The Future of Integrated Circuits: A Survey of Nano-electronics

Michael Haselman and Scott Hauck, Department of Electrical Engineering,
University of Washington, Seattle, WA
haselman@ee.washington.edu, hauck@ ee.washington.edu

Abstract

While most of the electronics industry is dependent on the ever-decreasing size of lithographic transistors, this scaling cannot continue indefinitely. Nano-electronics (circuits built with components on the scale of 10nm) seem to be the most promising successor to lithographic based ICs. Molecular scale devices including diodes, bistable switches, carbon nanotubes, and nanowires have been fabricated and characterized in chemistry labs. Techniques for self-assembling these devices into different architectures have also been demonstrated and used to build small scale prototypes. While these devices and assembly techniques will lead to nanoscale electronics, they also have the drawback of being prone to defects and transient faults. Fault tolerance techniques will be crucial to the use of nano-electronics. Finally, changes to the software tools that support the fabrication and use of ICs will be needed to extend them to support nano-electronics. This survey introduces nano-electronics and reviews the current progress made in research in the areas of technologies, architectures, fault tolerance, and software tools.

1. Introduction

Moore's law cannot hold forever. In 1975 Gordon Moore, cofounder of Intel, predicted that the number of transistors that could be placed on a chip would double every two years [Moore65]. Chip manufacturers have relied on the continued scaling down of the transistor size to achieve the exponential growth in transistor counts, but the scaling will soon end. Three obstacles stand in the way: the rising costs of fabrication, the limits of lithography, and the size of the transistor. For example, parts of the latest transistors are only a few atoms thick, and shrink with the scaling of transistors. Thus, when these reach the limit of 1-2 atoms thick, the scaling will have to cease and a new technology will have to be adopted. One possible heir to lithography based integrated circuits is nanotechnology and the nano-scale electrical devices.

Process scaling is fundamental to most of the benefits achieved by modern electronics. For some applications, scaling allows for more devices to be integrated on a single die, and thus provide greater functionality per chip. For example, increasing integration levels allow microprocessor designers to include things such as larger caches to speed up memory accesses and floating-point units to speed up floating point operations. Scaling also allows the same circuit to be smaller, cheaper, faster, and consume less power, thus driving new applications such as the cheap mobile electronics we now take for granted.

Ultimately, the goal of scaling is to build an individual transistor that is smaller, faster, cheaper, and consumes less power. Unfortunately, the scaling down of lithographically patterned transistors cannot continue forever, but nano-electronics may be able to continue the scaling when transistors hit their limit. Before we discuss nano-electronics, we first cover the structure and operation of MOSFETs, the building block of modern digital electronics. This will provide the necessary background to understand the issues that complicate the continued scaling of MOSFETs and provide a contrast to the nano-devices that will be surveyed later in this paper.

1.1 MOSFET Basics

The metal oxide semiconductor field-effect transistor (MOSFET) has been the building block for most computing devices for the last several decades. A MOSFET is a four-terminal device made up of a drain, source, gate and bulk (see Figure 1). In digital circuits, the MOSFET is essentially used as a switch. The source and drain are two ends of the switch, with the channel being turned on and off under the control of the gate. The gate controls the conduction through the channel through an electric field and is insulated from the channel by a thin of layer of silicon dioxide.

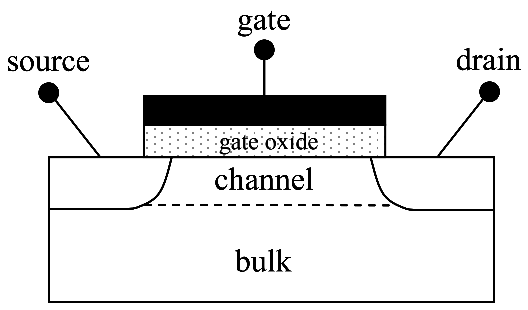

Figure 1: Illustration of a generic metal oxide semiconductor field-effect transistor (MOSFET).

There are two types of MOSFETs, nMOS and pMOS, differing in the voltages that turn on the switch. The type is dependent on element used to dope the silicon. Semiconducting materials such as silicon are not good conductors, so they are doped with other elements that either contain extra electrons (n-type) or are missing an electron (p-type). When the doping material has an extra electron, the majority carrier are electrons. When the dopant is missing an electron, the majority carrier is called a "hole". The extra holes and electrons are called carriers because they are the charged particles that allow current to flow. An nMOS transistor is a MOSFET with the drain and source heavily doped with an n-type material such as phosphorous, and the channel is lightly doped with a p-type material such as boron. A pMOS transistor on the other hand has p-type source and drains and an n-type bulk and channel.

