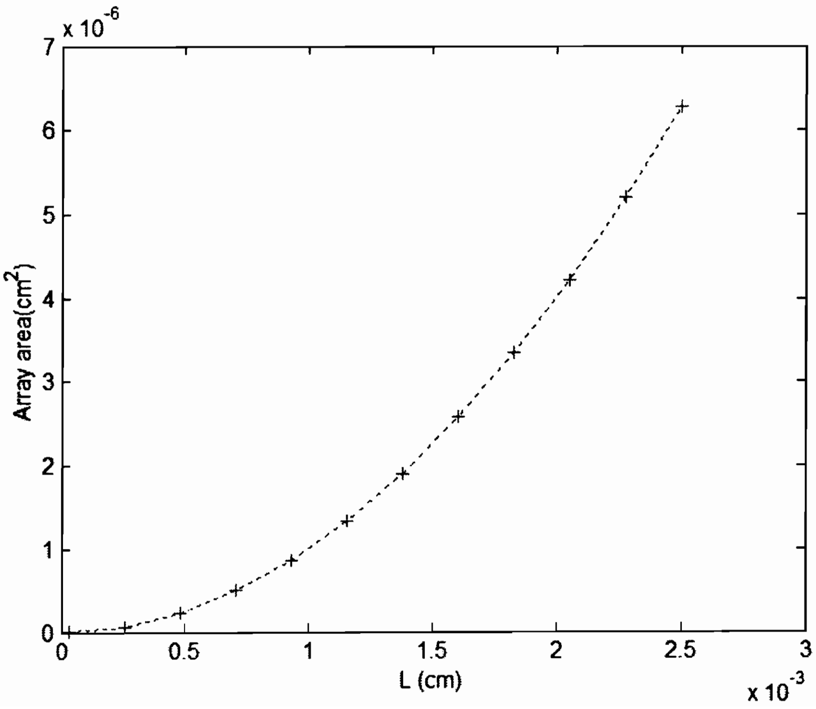
Conclusion # 1.7.1 – The total number of crossbar junction per unit area for a maximum nanoarray size of “450x450” is 4.8971 E^{+10}.

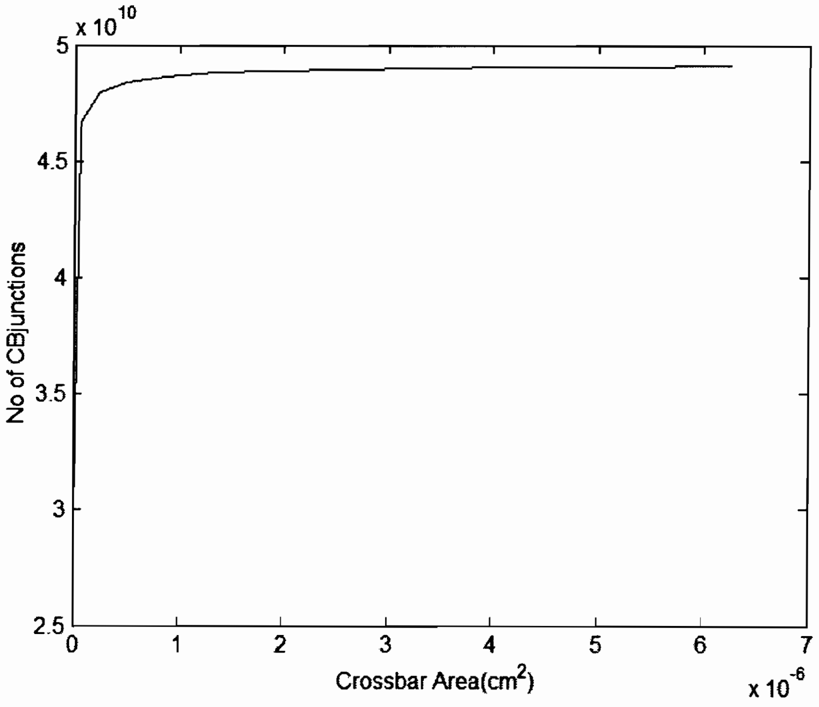
Simulation # 1.8 – Nanoarray Simulation (Comprehensive Model)

Model: "Exp5_nanoarray.m"

Input Parameters (Assumed): D = $15E^{-7}$ cm, ρ = 10 ohm-cm

Array Size	**L** **(μm)**	**Array Area** **(cm^2)**	**CB_{juns}/cm^2**
5x5	0.3	$8.70E^{-10}$	$2.8E^{+10}$
55x55	2.5	$6.47E^{-8}$	$4.6E^{+10}$
105x105	4.8	$2.30E^{-7}$	$4.7E^{+10}$
155x155	7	$4.96E^{-7}$	$4.84E^{+10}$
205x205	9.3	$8.64E^{-7}$	$4.86E^{+10}$
255x255	12	$1.33E^{-6}$	$4.87E^{+10}$
305x305	14	$1.90E^{-6}$	$4.88E^{+10}$
355x355	16	$2.57E^{-6}$	$4.89E^{+10}$
405x405	18	$3.34E^{-6}$	$4.90E^{+10}$
455x455	21	$4.22E^{-6}$	$4.90E^{+10}$
505x505	23	$5.20E^{-6}$	$4.90E^{+10}$





Conclusion # 1.8.1 – The total number of crossbar junction per unit area becomes almost constant for nanoarray sizes greater than "350x350"

Simulation # 1.9 – Leiber Model Delay

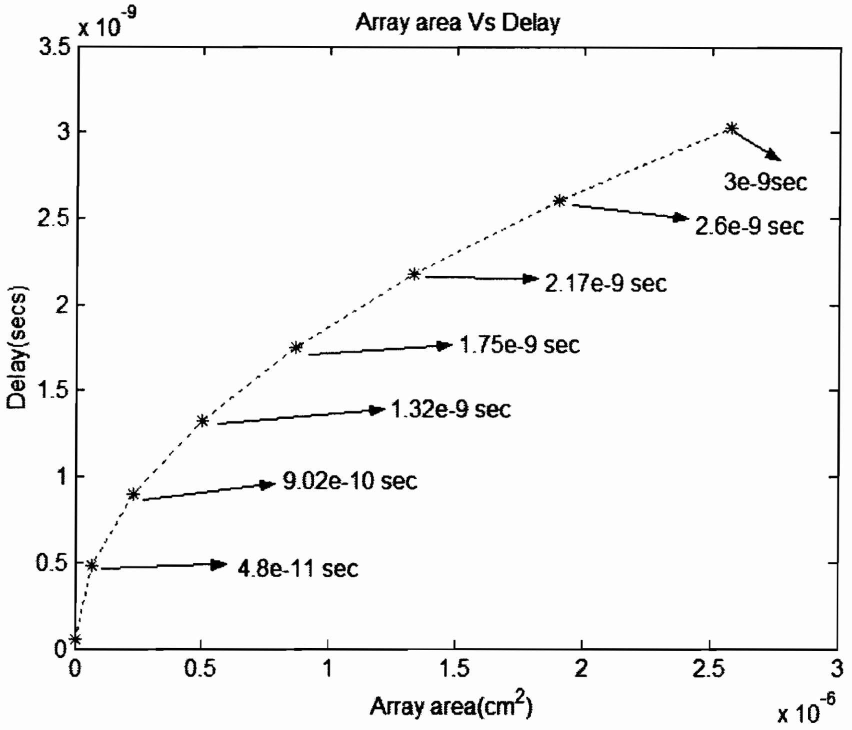
Model: "Exp6_nanoarray_leiber.m"

Input Parameters (Assumed): $D = 15E^{-7}$, $\rho = 10$ ohm-cm,

R_con = $1E^{+6}$, C_con = $1E^{-18}$ F,

R_jun = $6.66E^{+4}$ ohm, C_jun = $1E^{-18}$ F

Array Size	Delay (secs)
5x5	5.30E^{-11}
55x55	4.78E^{-10}
105x105	9.02E^{-10}
155x155	1.32E^{-9}
205x205	1.75E^{-9}
255x255	2.17E^{-9}
305x305	2.60E^{-9}
355x355	3.02E^{-9}



Conclusion # 1.9.1 – The total “RC” delay increases with the array size. This conclusion can be explained as the array size increases the length of the conducting channel in the current path also increases and thus increasing the total resistance in the current path. We can conclude that the total “RC” delay in this model is directly proportional to the length of the conducting channel.