The operation of a p-type MOSFET (where the drain and source are p-type semiconductors and the channel is n-type) is illustrated in Figure 2. Figure 2a shows a MOSFET in the "off" state, where no current it present in the channel. With no voltage potential across the gate and bulk, a depletion region forms around the drain and source blocking any current flow. A depletion region forms at a p-n junction when holes from the p-type material (source and drain in Figure 2) and electrons from the n-type material (channel in Figure 2) combine around the interface to create a region void of any free carriers. As the gate voltage drops, the electrons in the bulk are "pushed" away from the gate. When the voltage drops enough (beyond the threshold voltage), and enough electrons have left, the region just below the gate inverts to become p-type material (more holes than free electrons). There is now a continuous band of p-type material from the source to drain. This, along with an electric field set up from source to drain, causes electrons to move from hole to hole, creating a current.

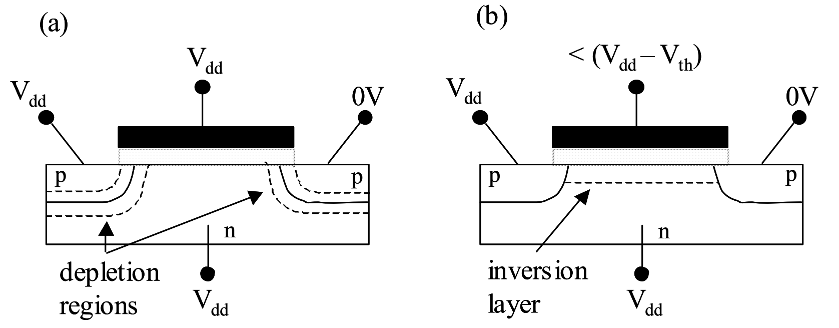

Figure 2: Illustration of the operation of a p-type MOSFET. (a) With no potential difference between the gate and bulk, a depletion region forms around the source and drain blocking current flow. (b) When the gate voltage drops a threshold voltage (V_{th}) below V_{dd}, the bulk just below the gate inverts to p-type, allowing current flow from the source to drain.

In addition to its ability to perform logic, the MOSFET also isolates the input from the output (gate to source or drain), which allows the transistor to exhibit gain. Gain is the ability for output voltage to reach the maximum operating voltage, even if the input to the gate is slightly less than the maximum operating voltage. This is important because a signal can go through thousands of transistors, and if a little voltage were lost at each the final signal would be severely degraded. Another key feature of MOSFETs is the ability to use them to build more complex structures. Complementary metal-oxide-semiconductor (CMOS), the most common logic family, uses complimentary nMOS and pMOS transistors to build logic gates such as inverters and NAND gates.

The MOSFET has been the primary building block of integrated circuits for more than forty years. The advances in electronics have been driven primarily by the ability to scale down the size of the MOSFETs used in integrated circuits. This scaling achieves improvements on many fronts. Smaller transistors allow more to be put on the same size chip, which has allowed integrations levels to rise from the hundreds of transistors when Moore made his prediction in 1965 to hundreds of millions of transistors today. Shrinking the feature size also makes each transistor faster and consume less power (This should not be confused with lower chip power, since the number of transistors per chip generally increases faster the than power per transistor decreases). The increase in speed comes from two factors: decreased capacitance and increased current. The capacitance of wires and gates lowers as these elements decrease in size, so the amount of charge a transistor has to place on a wire or gate decreases. The increase in current can be seen from the current flow equation for a transistor when the gate voltage is at its highest value. A first order approximation of the current through the channel is given by the equation [Jaeger97]:

$$I_D = \frac{\mu C_{ox} W}{2L}(V_{GS} - V_{th})^2 \tag{1}$$

The important part of equation (1) shows how different parameters of the MOSFET affect its performance. As the gate oxide thickness decreases, C_{ox} increases, which leads to higher current. A smaller feature size also means that the length of the channel (L) also decreases, which reduces the channel resistance. However, as the transistor scales down, V_{GS} (voltage gate to source) and V_{th} (threshold voltage that turns "on" transistor) are reduced. Up until recently, engineers have been reaping the benefits of the scaling down transistors without any significant disadvantages. That is beginning to change as the feature size (1/2 of the minimum distance between two adjacent gates) is reduced to tens of nanometers.

1.2 Issues around MOSFET scaling

The current projections by the International Technology Roadmap for Semiconductors (ITRS) say that the end of the road on MOSFET scaling will arrive sometime around 2018 with a 22nm process [ITRS05]. Even getting to 22nm presents some major unsolved hurdles. Among these are increasing power consumption, particularly through leakage currents, less tolerance for process variation, and increasing cost. Each of these issues are described in the following sections.