Simulation # 1.10.1 – Molecular Model Delay

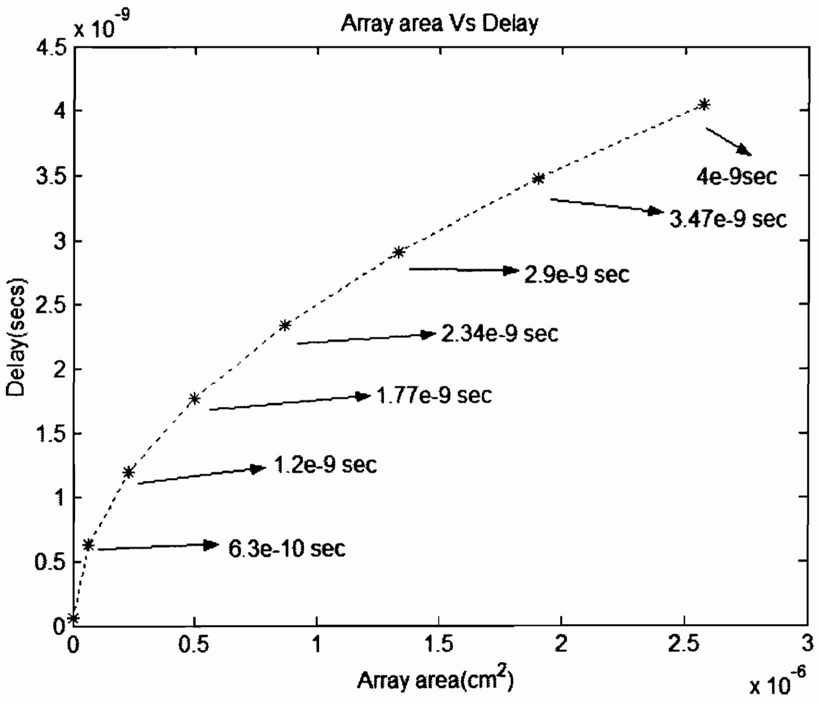
Model: "Exp6_nanoarray_molecular.m"

Input Parameters (Assumed): D = 15E^{-7}, ρ = 10 ohm-cm,

R_con = 1E^{+6}, C_con = 1E^{-18}F,

R_jun = 6.66E^{+4}ohm, C_jun = 1E^{-18}F

Array Size	N_{juns}	Delay (secs)
5x5	9	6.50E^{-11}
55x55	109	6.33E^{-10}
105x105	209	1.20E^{-9}
155x155	309	1.77E^{-9}
205x205	409	2.34E^{-9}
255x255	509	2.90E^{-9}
305x305	609	3.47E^{-9}
355x355	709	4.04E^{-9}



Conclusion # 1.10.1 – The total "RC" delay increases with the array size. This can be explained as the array size increases the total array area increases and the total number of junctions in the current path also increases thus increasing the total resistance.

Simulation # 1.11.1 – Comprehensive Simulation

Model: "Exp7_nanoarray_both.m"

Input Parameters (Assumed): D = 15E^{-7}, R_con = 1E^{+6},
C_con = 1E^{-18} F, R_jun = 6.66E^{+4} ohm,
C_jun = 1E^{-18} F,

ρ = 1 ohm-cm

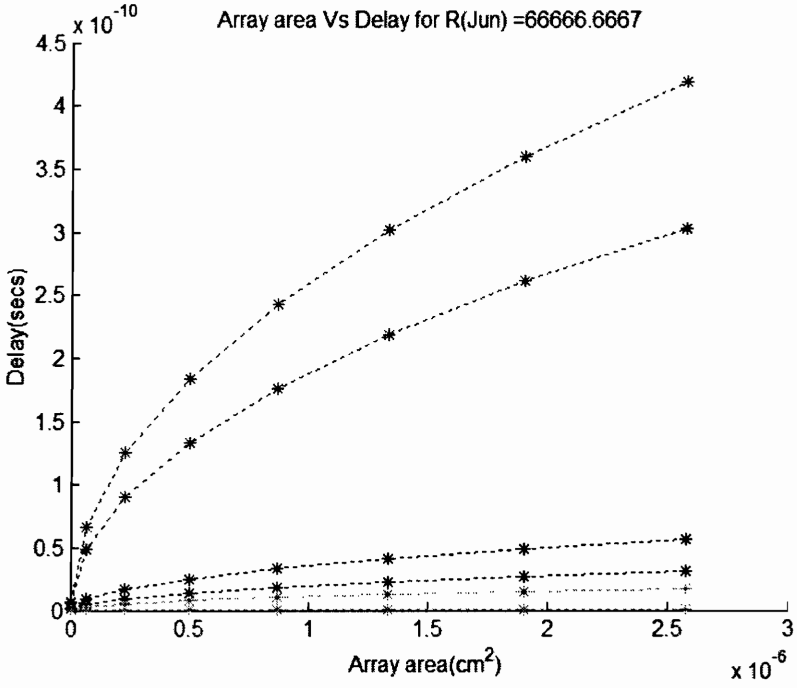
Array Size	Leiber Delay (secs)	Molecular Delay (secs)
5x5	5.97E^{-12}	7.28E^{-12}
55x55	4.84E^{-11}	6.60E^{-11}
105x105	9.08E^{-11}	1.25E^{-10}
155x155	1.33E^{-10}	1.83E^{-10}
205x205	1.75E^{-10}	2.42E^{-10}
255x255	2.18E^{-10}	3.01E^{-10}
305x305	2.60E^{-10}	3.60E^{-10}
355x355	3.03E^{-10}	4.19E^{-10}

$\rho = 0.1$ ohm-cm

Array Size	Leiber Delay (secs)	Molecular Delay (secs)
5x5	$1.21E^{-12}$	$1.50E^{-12}$
55x55	$5.46E^{-12}$	$9.40E^{-12}$
105x105	$9.70E^{-12}$	$1.72E^{-11}$
155x155	$1.40E^{-11}$	$2.51E^{-11}$
205x205	$1.81E^{-11}$	$3.30E^{-11}$
255x255	$2.24E^{-11}$	$4.09E^{-11}$
305x305	$2.66E^{-11}$	$4.88E^{-11}$
355x355	$3.09E^{-11}$	$5.66E^{-11}$

$\rho = 1E^{-3}$ ohm-cm

Array Size	Leiber Delay (secs)	Molecular Delay (secs)
5x5	$6.94E^{-13}$	$8.73E^{-13}$
55x55	$7.36E^{-13}$	$3.15E^{-12}$
105x105	$7.79E^{-13}$	$5.43E^{-12}$
155x155	$8.21E^{-13}$	$7.71E^{-12}$
205x205	$8.64E^{-13}$	$9.99E^{-12}$
255x255	$9.06E^{-13}$	$1.22E^{-11}$
305x305	$9.48E^{-13}$	$1.45E^{-11}$
355x355	$9.91E^{-13}$	$1.68E^{-11}$



Conclusion # 1.11.1 – The "Leiber" model delay is lower when compared to the "Molecular" model delay. The delay of both models decreases slightly with increasing values of silicon nanowire resistivity.

Simulation # 1.12.1 – Defects Analysis

Model: "Exp8_nanoarray.m"

Input Parameters(Assumed): D = 15E^{-7} cm, ρ = 10 ohm-cm

L_{end} = 50E^{-7} cm, NW_{pitch} = 30E^{-7} cm

Yield = 95%

Array Size	Redundancy	Original Crossbar Area (cm^2)	Total Crossbar Area (cm^2)
5x5	1	$8.7E^{-10}$	$1E^{-9}$
55x55	12	$6.4E^{-8}$	$6.8E^{-8}$
105x105	23	$2.3E^{-7}$	$2.4\ E^{-7}$
155x155	35	$5E^{-7}$	$5.2E^{-7}$
205x205	46	$8.6E^{-7}$	$9.1E^{-7}$
255x255	57	$1.3E^{-6}$	$1.4E^{-6}$

Yield = 90%

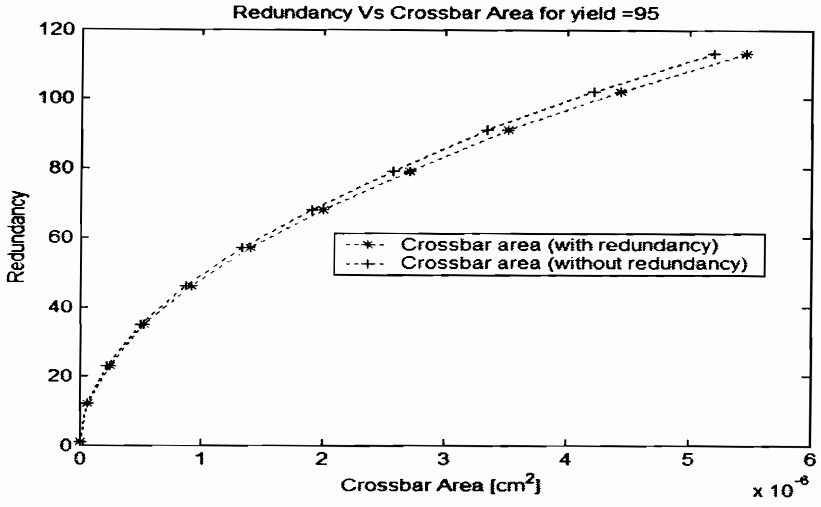
Array Size	Redundancy	Original Crossbar Area (cm^2)	Total Crossb Area (cm^2)
5x5	2	$8.7E^{-10}$	$1.2E^{-9}$
55x55	17	$6.5E^{-8}$	$7.2E^{-8}$
105x105	33	$2.3E^{-7}$	$2.5E^{-7}$
155x155	49	$5E^{-7}$	$5.5E^{-7}$
205x205	65	$8.6E^{-7}$	$9.5E^{-7}$
255x255	81	$1.3E^{-6}$	$1.5E^{-6}$