Leakage currents

An ideal transistor only has current flow when it is "on"; when the channel is "off" there is no current. This means that the transistor should consume no power if it is "off". Unfortunately, transistors are not ideal and, as they get smaller, they get less ideal. Leakage current, the flow of electrons through paths that should not conduct in an ideal transistor, now constitutes almost half of the power consumed by a chip [ITRS05]. Leakage currents come from two primary sources. Gate oxide leakage occurs when electrons jump ("tunnel") from the gate to the channel through the gate oxide. Scaling reduces the thickness of the oxide, and the thinner the oxide, the higher the leakage due to tunneling becomes. Subthreshold leakage occurs when a current between the drain and source is present even though the gate voltage is below V_{th} and the channel should be "off". Subthreshold leakage becomes worse as V_{th} is lowered and as the channel length is decreased, both of which generally occur when a transistor is scaled down.

Total Chip Power

With the advent of portable computing devices, power consumption is becoming a primary focus of IC manufacturers. Power is the product of current and voltage. The voltage is set by the fabrication process, so power is essentially dependent on the current levels in a device. Currents are considered in two separate areas; dynamic and static currents (the leakage current discussed above is a portion of static current). Dynamic current, and thus dynamic power, occurs when transistors are actively switching. Dynamic power per transistor is reduced with scaling, as less current is required to switch the transistor. However, the static currents, and thus static power, are increasing with scaling because of the leakage currents discussed above. Overall, power consumption is rising because of the increase in leakage currents, as well as the integration of more and more transistors. For example, the Intel Itanium 2 processor (90nm process) consumes about 177Watts at peak usage while the Intel Pentium (250nm process) consumes about 15Watts [Intel05]. Besides decreasing battery life of portable devices, power consumption creates heat, which degrades the chips' performance and must be dissipated. With increasing transistor density, localized heating can become a large problem. Heat increases the resistance of a transistor, thus decreasing its performance. This sets up the risk of thermal runaway, which can destroy a chip. Thermal runaway is a destructive cycle of increasing resistance causing increasing power consumption (heat generation), which in turn further increases the resistance.

Process Variation

Another drawback of scaling down the transistors is the decreased ability to handle fabrication process variations. As transistors and wires become smaller, fewer atoms make up the individual parts. For example, the gate oxide is currently only about five atoms thick. If merely a single atom is out of place, the gate oxide thickness varies by 20%. This lack of predictability significantly complicates the design process, and it will only become worse as scaling continues.

Costs

Probably the largest hurdle to further scaling of the MOSFET is simple economics. The cost of a fabrication facility is growing exponentially, along with the exponential growth of the number of transistors per chip. Currently (in 2005), a new fabrication facility costs around 3 billion US dollars [ITRS05] to construct, and this is rising exponentially. The exponential increase in cost is a direct result of the increase in mechanical precision required to fabricate the integrated circuits. Since the cost of the fabrication plant is spread across the cost of each chip, this drives up either the cost-per-chip, or the number of chips that must be produced.

1.3 Nano-electronics

Given the history of the semiconductor industry, most of these issues can probably be solved with current processes. However, there are two significant exceptions. Physical size limitations and astounding costs may require a shift in the fundamental way integrated circuits are fabricated. Many researchers believe this shift will be to nano-electronics. With a mix of chemistry, physics, biology and engineering, nano-electronics may provide a solution to increasing fabrication costs, and may allow integrated circuits to be scaled beyond the limits of the modern transistor.

The largest change in a shift to nano-electronics is the method of fabrication. Individual wires, diodes, field effect transistors (FETs), and switches can be created abundantly and cheaply in a test tube. All of these devices are only a few nanometers in size, and may reach a level of integration not possible with conventional ICs. It is estimated that nano-electronics will be able to integrate 10^{12} devices per cm^2, while the ITRS [ITRS05] estimatesp that at the end of the roadmap in 2018 manufacturers will only be able to achieve 10^{10} MOSFET transistors per cm^2.

This level of integration will be difficult to achieve due to the components' miniscule dimensions. It might be impossible to individually pattern the small components of the nano-electronics in the ways that current fabrication processes allow. While current ICs can have almost any arbitrary pattern, nano-electronics will likely have a regular structure generated by a stochastic self-assembly process. Unlike deterministic self-assembly, stochastic self-assembly means that chips will be fabricated with methods that allow components to guide each other in constructing a structure with little or no outside intervention. This is often referred to as a "bottom up" method, because the individual parts are built and then assembled into an architecture, and the use of the architecture is based on available resources. This is in contrast to a "top down" method used in current IC fabrication, where designs are conceived at a high level and the necessary components are put together to implement the design. The lack of outside intervention means that fabrication is more prone to defects and no single part can be absolutely relied on to be functional. In current lithography based electronics, the most popular model for handling defects is to reject any chip with even a single defect. This model will no longer work with nano-electronics because their defect densities will mean that no chip will be totally defect free. This suggests that nano-electronics will likely need to be reconfigurable like an FPGA in order to function in spite of defects.

In this paper, we consider the major research efforts for nano-electronics by surveying proposed technologies for replacing the transistor, possible chip architectures, techniques for handling defects, and software implications. We focus on the higher-level electronic aspect of these topics, though we provide references for readers interested in further details of the quantum mechanics, chemistry, and statistical analysis involved.

2. Technologies

The fundamental element of any nano-electronic circuit is the devices used to build it. For current VLSI systems these include silicon transistors and copper wires. For nano-electronics, it appears that the copper wires will be replaced by either carbon nanotubes (CNT) or silicon nanowires (SNW). The move to CNT or SNW is because they can be chemically assembled at much smaller sizes than copper wires can be patterned with lithography. There are a number of technologies that could replace the transistor as the basic logic device, these include negative differential resistors, nanowire or carbon nanotube transistors, quantum cellular automata, and reconfigurable switches. These devices offer sizes of a few nanometers, can be self-assembled.

2.1 Carbon Nanotubes

Carbon nanotubes (CNT) are cylindrical carbon molecules (Figure 3) that exhibit unique properties, making them potentially useful in areas including nano-electronics, materials, and optics. Their structure gives the nanotubes extraordinary strength, which is attractive for materials use, and can also increase the durability of a nano-electronic circuit over other materials. Nanotubes also possess electrical properties that make them attractive as nano-electronics wires and devices: they can behave as metallic wires or as semiconductors, depending on their structure.

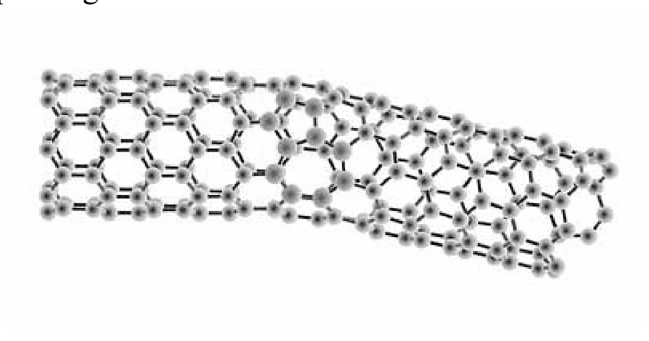

Figure 3: Illustrations of a single wall carbon nanotube.

Fabrication

CNTs were discovered in 1991 as a byproduct of an arc discharge experiment to create C_{60} buckyballs [Iijima91]. Since their discovery, two other fabrication methods have been discovered; laser ablation, and catalyst enhanced chemical vapor deposition (CCVD) [Graham05]. Arc discharge involves placing two carbon rods end-to-end about 1mm apart in an inert gas. An arc is induced between the two rods that vaporizes one of the rods. The vaporized carbon then reforms into nanotubes. Laser ablation uses the same mechanisms, but instead of using an arc to vaporize a carbon rod, a laser is used. Laser ablation produces purer CNTs than arc discharge. One drawback to both of these methods is that they also produce carbon sheets, fullerenes, and random carbon structures in addition to nanotubes, requiring a separate purification step to extract the nanotubes from the collection of carbon structures. This purification step is typically done in a solvent, and depositing the purified nanotubes on a substrate results in a random placement of the tubes. CCVD tackles the problem of nanotube placement by growing the nanotubes at a desired final location. In CCVD a catalyst particle is placed on a silicon wafer using photolithography or a random method, and carbon gas is passed over the wafer. The catalyst induces the growth of the nanotube. Besides growing the tubes in place, CCVD has the advantage of not producing other stray carbon structures.

Many different CNT structures can be produced with each method, and the properties of the nanotube are dependent on its structure [McEuen00]. CNTs behave as a metal or semiconductor depending on their chirality. The chirality is the amount of "twist" present in the tube. If you think of CNTs as a rolled-up graphene sheet made up of hexagons (see Figure 4), the chirality is how far the axis of the tube (line down center of tube) is from being parallel to one side of the hexagons (y-axis in Figure 4) [Raja04]. If the tube axis is parallel, the CNT will be semiconducting.