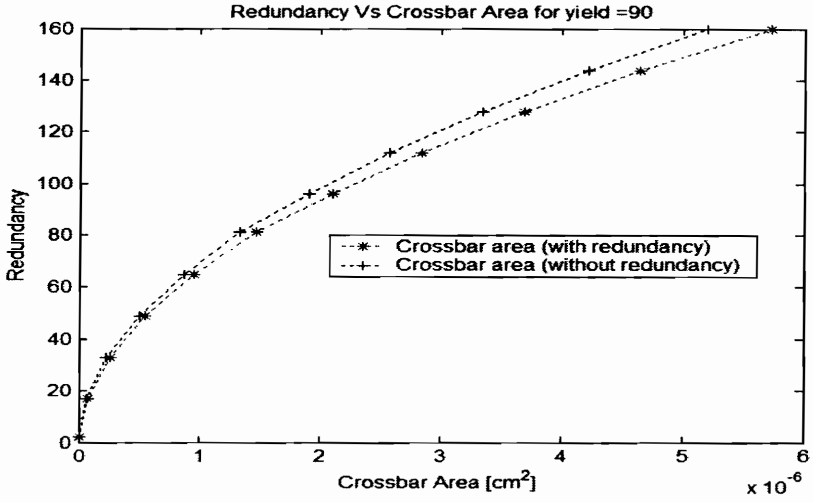
Yield = 85%

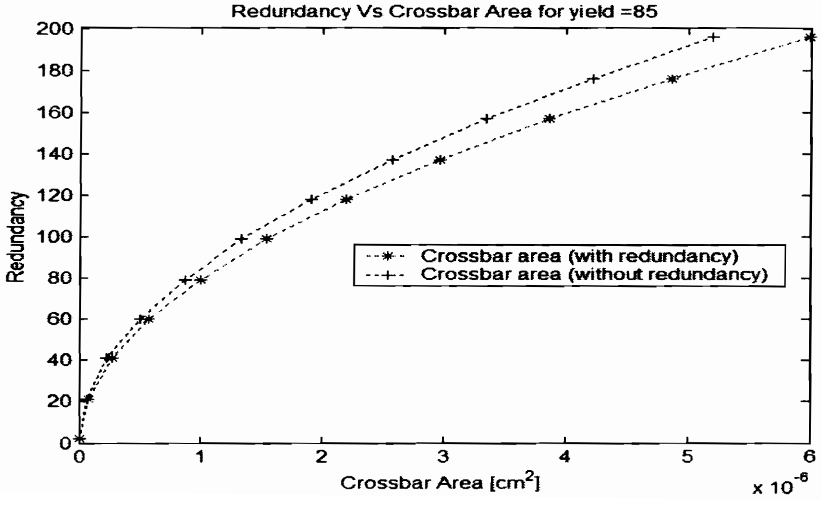
Array Size	Redundancy	Original Crossbar Area (cm^2)	Total Crossb Area (cm^2)
5x5	2	$8.7E^{-10}$	$1.2E^{-9}$
55x55	21	$6.5E^{-8}$	$7.5E^{-8}$
105x105	41	$2.3E^{-7}$	$2.7E^{-7}$
155x155	60	$5E^{-7}$	$5.7E^{-7}$
205x205	79	$8.6E^{-7}$	$10E^{-7}$
255x255	99	$1.3E^{-6}$	$1.5E^{-6}$
305x305	118	$1.9E^{-6}$	$2.2E^{-6}$
355x355	137	$2.6E^{-6}$	$3E^{-6}$
405x405	157	$3.3E^{-6}$	$3.8E^{-6}$
455x455	176	$4.2E^{-6}$	$4.9E^{-6}$
505x505	196	$5.2E^{-6}$	$6E^{-6}$

Yield = 80%

Array Size	Redundancy	Original Crossbar Area (cm^2)	Total Crossb Area (cm^2)
5x5	2	$8.7E^{-10}$	$1.2E^{-9}$
55x55	25	$6.5 E^{-8}$	$7.95E^{-8}$
105x105	47	$2.3E^{-7}$	$2.8E^{-7}$
155x155	69	$5E^{-7}$	$6E^{-7}$
205x205	92	$8.6E^{-7}$	$1E^{-6}$
255x255	114	$1.3E^{-6}$	$1.6E^{-6}$
305x305	136	$1.9E^{-6}$	$2.3E^{-6}$
355x355	159	$2.6E^{-6}$	$3.1E^{-6}$
405x405	181	$3.3E^{-6}$	$4E^{-6}$
455x455	203	$4.2E^{-6}$	$5E^{-6}$


Redundancy Vs Crossbar Area for yield =95
Redundancy
0
20
40
60
80
100
120
Crossbar Area [cm^2]
0
1
2
3
4
5
6
x 10^{-6}
Crossbar area (with redundancy)
Crossbar area (without redundancy)


Redundancy Vs Crossbar Area for yield =90
Redundancy
0
20
40
60
80
100
120
140
160
Crossbar Area [cm^2]
0
1
2
3
4
5
6
x 10^{-6}
Crossbar area (with redundancy)
Crossbar area (without redundancy)


Redundancy Vs Crossbar Area for yield =85
Redundancy
0
20
40
60
80
100
120
140
160
180
200
Crossbar Area [cm^2]
0
1
2
3
4
5
6
x 10^{-6}
Crossbar area (with redundancy)
Crossbar area (without redundancy)

Conclusion # 1.12.1 – From above simulation results we say that a crossbar array has better defects tolerance, for yield percentages between 80 to 95 and occupy relatively less area, for array sizes between "255x255" and "305x305".

Simulation # 1.13.1 – Memory Density

Model: "Exp9_nanoarray.m"

Input Parameters (Assumed) : D = $15E^{-7}$ cm, ρ = 10 ohm-cm

$L_{end} = 50E^{-7}$ cm, $NW_{pitch} = 30E^{-7}$ cm

Yield = 95%, Multiplexer area = 50x

```
ARRRAY SIZE
Number of Rows      250
Number of Columns    250
Length of the Row Nanowire(cm)        0.0011
New Length of the Row Nanowire(cm)    3.6250e-004
Total Area of the crossbar array(cm^2) 1.2814e-006
Total Area of the crossbar array with redundancy (cm^2)            1.4128e-006


Final Area of the crossbar array with redundancy (including multiplexer
area)(cm^2)  7.2054e-005
Number of Crossbar Junctions in 1cm^2 area  8.6740e+008
The number of "250x250" cells required for different memory capacities,
    cells_256  =  36571
    cells_512  =  73143
    cells_1024 = 146286
    cells_2048 = 292571
Total array area, including redundant array area and multiplexer area,
assuming multiplexer occupies 50 x of crossbar array area.
array_area_256 =  2.63  cm^2
array_area_512 =  5.27  cm^2
array_area_1024 = 10.54  cm^2
array_area_2048 = 21.10 cm^2
```

```
mem_density_256 =  8.6740e+008 bits/cm^2
mem_density_512 =  8.6740e+008 bits/cm^2
mem_density_1024 = 8.8540e+008 bits/cm^2
mem_density_2048 = 8.8540e+008 bits/cm^2
```

Conclusion # 1.13.1 – Memory density for a "250x250" array without redundancy (including multiplexer area) is 9.56E+008 and the memory density with redundancy(including multiplexer area) is 8.67E+8. The difference is approximated 8.9E+7 bits/cm^2. Thus redundancy causes a considerable amount of loss in memory density.

Conclusions – Based on our simulations we conclude the following

- The resistance of a typical nanowire in the range of Giga Ohms
- The optimal array size is "360x360", based on a typical length of the nanowire that can be fabricated.
- The optimal array size based on fan-in and fan-out constraints is "205x205".
- The total number of crossbar junctions per unit area becomes almost constant for array sizes greater than "360x360".
- Based on "RC" delay simulations "Leiber" model has less delay compared to the "Molecular" model.
- A crossbar array of size between "255x255" and "305x305" has better defect tolerance characteristics.
- The redundancy when introduced in crossbar array in order to have a better defect tolerance considerably reduces the memory density.