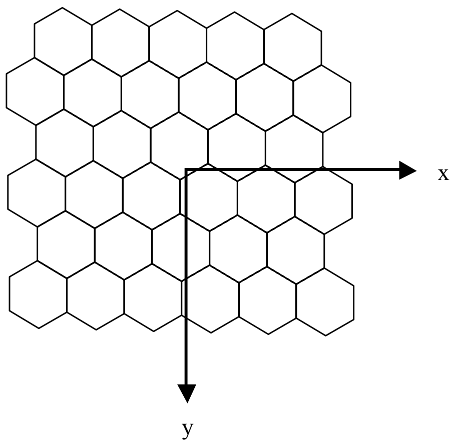

Figure 4: Illustration of the chirality of a CNT. If the nanotube is rolled up around the x-axis, the nanotube will be a metal. If the nanotube is rolled up around the y-axis, it will behave as a semiconductor. [McEuen00]

A second property of nanotubes that affects their electrical properties is the number of walls. Figure 3 shows nanotubes in two different configurations; single-walled and multi-walled. The main difference between the two varieties is the diameter of the tubes. The diameter of single-wall carbon nanotubes (SWCNT) are generally between .7nm and 2nm while multi-wall carbon nanotubes (MWCNT) are typically between 10nm and 20nm, depending on the number of walls [Graham05].

The bandgap energy is inversely proportional to the diameter of the nanotube [Graham05]. The bandgap energy is "the minimum energy required to break a covalent bond in the semiconductor crystal, thus freeing an electron for conduction" [Jaeger97]. In other words, the lower the band gap energy, the better a conducting material. The diameter of single wall nanotubes puts their band gap energy at levels that are good for transistor or diode applications. The larger diameter of MWCNTs decreases their band gap energy so low that they behave like metals regardless of their chirality.

CNT electrical devices

Currently, the most promising use of semiconducting CNTs is as a transistor component. As can be see in Figure 5, carbon nanotube field effect transistors (CNTFET) appear very similar to MOSFETs, with the silicon channel replaced with a CNT. Most of the CNT transistors have been fabricated with SWCNTs [Graham05, Bachtold01, Martel98] because their band gap energy is in the range of a semiconductor. One group, however, found that MWCNTs could be used if the nanotubes were collapsed or crushed [Martel98]. This is probably impractical for large-scale systems, since each nanotube would have to be individually collapsed or selected amongst many "normal" nanotubes. Two varieties of CNT transistors have been fabricated. Figure 5a shows an illustration of a CNT transistor with a back gate (gate placed under the channel instead of over it), which uses the silicon substrate to control the conduction through the CNT. The use of a back gate is easier to fabricate, but it has the disadvantage of not being able to control the individual transistor because the substrate is shared between all transistors. This configuration is good for research, but is probably not a realistic candidate for commercialization. The other variety uses a gate that is over the top of the CNT, as in Figure 5b [Graham05, Wind02]. These so-called second-generation CNT transistors have two advantages over their counterparts with a back gate [Wind02, Kanwal03]. The most obvious advantage is the ability to individually control the FETs because the gates are isolated. The gate on top also allows for a thinner gate oxide, which means that the controlling voltage can be lower. Also, CNTs are intrinsically p-type, but they can be altered to behave as an n-type semiconductor [Nosho05]; however, exposing an n-type CNT to oxygen will cause it to revert back to its native p-type. Covering the CNT with the gate is a good means to isolate it from oxygen. Individual gating and the formation of both p-type and n-type allows for CNT transistors to be arranged in complementary pairs much like current CMOS. Unfortunately, it is much more difficult to fabricate these transistors with a top gate.

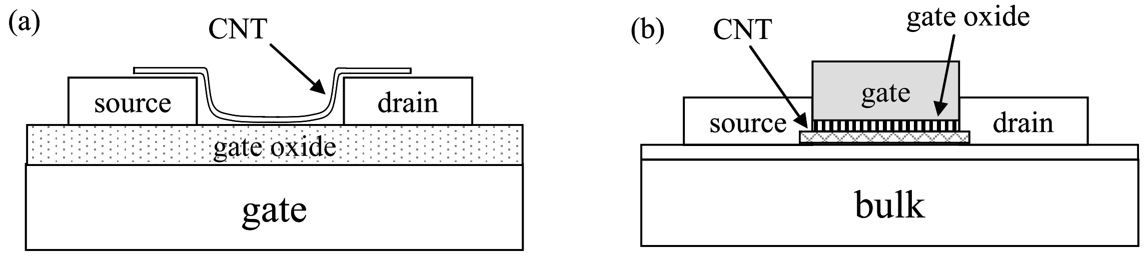

Figure 5: Carbon nanotube FET (a) with a back gate [Martel98] and (b) a top gate [Wind02]. A back gate uses the substrate to control the conduction through the CNT, while a top gate uses a conventional gate that covers the CNT (channel).