Chapter V

REFERENCES

[1] **Nanoscale molecular-switch crossbar circuits** - *Yong Chen (Quantum Sci. Res., Hewlett Packard Labs., Palo Alto, CA, USA;); Gun-Young Jung; Ohlberg, D.A.A.; Xuema Li; Stewart, D.R.; Jeppesen, J.O.; Nielsen, K.A.; Stoddart, J.F.; Williams, R.S.Nanotechnology, v 14, n 4, April 2003, p 462-8*

[2] **Nanowire crossbar arrays as address decoders for integrated nanosystems** - *ZhaoHui Zhong (Dept. of Chem. & Chem. Biol., Harvard Univ., Cambridge, MA, USA;); Deli Wang; Yi Cui; Bockrath, M.W.; Lieber, C.M.Science, v 302, n 5649, 21 Nov. 2003, p 1377-9*

[3] **The future of nanocomputing -** *Bourianoff, George Computer, v 36, n 8, August, 2003, p 44-53*

[4] **A case for CMOS/nano co-design** - *Ziegler, M.M. (Dept. of Electr. & Comput. Eng., Virginia Univ., Charlottesville, VA, USA); Stan, M.R. IEEE/ACM International Conference on Computer Aided Design. IEEE/ACM Digest of Technical Papers (Cat. No.02CH37391), 2002, p 348-52*

[5] **OHSU News -** *http://www.ohsu.edu/news/2004/022404nano.html*

[6] **Plenty of room, indeed** - *Roukes, M. Scientific American, v 285, n 3, September, 2001, p 42*

[7] **Electronically configurable molecular-based logic gates** - *Collier, C.P. (Dept. of Chem. & Biochem., California Univ., Los Angeles, CA, USA;); Wong, E.W.; Belohradsky, M.; Raymo, F.M.; Stoddart, J.F.; Kuekes, P.J.; Williams, R.S.; Heath, J.R. Science, v 285, n 5426, 16 July 1999, p 391-4*

[8] **Defect-tolerant Logic with Nanoscale Crossbar Circuits –** *Tad Hogg and Greg Snider, HP LABS May 25 2004.*

[9] **Carbon nanotube-based nonvolatile random access memory for molecular computing** - *Rueckes, T. (Dept. of Chem., Harvard Univ., Cambridge, MA, USA;); Kim, K.; Joselevich, E.; Tseng, G.Y.; Cheung, C.-L.; Lieber, C.M. Science, v 289, n 5476, 7 July 2000, p 94-7*

[10] **Logic gates and computation from assembled nanowire building blocks** - *Huang, Y. (Dept. of Chem., Harvard Univ., Cambridge, MA, USA;); Duan, X.; Cui, Y.; Lauhon, L.J.; Kim, K.-H.; Lieber, C.M. Science, v 294, n 5545, 9 Nov. 2001, p 1313-17*

[11] **CMOS/nano co-design for crossbar-based molecular electronic systems** - *Ziegler, M.M. (Dept. of Eng. & Comput. Electron., Univ. of Virginia, Charlottesville, VA, USA); Stan, M.R. IEEE Transactions on Nanotechnology, v 2, n 4, Dec. 2003, p 217-30*

[12] **A multilevel cache memory architecture for nanoelectronics** - *Crawley, D. (Dept. of Phys. & Astron., Univ. Coll. London, UK) Proceedings Ninth Great Lakes Symposium on VLSI, 1999, p 346-7*

[13] **Hybrid semiconductor-molecular nanoelectronics** - *Likharev, Konstantin (Department of Physics, State University of New York) Industrial Physicist, v 9, n 3, June/July, 2003, p 20-23*

[14] **Stochastic assembly of sublithographic nanoscale interfaces** - *DeHon, A. (Dept. of Comput. Sci., California Inst. of Technol., Pasadena, CA, USA;); Lincoln, P.; Savage, J.E. IEEE Transactions on Nanotechnology, v 2, n 3, Sept. 2003, p 165-74*

[15] **The CMOS/nano interface from a circuits perspective** - *Ziegler, M.M. (ECE Dept., Virginia Univ., Charlottesville, VA, USA); Stan, M.R. Proceedings of the 2003 IEEE International Symposium on Circuits and Systems (Cat. No.03CH37430), 2003, pt. 4, p IV-904-7 vol.4*

[16] **Design and analysis of crossbar circuits for molecular nanoelectronics** - *Ziegler, M.M. (ECE Dept., Virginia Univ., Charlottesville, VA, USA); Stan, M.R. Proceedings of the 2002 2nd IEEE Conference on Nanotechnology (Cat. No.02TH8630), 2002, p 323-7*

[17] **Array-based architecture for FET-based, nanoscale electronics** - *DeHon, A. (Dept. of Comput. Sci., California Inst. of Technol., Pasadena, CA, USA) IEEE Transactions on Nanotechnology, v 2, n 1, March 2003, p 23-32*

[18] **Decoding of stochastically assembled nanoarrays** - *Gojman, B. (Dept. of Comput. Sci., Brown Univ., Providence, RI, USA); Rachlin, E.; Savage, J.E. IEEE Computer Society Annual Symposium on VLSI, 2004, p 11-18*

[19] **A universal device model for nanoelectronic circuit simulation** - *Ziegler, M.M. (ECE Dept., Virginia Univ., Charlottesville, VA, USA); Rose, G.S.; Stan, M.R.*

Proceedings of the 2002 2nd IEEE Conference on Nanotechnology (Cat. No.02TH8630), 2002, p 83-8

[20] **Is nanoelectronics the future of microelectronics** – *Lundstrom, M. (Electr. & Comput. Eng., West Lafayette, IN, USA) ISLPED'02: Proceedings* ***of the*** *2002 International Symposium on Lower Power Electronics and Design (IEEE Cat. No.02TH8643), 2002, p 172-7*

[21] **Computing with Hysteretic Crossbar Arrays –** *G.Snider Applied Physics A 80, 2005, p 1165-1172*

[22] **Defect-tolerant demultiplexers for nano-electronics constructed from error-correctiong codes-** *Kuekes, P.J. ,Robinett, W., Seroussi, G., Stanley Williams R. Applied Physics A 80, 2005, 1161-1164*

[23] **Nanoelectronic Architecture -** *Kuekes, P.J. ,Snider, G., Hogg, T., Stanley Williams R. Applied Physics A 80, 2005, 1183-1195*

[24] **A novel interconnection technique for manufacturing nanowire devices –** *Saif Islam, M., Sharma, S., KAmins, T.I., Stanley Williams, R. Applied Physics A 80,2005, 1133-1140*

MATLAB CODE

Exp0_nanowire.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t PHASE # 1 - Simulation of a SILICON
Nanowire                         ')
R = 0;
A = 0;
D = 15e-7;
L = 15e-4;
K = 10;
R1_store = 0;
D1_store = 0;
A = ( pi * (D^2) ) / 4;              % Resistance Calculation %
R = (K * L) / A;
R1_store = R;
D1_store = D;
fprintf('\n\n\n\t\t\t\t  S I M U L A T I O N  #  1 - "R" of Nanowire
for a Nanowire "D" ')
fprintf('\n\n\t\tDiameter of the nanowire(cm)\n ')
disp(D1_store)
fprintf('\n\t\tResistance of the nanowire(Ohms)\n ')
disp(R1_store)
```

Exp1_nanowire.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t PHASE # 1 - Simulation of a SILICON
Nanowire                         ')
R = 0;
A = 0;
D = 0;
L = 10e-4;
K = 10;
R1_store(1) = 0;
D1_store(1) = 0;
count = 1;
% Resistance for Range of Diameter %
for D = 5e-7 : 2e-7: 40e-7
    A = ( pi * (D^2) ) / 4;
    R = (K * L) / A;
    R1_store(count) = R;
    D1_store(count) = D;
    count = count + 1;
end
fprintf('\n\n\n\t\t\t\t  S I M U L A T I O N  #  1 - "R" of Nanowire
for a range of Nanowire "D" ')
fprintf('\n\n\t\tDiameter of the nanowire(cm)\n ')
disp(D1_store)
```

```
fprintf('\n\t\tResistance of the nanowire(Ohms)\n ')
disp(R1_store)
plot(D1_store,R1_store,'r:*');
xlabel('D(cm)')
ylabel('R(Ohms)')
title(['D(SiNW) Vs R(SiNW) for Typical L = ',num2str(L),'cm']);
```

Exp1_1_nanowire.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t PHASE # 1 - Simulation of a SILICON
Nanowire                      ')
R1 = 0;
A1 = 0;
D1 = 10e-7;
L1 = 0;
K = 10;
R2_store(1) = 0;
L2_store(1) = 0;
count1 = 1;
% Resistance for Range of Length %
for L1 = 1e-4 : 2e-4: 20e-4;
    A1 = ( pi * (D1^2) ) / 4;
    R1 = (K * L1) / A1;
    R2_store(count1) = R1;
    L2_store(count1) = L1;
    count1 = count1 + 1;
end
fprintf('\n\n\t\t\t  S I M U L A T I O N  #  2 - "R" of Nanowire for
a range of Nanowire "L" ')
fprintf('\n\n\t\tLength of the nanowire(cm)\n\n')
disp(L2_store)
fprintf('\n\t\tResistance of the nanowire(Ohms)\n ')
disp(R2_store)
figure(1)
plot(L2_store,R2_store,'r:+')
xlabel('L(cm)')
ylabel('R(Ohms)')
title(['L(SiNW) Vs R(SiNW) for Typical D = ',num2str(D1),'cm']);
```

Exp2_nanowire.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t\t\tPHASE # 1 - Simulation of a SILICON
Nanowire\n')
R = 0;
A = 0;
D = 0;
L = 1e-4;
K = 10;
R1_store(1) = 0;
D1_store(1) = 0;
L1_store(1) = 0;
count = 1;
```

```
count1 = 1;
i = 1;
n = 1;
pt = 0;
s = 0;
% Resistance for various cominations of Length and Diameter
%
while L < 20e-4
 fprintf('\n\n Length of the Nanowire(cm):',L);
 disp(L)
 for D = 5e-7 : 4e-7: 25e-7;
    A = ( pi * (D^2) ) / 4;
    R = (K * L) / A;
    R1_store(count) = R;
    D1_store(count) = D;
    count = count + 1;
end
fprintf('\t\t Diameter of the Nanowire(cm)\n');
disp(D1_store)
fprintf('\t\t Resistance of the Nanowire(ohms)\n');
disp(R1_store)
if(i < 4)
figure(n)
subplot(2,2,i)
plot(D1_store,R1_store,':r*')
xlabel('D(cm)')
ylabel('R(Ohms)')
A = L/1e-7;
legend('L=',num2str(A),'nm')
i=i+1;
else
    figure(n)
    subplot(2,2,i)
    plot(D1_store,R1_store,':r*')
    xlabel('D(cm)')
    ylabel('R(Ohms)')
    A = L/1e-7;
    legend('L=',num2str(A),'nm')
    i = 1;
    n = n+1;
end
count = 1;
L1_store(count1) = L;
L = L + 5e-4;
count1 = count1 + 1;
R1_store = 0;
D1_store = 0;
End
```

Exp2_1_nanowire.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t\t\tPHASE # 1 - Simulation of a SILICON
Nanowire\n')
R = 0;
```

```
A = 0;
D = 1e-7;
L = 0;
K = 10;
R1_store(1) = 0;
D1_store(1) = 0;
L1_store(1) = 0;
count = 1;
count1 = 1;
i = 1;
n = 1;
% Resistance for various combinations of Length and
Diameter %
while D < 20e-7
 fprintf('\n\n Diameter of the Nanowire(cm):',L);
 disp(D)
 for L = 1e-4 : 4e-4: 20e-4;
    A = ( pi * (D^2) ) / 4;
    R = (K * L) / A;
    R1_store(count) = R;
    L1_store(count) = L;
    count = count + 1;
end
fprintf('\t\t Length of the Nanowire(cm)\n');
disp(L1_store)
fprintf('\t\t Resistance of the Nanowire(ohms)\n');
disp(R1_store)
if(i < 4)
    figure(n)
    subplot(2,2,i)
    plot(L1_store,R1_store,':r*')
    xlabel('L(cm)')
    ylabel('R(Ohms)')
    A = D/1e-7;
    title(['For D=',num2str(A),'nm'])
    i=i+1;
else
    figure(n)
    subplot(2,2,i)
    plot(L1_store,R1_store,':r*')
    xlabel('L(cm)')
    ylabel('R(Ohms)')
    A = D/1e-7;
    title(['For D=',num2str(A),'nm'])
    i = 1;
    n = n+1;
end
count = 1;
D1_store(count1) = D;
D = D + 5e-7;
count1 = count1 + 1;
R1_store = 0;
D1_store = 0;

End
```

Exp3_nanoarray.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t  Phase # 1 - "L" and "R" of the SiNW for
VARIOUS SIZED ARRAYS\n')
R_row = 0;
R_col = 0;
A = 0;
D = 15e-7;
L_row = 0;
L_col = 0;
K = 10;
no_row = 350;
no_col = 350;
% Length of Row and Column Nanowires %
L_col = (50e-7+(no_row * 15e-7)+((no_row-1) * 30e-7)+50e-7) + 15e-7;
L_row = (50e-7+(no_col * 15e-7)+((no_col-1) * 30e-7)+50e-7) + 15e-7;
A = ( pi * (D^2) )  / 4;
R_row = (K * L_row) / A;
R_col = (K * L_col) / A;
fprintf('\n\tARRRAY SIZE\n')
fprintf('\n\t\tNumber of Rows      ')
disp(no_row)
fprintf('\t\tNumber of Columns  ')
disp(no_col)
fprintf('\n\t\tLength of the Row Nanowire(Cm)    ')
disp(L_row)
fprintf('\n\t\tLength of the Column Nanowire(Cm)')
disp(L_col)
fprintf('\n\t\tResistance of the Row Nanowire(Ohms)    ')
disp(R_row)
fprintf('\n\t\tResistance of the Column Nanowire(Ohms)')
disp(R_col)
```

Exp3_1_nanoarray.m

```
clc
clear all
fprintf('\n\t\t\t\t\t\t\t Size of nanoarray based on fan-in & fan-out
constraints]\n')
R_jun = 0;
R_con = 1e6;
R_tot = 0;
R_wire = 0;
A = 0;
A_jun = 15e-7 * 15e-7;
L_jun = 15e-7;
D = 15e-7;
K1 = 10;
K2 = 0.1;
% Assumed Input Current and Voltage %
I = 100e-12;
V1 = 2;
count = 1;
L_wire = 0;
```

```
no_row = 5;
no_col = 5;
I_in = 1;
I_out = no_row;
while(no_row <= 400)
    track = 0;
    display = 0;
    no_juns = 0;
    x = 1; y = I_in;
    while(x < I_out)
        if(no_row <=8)
            track(no_row,no_col) = 0;
            track(x,y) = 1;
        end
        x = x+1;
        no_juns = no_juns +1;
    end
    while(y <= no_col)
        if(no_row <=8)
            track(x,y) = 1;
            display = 1;
        end
        y = y +1;
        no_juns = no_juns +1;
    end
% Length of Nanowire in Current Path %
L_wire = (50e-7+((no_juns-1) * 30e-7)+50e-7);
A = ( pi * (D^2) ) / 4;
% Resistance of Nanowire in Current Path %
R_wire = (K1 * L_wire) / A;
R_jun =  (K2 * L_jun) / A_jun;
% Total Resistance and Volatage Drop Calculations %
R_tot = (R_con * 2) + (R_jun * no_juns) + R_wire;
V2 = I * R_tot;
V_drop = V1 - V2;
V_drop_store(count) = V_drop;
array_size_store(count) = no_row;
fprintf('\n\tARRRAY SIZE\n')
fprintf('\n\t\tNumber of Rows      ')
disp(no_row)
fprintf('\t\tNumber of Columns  ')
disp(no_col)
fprintf('\tCURRENT FLOW')
fprintf('\n\t\tCurrent entering col')
disp(I_in)
fprintf('\t\tCurrent leaving row   ')
disp(I_out)
fprintf('\t\tNumber of crossbar junctions for current flow:')
disp(no_juns)
fprintf('\t\tTotal Resistance(Ohms):')
disp(R_tot)
fprintf('\t\tVoltage (V):')
disp(V1)
fprintf('\t\tCurrent (Amps):')
disp(I)
fprintf('\t\tVoltage Drop (V)')
```

```
disp(V_drop)

if(display == 1)
    fprintf('\t\tCURRENT FLOW MATRIX FOR THIS CASE:\n\n')
    disp(track)
else
    fprintf('\t\tCURRENT FLOW MATRIX FOR THIS CASE WAS OMITTED TO AVOID
COMPLEXITY\n\n')
end
no_row = no_row + 50;
no_col = no_col + 50;
I_in = 1;
I_out = no_row;
count = count + 1;
end
plot(array_size_store,V_drop_store,'r:*')
xlabel('Array area(cm^2)')
ylabel('Currnet(Amps)')
title(['Array area Vs Current'])
```

Exp4_nanoarray.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t\t\tNumber of the Crossbar junctions in
a given area \n')
R_row = 0;
R_col = 0;
L_row = 0;
L_col = 0;
no_row = 3;
no_col = 3;
no_cbjuns = 0;
% Length of Column and Row Nanowires %
L_col = (50e-7+(no_row * 15e-7)+((no_row-1) * 30e-7)+50e-7) + 15e-7;
L_row = (50e-7+(no_col * 15e-7)+((no_col-1) * 30e-7)+50e-7) + 15e-7;
no_juns = no_row * no_col;
crossbar_area = L_row * L_col;
% Number of Crossbar Junctions per unit area %
no_cbjuns = no_juns/crossbar_area;
fprintf('\n\tARRRAY SIZE\n')
fprintf('\n\t\tNumber of Rows     ')
disp(no_row)
fprintf('\t\tNumber of Columns  ')
disp(no_col)
fprintf('\n\t\tLength of the Row Nanowire(cm)         ')
disp(L_row)
fprintf('\n\t\tLength of theColumn Nanowire(cm)       ')
disp(L_col)
fprintf('\n\t\tTotal Area of the crossbar array(cm^2) ')
disp(crossbar_area)
fprintf('\n\t\tNumber of Crossbar Junctions in 1cm^2 area')
disp(no_cbjuns)
```

Exp5_nanoarray.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t  Phase # 1 - "L" and "R" of the SiNW for
VARIOUS SIZED ARRAYS\n')
R_row = 0;
R_col = 0;
A = 0;
D = 15e-7;
K = 10;
L_row = 0;
L_col = 0;
Lrow_store = 0;
Lcol_store = 0;
Rrow_store = 0;
Rcol_store = 0;
CBjuns_store = 0;
CBarea_store = 0;
count = 1;
no_cbjuns = 0;
i = 1;
j = 1;
no_row = 5;
no_col = 5;
% Length and Crossbar junctions per unit area for various
array sizes %
while (no_row <= 560 & no_col<560)
    L_col = (50e-7+(no_row * 15e-7)+((no_row-1) * 30e-7)+50e-7);
    L_row = (50e-7+(no_col * 15e-7)+((no_col-1) * 30e-7)+50e-7);
    A = ( pi * (D^2) ) / 4;
    R_row = (K * L_row) / A;
    R_col = (K * L_col) / A;
    no_juns = no_row * no_col;
    crossbar_area = L_row * L_col;
    no_cbjuns = no_juns/crossbar_area;
    Lrow_store(count) = L_row;
    Rrow_store(count) = R_row;
    Lcol_store(count) = L_col;
    Rcol_store(count) = R_col;
    CBjuns_store(count) = no_cbjuns;
    CBarea_store(count) = crossbar_area;
    fprintf('\n\tARRRAY SIZE\n')
    fprintf('\n\t\tNumber of Rows      ')
    disp(no_row)
    fprintf('\t\tNumber of Columns  ')
    disp(no_col)
    fprintf('\n\t\tLength of the Row Nanowire(Cm)    ')
    disp(L_row)
    fprintf('\n\t\tLength of the Column Nanowire(Cm)')
    disp(L_col)
    fprintf('\n\t\tResistance of the Row Nanowire(Ohms)    ')
    disp(R_row)
    fprintf('\n\t\tResistance of the Column Nanowire(Ohms)')
    disp(R_col)
    fprintf('\n\t\tTotal Area of the crossbar array(cm^2) ')
    disp(crossbar_area)
    fprintf('\n\t\tNumber of Crossbar Junctions in 1cm^2 area')
    disp(no_cbjuns)
    no_row = no_row + 50;
    no_col = no_col + 50;
    i = i+1;
    j = j+1;
    count = count + 1;
end
figure(1)
plot(Lrow_store,Rrow_store,'r:+')
xlabel('L [row](cm)')
ylabel('R [row](Ohms)')
title(['L(SiNW) Vs R(SiNW)']);
figure(2)
plot(Lcol_store,Rcol_store,'r:+')
xlabel('L [col](cm)')
ylabel('R [col](Ohms)')
title(['L(SiNW) Vs R(SiNW)']);
figure(3)
plot(Lrow_store,CBarea_store,'r:+')
xlabel('L (cm)')
ylabel('Array area(cm^2)')
figure(4)
plot(CBarea_store,CBjuns_store)
xlabel('Crossbar Area(cm^2)')
ylabel('No of CBjunctions')
```