CNT based transistors have promising enough characteristics to prompt companies such as Intel, NEC and IBM to investigate them as replacements fot modern transistors. The first advantage is the small size of the CNT. The small diameter of the CNT means that all parts of the channel are close to the gate, and they are easier to control. Another advantage of using CNTs is that they exhibit ballistic transport of electrons because of the tube structure. Since all of the atoms in the tube are bonded to the same number of neighbors, there is no electron backscattering. This is in contrast to a wire made of a crystal, which has irregular bonds at the surface. Ballistic electron transport means that transistors with CNTs will exhibit higher on currents that will not be affected by the length of the transistor channel. For MOSFETs, the current decreases as the channel length (distance between the source and drain) increases. An unsolved problem with the use of CNTs for the transistor channel is increasing the width of the channel. For MOSFETs increasing the width of the channel (dimension into the page on Figure 1) increases the current drive capabilities of the transistor, which is absolutely critical for circuit design. With CNT transistors, the only way to achieve this would be to "lay" CNTs side by side, since the tube dimension is set. Unfortunately, there is currently no technique for performing this.

Although CNT transistors and the MOSFETs discussed in section 1.1 behave alike and appear very similar in structure, the operational physics are very different. The CNT is not in contact with the bulk to transfer carriers, as is done with MOSFETs. The transistor behavior arises from Schottky barriers at the source/CNT interface [Appenzeller02] and its interaction with applied electric fields. Schottky barriers are formed when a metal and a semiconductor are joined together, and there is an energy difference between the Fermi level (E_f) of the metal and the energy level of the carrier (holes or electrons) of the semiconductor. The Fermi level is the top energy state possible for an electron in the metal at 0 Kelvin. When the Fermi level of the metal is between the conduction (E_c) and valence (E_v) band of the semiconductor, carriers have to acquire energy to move between the source and the semiconductor. In order to clarify the process of how the transistor is turned on and off, an example of a p-type CNTFET is given in Figure 6. Figure 6a shows the band energies of the CNTFET without any voltage stimulation. The Fermi levels of the source and drain are different because the positive voltage on the source lowers the energy level and raises the Fermi level of the drain. When there is no bias on the gate, the Fermi level of the source is higher than the energy level of the holes in the valence band of the CNT. This barrier means that very few electrons can move from the holes in the CNT to the source even though an electric field exists between the source and drain. (even though holes are carriers in p-type transistors, it is electrons moving between the holes in the valence band that actually create the current). When a negative bias is placed on the gate in Figure 6b, the valence and conduction bands are raised. Except for a small portion near the source/CNT interface, the valence band is above the Fermi level of the source. This means that the Schottky barrier is very low and electrons easily tunnel from the CNT valence band to the source because they are in a higher energy state for most of the valence band. The process is the same for an n-type transistor, except that now the electrons move through the conduction band, the Schottky barrier would be on the drain side of the diagrams in Figure 6, and a positive bias on the gate would lower the conduction and valence bands in Figure 6b.

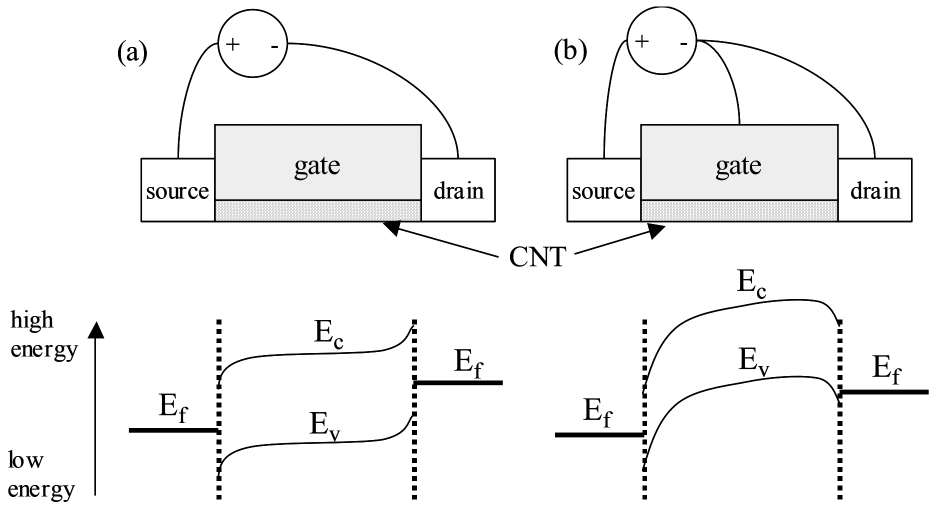

Figure 6: Band structure diagram of a p-type carbon nanotube field effect transistor. (a) With no bias on the gate, a large Schottky barrier exists between the valence band on the CNT and the Fermi level (E_f) of the source. A positive bias on the source lowers the Fermi level of the source and raises the level of the drain. (b) A negative bias on the gate raises the conduction and valence band of the CNT. The shift in bands lowers the Schottky barrier at the source/CNT interface and allows holes to be transported from the source to the valence band of the CNT.