Exp6_nanoarray_leiber.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t\t\t  Phase # 1 - RC Delay Calculation -
LEIBER MODEL \n')
R_row = 0;
R_col = 0;
R_jun = 0;
C_jun = 1e-18;
R_con = 1e6;
C_con = 1e-18;
R_tot = 0;
C_tot = 0;
R_wire = 0;
R_tot_store = 0;
C_tot_store = 0;
delay_store = 0;
A = 0;
A_jun = 15e-7 * 15e-7;
L_jun = 15e-7;
D = 15e-7;
K1 = 10;
K2 = 0.1;
count = 1;
L_wire = 0;
no_row = 5;
```

```
no_col = 5;
while(no_row <= 400)
    L_col = (50e-7+(no_row * 15e-7)+((no_row-1) * 30e-7)+50e-7);
    L_row = (50e-7+(no_col * 15e-7)+((no_col-1) * 30e-7)+50e-7);
    A = ( pi * (D^2) ) / 4;
    R_row = (K1 * L_row) / A;
    R_col = (K1 * L_col) / A;
    no_juns = no_row * no_col;
    crossbar_area = L_row * L_col;
    no_cbjuns = no_juns/crossbar_area;
    Lrow_store(count) = L_row;
    Rrow_store(count) = R_row;
    Lcol_store(count) = L_col;
    Rcol_store(count) = R_col;
    CBjuns_store(count) = no_cbjuns;
    CBarea_store(count) = crossbar_area;
   L_wire = (50e-7+((no_row-1) * 15e-7)+((no_row-1) * 30e-7)+50e-7);
% Length of Wire in Current Path %
    R_wire = (K1 * L_wire) / A;
% Resistance of Wire in Current Path %
    R_jun =  (K2 * L_jun)  / A_jun;
    R_tot = (R_con * 2) + R_jun + R_wire;
% Total Resistance in current path %
    C_tot = (C_con * C_jun) / ((2 * C_jun) + (C_con));
% Total Capacitance in current path  %
    delay = R_tot * C_tot;
% Delay Calculation %
    R_tot_store(count) = R_tot;
    C_tot_store(count) = C_tot;
    delay_store(count) = delay;
    fprintf('\n\tARRRAY SIZE\n')
    fprintf('\n\t\tNumber of Rows     ')
    disp(no_row)
    fprintf('\t\tNumber of Columns  ')
    disp(no_col)
    fprintf('\t\tLength of the Row Nanowire(Cm)   ')
    disp(L_row)
    fprintf('\t\tLength of the Column Nanowire(Cm)')
    disp(L_col)
    fprintf('\t\tResistance of the Row Nanowire(Ohms)   ')
    disp(R_row)
    fprintf('\t\tResistance of the Column Nanowire(Ohms)')
    disp(R_col)
    fprintf('\t\tTotal Area of the crossbar array(cm^2) ')
    disp(crossbar_area)
    fprintf('\t\tNumber of Crossbar Junctions in 1cm^2 area')
    disp(no_cbjuns)
    fprintf('\t\tTotal Resistance(Ohms):')
    disp(R_tot)
    fprintf('\t\tTotal Capacitance(F):')
    disp(C_tot)
    fprintf('\t\tRC Delay [WORST CASE] (secs):')
    disp(delay)
    no_row = no_row + 50;
    no_col = no_col + 50;
    I_in = 1;
    I_out = no_row;
    count = count + 1;
end
figure(1)
plot(Lrow_store,delay_store,'r:+')
xlabel('L(NW) [cm]')
ylabel('RC Delay [secs]')
title(['L(NW) Vs Delay'])
figure(2)
plot(CBarea_store,log(delay_store),'r:*')
xlabel('Array area(cm^2)')
ylabel('Delay[LOG SCALE]')
title(['Array area Vs Delay'])
figure(3)
plot(CBarea_store,CBjuns_store,':r*')
xlabel('Array area(cm^2)')
ylabel('CB Juns per cm^2')
title(['No CBjuns Vs Delay'])
figure(4)
plot(CBarea_store,delay_store,'r:*')
xlabel('Array area(cm^2)')
ylabel('Delay(secs)')
title(['Array area Vs Delay'])
disp(R_jun)
disp(L_wire)
disp(R_wire)
```

Exp6_nanoarray.molecular.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t\t\t  Phase # 1 - RC Delay Calculation -
MOLECULAR MODEL \n')
R_row = 0;
R_col = 0;
R_jun = 0;
C_jun = 1e-18;
R_con = 1e6;
C_con = 1e-18;
R_tot = 0;
C_tot = 0;
R_wire = 0;
R_tot_store = 0;
C_tot_store = 0;
delay_store = 0;
A = 0;
A_jun = 15e-7 * 15e-7;
L_jun = 15e-7;
D = 15e-7;
K1 = 10;
K2 = 0.1;
count = 1;
L_wire = 0;
no_row = 5;
no_col = 5;
I_in = 1;
```

```
I_out = no_row;
while(no_row <= 400)
    L_col = (50e-7+(no_row * 15e-7)+((no_row-1) * 30e-7)+50e-7);
    L_row = (50e-7+(no_col * 15e-7)+((no_col-1) * 30e-7)+50e-7);
    A = ( pi * (D^2) ) / 4;
    R_row = (K1 * L_row) / A;
    R_col = (K1 * L_col) / A;
    no_juns = no_row * no_col;
    crossbar_area = L_row * L_col;
    no_cbjuns = no_juns/crossbar_area;
    Lrow_store(count) = L_row;
    Rrow_store(count) = R_row;
    Lcol_store(count) = L_col;
    Rcol_store(count) = R_col;
    CBjuns_store(count) = no_cbjuns;
    CBarea_store(count) = crossbar_area;
    track = 0;
    display = 0;
    no_juns = 0;
    x = 1; y = I_in;
    while(x < I_out)
        if(no_row <=8)
            track(no_row,no_col) = 0;
            track(x,y) = 1;
        end
        x = x+1;
        no_juns = no_juns +1;
    end
    while(y <= no_col)
        if(no_row <=8)
            track(x,y) = 1;
            display = 1;
        end
        y = y +1;
        no_juns = no_juns +1;
    end
% Length of nanowire in current path %
L_wire = (50e-7+((no_juns-1) * 30e-7)+50e-7);
% Resistance of nanowire in current path%
R_wire = (K1 * L_wire) / A;                                     R_jun =
(K2 * L_jun) / A_jun;
% Total Resistance in current path %
R_tot = (R_con * 2) + (R_jun * no_juns) + R_wire;
% Total Capacitance in current path %
C_tot = (C_con * C_jun) / ((2 * C_jun) + (C_con));
% Delay Calculation %
delay = R_tot * C_tot;
R_tot_store(count) = R_tot;
C_tot_store(count) = C_tot;
delay_store(count) = delay;
fprintf('\n\tARRRAY SIZE\n')
fprintf('\n\t\tNumber of Rows      ')
disp(no_row)
fprintf('\t\tNumber of Columns  ')
disp(no_col)
fprintf('\tCURRENT FLOW')
fprintf('\n\t\tCurrent entering col')
disp(I_in)
fprintf('\t\tCurrent leaving row   ')
disp(I_out)
fprintf('\t\tLength of the Row Nanowire(Cm)   ')
disp(L_row)
fprintf('\t\tLength of the Column Nanowire(Cm)')
disp(L_col)
fprintf('\t\tResistance of the Row Nanowire(Ohms)   ')
disp(R_row)
fprintf('\t\tResistance of the Column Nanowire(Ohms)')
disp(R_col)
fprintf('\t\tTotal Area of the crossbar array(cm^2) ')
disp(crossbar_area)
fprintf('\t\tNumber of Crossbar Junctions in 1cm^2 area')
disp(no_cbjuns)
fprintf('\t\tNumber of crossbar junctions for current flow:')
disp(no_juns)
fprintf('\t\tTotal Resistance(Ohms):')
disp(R_tot)
fprintf('\t\tTotal Capacitance(F):')
disp(C_tot)
fprintf('\t\tRC Delay [WORST CASE] (secs):')
disp(delay)

if(display == 1)
    fprintf('\t\tCURRENT FLOW MATRIX FOR THIS CASE:\n\n')
    disp(track)
else
    fprintf('\t\tCURRENT FLOW MATRIX FOR THIS CASE WAS OMITTED TO AVOID
COMPLEXITY\n\n')
end
no_row = no_row + 50;
no_col = no_col + 50;
I_in = 1;
I_out = no_row;
count = count + 1;
end
figure(1)
plot(Lrow_store,delay_store,'r:+')
xlabel('L(NW) [cm]')
ylabel('RC Delay [secs]')
title(['L(NW) Vs Delay'])
figure(2)
plot(CBarea_store,log(delay_store),'r:*')
xlabel('Array area(cm^2)')
ylabel('Delay[LOG SCALE]')
title(['Array area Vs Delay'])
figure(3)
plot(CBarea_store,CBjuns_store,':r*')
xlabel('Array area(cm^2)')
ylabel('CB Juns per cm^2')
title(['No CBjuns Vs Delay'])
figure(4)
plot(CBarea_store,delay_store,'r:*')
xlabel('Array area(cm^2)')
ylabel('Delay(secs)')
```

```
title(['Array area Vs Delay'])
```