Another promising application that takes advantage of CNTs strength properties instead of their electrical properties is as non-volatile memory devices. The first proposal was an array of SWCNTs with contacts at one end of each CNT (see Figure 7) [Rueckes00]. One layer of CNTs sits on the substrate while the other layer is suspended over the first layer by a spacer. To write to the memory, opposite charges are placed on two orthogonal CNTs. The opposite charges causes the two CNTs to be attracted to one another. Once the two CNTs make contact, molecular bond forces called van der Waals forces keep them together, even if the opposite charges are released. The two contacting CNTs now have a non-infinite resistance between each other, and are considered on or '1'. Locations where the CNTs have not been bent, and thus there is no connection between the perpendicular CNTs, are a '0'. To read a cell, current is sent down one CNT; if current is detected on the output of the orthogonal CNT, the two CNTs are making contact. A like charge can be placed on two contacting CNTs to separate them and erase a '1'. Mechanical forces will keep the two CNTs separated when the like charges are removed. The fact that the CNTs stay in their configuration without electrical charge due to van der Waals or mechanical forces makes this memory non-volatile.

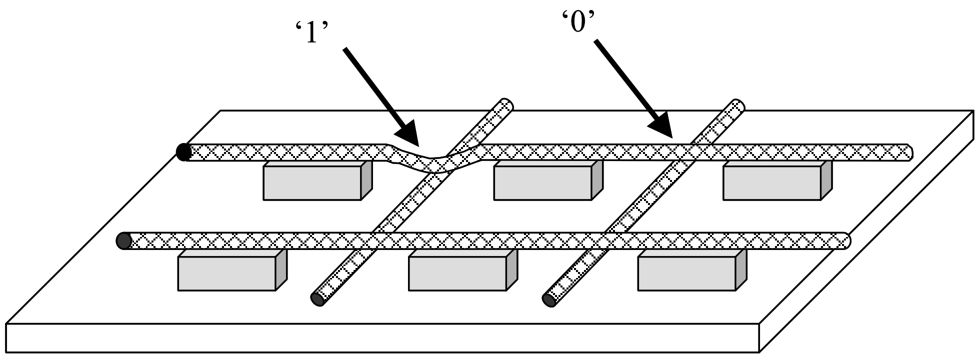

Figure 7: A three dimensional view of four memory cells of a CNT non-volatile RAM. [Rueckes00]

The RAM in Figure 7 requires two layers of CNTs, with a placement of the top layer over the spacers. This is a difficult task with CNTs, so the design was modified to only have one layer of CNTs, which are suspended over metal electrodes (see Figure 8) [Ward04]. The metal electrodes are arranged in long troughs, and the CNTs are placed orthogonally over the troughs, eliminating the need for exact placement. To increase the robustness of the memory, each cell contains multiple CNTs connected to a contact. The read/write procedure is identical to the above architecture.

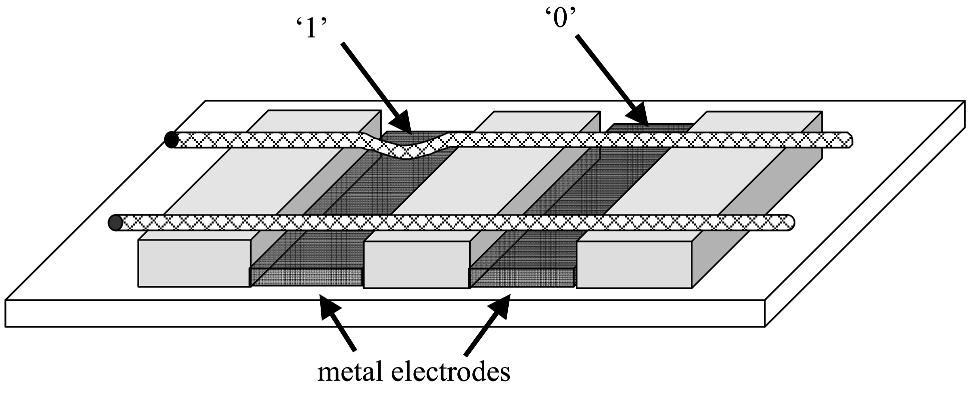

Figure 8: Cross-section view of a CNT memory cell with metal electrodes. [Ward04].