Exp7_nanoarray_both.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t\t\t  Phase # 1 - RC Delay Calculation -
LEIBER & MOLECULAR MODEL \n')
fprintf('\n\t\t\t\t\t\t CASE # 1 Delay for current flowing from column
nanowire to row nanowire[Exits in the right]\n')
i=1;
R_row = 0;
R_col = 0;
R_jun = 0;
C_jun = 1e-18;
R_con = 1e6;
C_con = 1e-18;
R_tot_mol = 0;
C_tot_mol = 0;
R_tot_lei = 0;
C_tot_lei = 0;
R_wire_lei = 0;
R_wire_mol = 0;
delay_lei = 0;
delay_mol = 0;
R_tot_store_lei = 0;
R_tot_store_mol = 0;
C_tot_store_mol = 0;
C_tot_store_lei = 0;
delay_store_mol = 0;
delay_store_lei = 0;
A = 0;
A_jun = 15e-7 * 15e-7;
L_jun = 15e-7;
D = 15e-7;
K1 = 1e-3;
K2 = 0.1;
count = 1;
L_wire_lei = 0;
L_wire_mol = 0;
no_row = 5;
no_col = 5;
I_in = 1;
I_out = no_row;
fprintf('\n\tRESISTANCE OF THE CROSSBAR JUNCTION(Ohms)')
disp(R_jun)

while(no_row <= 400)
    L_col = (50e-7+(no_row * 15e-7)+((no_row-1) * 30e-7)+50e-7);
    L_row = (50e-7+(no_col * 15e-7)+((no_col-1) * 30e-7)+50e-7);
    A = ( pi * (D^2) ) / 4;
    R_row = (K1 * L_row) / A;
    R_col = (K1 * L_col) / A;
    no_juns = no_row * no_col;
    crossbar_area = L_row * L_col;
    no_cbjuns = no_juns/crossbar_area;
    Lrow_store(count) = L_row;
    Rrow_store(count) = R_row;
    Lcol_store(count) = L_col;
    Rcol_store(count) = R_col;
    CBjuns_store(count) = no_cbjuns;
    CBarea_store(count) = crossbar_area;
    track = 0;
    display = 0;
    no_juns = 0;
    x = 1; y = I_in;
    while(x < I_out)
        if(no_row <=8)
            track(no_row,no_col) = 0;
            track(x,y) = 1;
        end
        x = x+1;
        no_juns = no_juns +1;
    end
    while(y <= no_col)
        if(no_row <=8)
            track(x,y) = 1;
            display = 1;
        end
        y = y +1;
        no_juns = no_juns +1;
    end
% Molecular Model %
L_wire_mol = (50e-7+((no_juns-1) * 30e-7)+50e-7);
% Leiber Model %
L_wire_lei = (50e-7+((no_row-1) * 15e-7)+((no_row-1) * 30e-7)+50e-7);

R_wire_lei = (K1 * L_wire_lei) / A;
R_wire_mol = (K1 * L_wire_mol) / A;

R_jun =  (K2 * L_jun) / A_jun;

R_tot_mol = (R_con * 2) + (R_jun * no_juns) + R_wire_mol;
R_tot_lei = (R_con * 2) + R_jun + R_wire_lei;

C_tot_mol = (C_con * C_jun) / ((2 * C_jun) + (C_con));
C_tot_lei = (C_con * C_jun) / ((2 * C_jun) + (C_con));

delay_mol = R_tot_mol * C_tot_mol;
% Molecular Model Delay %
delay_lei = R_tot_lei * C_tot_lei;
% Leiber Model Delay %

R_tot_store_lei(count) = R_tot_lei;
C_tot_store_lei(count) = C_tot_lei;

R_tot_store_mol(count) = R_tot_mol;
C_tot_store_mol(count) = C_tot_mol;

delay_store_mol(count) = delay_mol;
delay_store_lei(count) = delay_lei;
```

```
fprintf('\tARRRAY SIZE\n')
fprintf('\n\t\tNumber of Rows     ')
disp(no_row)
fprintf('\t\tNumber of Columns  ')
disp(no_col)
fprintf('\n\tARRRAY DESCRIPTION\n\n')
fprintf('\t\tLength of the Row Nanowire(Cm)   ')
disp(L_row)
fprintf('\t\tLength of the Column Nanowire(Cm)')
disp(L_col)
fprintf('\t\tResistance of the Row Nanowire(Ohms)   ')
disp(R_row)
fprintf('\t\tResistance of the Column Nanowire(Ohms)')
disp(R_col)
fprintf('\t\tTotal Area of the crossbar array(cm^2) ')
disp(crossbar_area)
fprintf('\t\tNumber of Crossbar Junctions in 1cm^2 area')
disp(no_cbjuns)
fprintf('\tCURRENT FLOW\n')
fprintf('\n\t\tCurrent entering col')
disp(I_in)
fprintf('\t\tCurrent leaving row   ')
disp(I_out)
fprintf('\t\tNumber of crossbar junctions for current flow(for Mol):')
disp(no_juns)
fprintf('\tDELAY CALCULATION\n\n')
fprintf('\t\tTotal Resistance(Ohms)-Leiber:')
disp(R_tot_lei)
fprintf('\t\tTotal Resistance(Ohms)-Molecular:')
disp(R_tot_mol)
fprintf('\t\tTotal Capacitance(F)[Molecular]:')
disp(C_tot_mol)
fprintf('\t\tRC Delay (secs)[Molecular]:     ')
disp(delay_mol)
fprintf('\t\tTotal Capacitance(F)[Leiber]:   ')
disp(C_tot_lei)
fprintf('\t\tRC Delay (secs)[Leiber]:        ')
disp(delay_lei)
if(display == 1)
    fprintf('\t\tCURRENT FLOW MATRIX FOR THIS CASE:\n\n')
    disp(track)
else
    fprintf('\t\tCURRENT FLOW MATRIX FOR THIS CASE WAS OMITTED TO AVOID
COMPLEXITY\n\n')
end
no_row = no_row + 50;
no_col = no_col + 50;
I_in = 1;
I_out = no_row;
count = count + 1;
end
figure(i)
plot(CBarea_store,Lrow_store,'r:*')
xlabel('Crossbar Area [cm^2]')
ylabel('L(NW) (cm)')
title(['CB Area Vs L(NW) for R(Jun) =',num2str(R_jun)])
i = i+1;
```

```
figure(i)
plot(Lrow_store,Rrow_store,'r:*')
xlabel('L(NW) [cm]')
ylabel('R(Ohms)')
title(['L(NW) Vs R for R(Jun) =',num2str(R_jun)])
i = i+1;
figure(i)
plot(CBarea_store,CBjuns_store,'r:*')
xlabel('Crossbar area (cm^2)')
ylabel('Crossbar juns')
title(['CB Area Vs No of Juns for R(Jun) =',num2str(R_jun)])
i = i+1;
figure(i)
subplot(2,1,1)
plot(Lrow_store,delay_store_mol,'r:+')
xlabel('L(NW) [cm]')
ylabel('RC Delay [secs]')
title(['L(NW) Vs Delay(Molecular) for R(Jun) =',num2str(R_jun)])
subplot(2,1,2)
plot(Lrow_store,delay_store_lei,'r:+')
xlabel('L(NW) [cm]')
ylabel('RC Delay [secs]')
title(['L(NW) Vs Delay(Leiber) for R(Jun) =',num2str(R_jun)])
i = i +1;
figure(i)
subplot(2,1,1)
plot(CBarea_store,delay_store_mol,'r:*')
xlabel('Array area(cm^2)')
ylabel('Delay(secs)')
title(['Array area Vs Delay(Molecular) for R(Jun) =',num2str(R_jun)])
subplot(2,1,2)
plot(CBarea_store,delay_store_lei,'r:*')
xlabel('Array area(cm^2)')
ylabel('Delay(secs)')
title(['Array area Vs Delay(Leiber) for R(Jun) =',num2str(R_jun)])
i = i + 1;
figure(i)
subplot(2,1,1)
semilogy(CBarea_store,delay_store_mol,'r:*')
xlabel('Array area(cm^2)')
ylabel('Delay[LOG SCALE]')
title(['Array area Vs Delay(Molecular) for R(Jun)=',num2str(R_jun)])
subplot(2,1,2)
semilogy(CBarea_store,delay_store_lei,'r:*')
xlabel('Array area(cm^2)')
ylabel('Delay[LOG SCALE]')
title(['Array area Vs Delay(Leiber) for R(Jun) =',num2str(R_jun)])
```

Exp8_nanoarray.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t\t\t\tDefects Tolerance based on Yield
Percentage and Redundancy \n')
yield = 80;
i = 1;
```

```
while (yield < 100)
    R_row = 0;
    R_col = 0;
    L_row = 0;
    L_col = 0;
    no_row = 5;
    no_col = 5;
    no_cbjuns = 0;
    new_L_row = 0;
    redundant_col = 0;
    new_no_row = 0;
    new_no_col = 0;
    total_crossbar_area = 0;
    count = 1;
    while(no_row < 550)
        % Total Junctions %
        no_juns = no_row * no_col;
        % Nonfunctional junctions %
        total_nonfunc_juns = ((100 - yield) / 100) * no_juns;
        nonfunc_juns = round(total_nonfunc_juns);
        % Redundancy Calculation %
        redundancy = sqrt(nonfunc_juns);
        redundant_row = round(redundancy);
        redundant_col = round(redundancy);
        if (redundant_row > 0)
            % Redundant rows %
            new_no_row = redundant_row;
            % Redundant Columns %
            new_no_col = redundant_col;
        end;
        L_col = (50e-7+(no_row * 15e-7)+((no_row-1) * 30e-7)+50e-7);
        L_row = (50e-7+(no_col * 15e-7)+((no_col-1) * 30e-7)+50e-7);
        crossbar_area = L_row * L_col;
        crossbar_area_store(count) = crossbar_area;
        if (new_no_row > 0)
            new_L_col = (50e-7+(new_no_row * 15e-7)+((new_no_row-1) *
30e-7)+50e-7) + 15e-7;
            new_L_row = (50e-7+(new_no_col * 15e-7)+((new_no_col-1) *
30e-7)+50e-7) + 15e-7;
            new_crossbar_area = new_L_row * new_L_col;
% Total Crossbar area including redundancy %
            total_crossbar_area = new_crossbar_area + crossbar_area;
new_crossbar_area_store(count) = new_crossbar_area;
            total_crossbar_area_store(count) = total_crossbar_area;
            redundancy_store(count) = new_no_row;
        end;
        diff_area = total_crossbar_area - crossbar_area;
        diff_area_store(count) = diff_area;
        no_cbjuns = no_juns/crossbar_area;
        array_size_store(count) = no_row;
        fprintf('\n\tARRRAY SIZE\n')
        fprintf('\n\t\tNumber of Rows     ')
        disp(no_row)
        fprintf('\t\tNumber of Columns  ')
        disp(no_col)
        fprintf('\n\t\tLength of the Row Nanowire(cm)         ')
        disp(L_row)
        fprintf('\n\t\tNew Length of the Row Nanowire(cm)         ')
        disp(new_L_row)
        fprintf('\n\t\tLength of theColumn Nanowire(cm)       ')
        disp(L_col)
        fprintf('\n\t\tTotal Area of the crossbar array(cm^2) ')
        disp(crossbar_area)
        fprintf('\n\t\tTotal Area of the crossbar array with
redundancy(cm^2) ')
        disp(total_crossbar_area)
        fprintf('\n\t\tRedundancy')
        disp(redundancy_store(count))
        fprintf('\n\t\tNumber of Crossbar Junctions in 1cm^2 area')
        disp(no_cbjuns)
        no_row = no_row + 50;
        no_col = no_col + 50;
        count = count + 1;
        disp(redundant_col)
        disp(new_no_col)
        disp(diff_area)
        disp(yield)
    end;
    figure(i)
    plot(total_crossbar_area_store,redundancy_store,'r:*')
    ylabel('Redundancy ')
    xlabel('Crossbar Area [cm^2]')
    title(['Redundancy Vs Crossbar Area for yield=',num2str(yield)])
    hold on
    plot(crossbar_area_store,redundancy_store,'b:+')
    ylabel('Redundancy ')
    xlabel('Crossbar Area [cm^2]')
    title(['Redundancy Vs Crossbar Area for yield=',num2str(yield)])
    legend('Crossbar area (with redundancy)','Crossbar area (without
redundancy)')
    hold off;
    i = i + 1;
    yield = yield + 5;
end;
```

Exp9_nanoarray.m

```
clc
clear all
fprintf('\n\n\t\t\t\t\t\t\t\t\t\t\tNano Memory Density comaparable to
todays memory capacity  \n')
yield = 90;
R_row = 0;
R_col = 0;
L_row = 0;
L_col = 0;
no_row = 250;
no_col = 250;
no_cbjuns = 0;
new_L_row = 0;
redundant_col = 0;
new_no_row = 0;
```

```
new_no_col = 0;
mem_capacity = 7e-3;
cells_256 = round(256/mem_capacity)
cells_512 = round(512/mem_capacity)
cells_1024 = round(1024/mem_capacity)
cells_2048 = round(2048/mem_capacity)
total_crossbar_area = 0;
% Total Junctions %
no_juns = no_row * no_col;
% Nonfunctional junctions %
total_nonfunc_juns = ((100 - yield) / 100) * no_juns;
% Redundancy Calculation %
redundancy = sqrt(total_nonfunc_juns);                   redundant_row =
round(redundancy);
redundant_col = round(redundancy);
% Redundant rows %
new_no_row = redundant_row;
% Redundant Columns %
new_no_col = redundant_col;
L_col = (50e-7+(no_row * 15e-7)+((no_row-1) * 30e-7)+50e-7);
L_row = (50e-7+(no_col * 15e-7)+((no_col-1) * 30e-7)+50e-7);
new_L_col = (50e-7+(new_no_row * 15e-7)+((new_no_row-1) * 30e-7)+50e-
7);
new_L_row = (50e-7+(new_no_col * 15e-7)+((new_no_col-1) * 30e-7)+50e-
7);
crossbar_area = L_row * L_col;
new_crossbar_area = new_L_row * new_L_col;
total_crossbar_area = new_crossbar_area + crossbar_area;    % Total
Crossbar area including redundancy %
multiplexer_area = 50 * total_crossbar_area;
final_crossbar_area = multiplexer_area + total_crossbar_area;
no_cbjuns = no_juns/final_crossbar_area;
% Area occupied by all the cells %
array_area_256 = cells_256 * final_crossbar_area
array_area_512 = cells_512 * final_crossbar_area
array_area_1024 = cells_1024 * final_crossbar_area
array_area_2048 = cells_2048 * final_crossbar_area
% Memory Density %
mem_density_256 = (250*250*cells_256) / array_area_256
mem_density_512 = (250*250*cells_512) / array_area_512
mem_density_1024 = (250*250*cells_1024) / array_area_1024
mem_density_2048 = (250*250*cells_2048) / array_area_2048
fprintf('\n\tARRRAY SIZE\n')
fprintf('\n\t\tNumber of Rows     ')
disp(no_row)
fprintf('\t\tNumber of Columns  ')
disp(no_col)
fprintf('\n\t\tLength of the Row Nanowire(cm)         ')
disp(L_row)
fprintf('\n\t\tNew Length of the Row Nanowire(cm)         ')
disp(new_L_row)
fprintf('\n\t\tLength of theColumn Nanowire(cm)       ')
disp(L_col)
fprintf('\n\t\tTotal Area of the crossbar array(cm^2) ')
disp(crossbar_area)
```

```
fprintf('\n\t\tTotal Area of the crossbar array with redundancy(cm^2)
')
disp(total_crossbar_area)
fprintf('\n\t\tFinal Area of the crossbar array with
redundancy(including multiplexer area)(cm^2) ')
disp(final_crossbar_area)
fprintf('\n\t\tNumber of Crossbar Junctions in 1cm^2 area')
disp(no_cbjuns)
```