Despite the many good qualities of CNTs, many hurdles must still be overcome before devices built with this technology are feasible. Most of these issues surround the fabrication of the CNTs [Graham05]. One problem is that while it is possible to bias the process to produce more of more of one kind of CNT (semiconducting or metallic), all methods of fabrication produce some of both. A method has been developed to separate the two varieties [Krupke03], but requires suspending the CNTs in a solution. This will not work for CNTs that are grown in place with CCVD, and since CCVD is likely the best solution for getting CNTs arranged into some kind of structure, putting the CNTs into solution seems impractical. Other aspects of fabrication that are not currently controllable include the diameter and the chirality of the nanotube. Since chirality and diameter affect the electrical properties of the nanotubes, and uniform device characteristics are critical to circuit design, it is very important to devise a method for obtaining consistent nanotubes.

2.2 Semiconducting Nanowires

Semiconducting nanowires (NWs), like CNTs, can be used as interconnect wires to carry signals as well as be used as an active device. While one CNT is either an active device or a wire, a single NW can be both an active devices and an interconnect wire. NWs are long thin wires made up of semiconducting materials, such as silicon or germanium that have been fabricated with a diameter as small as 3nm[Cui01a, Morales98] and a length of up to hundreds of micrometers [Wu00]. The diameter is about eight times smaller than lithographic-based fabrication methods will likely ever be able to achieve.

Fabrication

The growth of NWs has been achieved by methods such as laser ablation [Morales98], chemical vapor deposition [Wu01], and Vapor-Liquid-Solid (VLS) synthesis [Morales98, Wu00], or a combination of a couple of these methods. These methods are also employed to produce carbon nanotubes, except that instead of carbon, a semiconductor is used for the raw material. If a nanowire is going to be used as a semiconductor, the method of growth should have enough control that the dopant levels of the nanowire can be controlled along its length. One such method of controlled growth is VLS (see Figure 9). VLS growth is a method of growing crystalline structures using a liquid catalyst or seed such as gold or iron. The catalyst is in a chamber with vaporized nanowire materials (silicon or germanium plus a possible dopant). All of this is done in a heated chamber, where the temperature is kept high enough that the catalyst remains a liquid. The liquid catalyst absorbs the vaporized materials until it becomes supersaturated, at which point a solid crystal begins to form. The nanowire will continue to grow until the catalyst is cooled and becomes solid, or the vaporized crystalline material is used up.

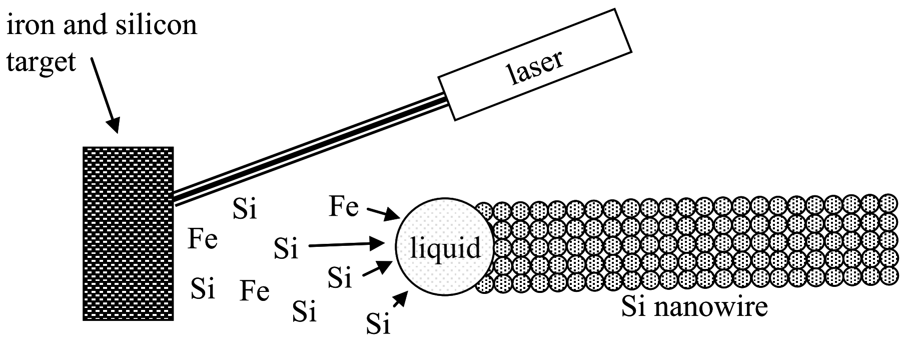

Figure 9: A proposed silicon nanowire growth method. Laser ablation is used to vaporize an iron and silicon target. The hot vapor condenses in a liquid catalyst, and the temperature is kept such that the iron/silicon seed remains liquid. The silicon nanowire grows as the catalyst absorbs more silicon and becomes saturated. [Morales98]

The size of the catalyst determines the diameter of the nanowire [Cui01a]. The catalysts are composed of metals such as gold or iron, and can be created with laser ablation [Morales98] of a target that contains both the metal and the nanowire material. Laser ablation has been shown to create very uniform diameter catalysts, which in turn creates uniform diameter nanowires. This provides relatively uniform electrical characteristics.

When dopant materials such as boron or phosphorus are added to the vapor, the nanowire will become semiconducting. It can act as a p-type or n-type conductor, depending on the dopant [Cui00]. In addition, the NWs can be so heavily doped that they begin to conduct like a metal [Cui00]. The controlled growth of the nanowires also allows for the doping to be varied along the length of the nanowire. This is done by controlling the type and amount of dopant material present in the vapor at specific time intervals. Nanowires can also be coated with different materials after fabrication [Lauhon02], resulting in a wire with a semiconducting core and an insulative covering. To form this covering, once the nanowire is made, a new material is vaporized so that it will bind to the whole wire, leading to a thin, uniform sheath. If this sheath is composed of an insulative material such as silicon dioxide, it can electrically isolate the NW. This can insulate overlapping wires from one another, or it can separate parallel wires [Whang03b] to help form an array (section 3.1).

Nanowire electrical devices

By controlling the doping profile along the length of the NW, active devices can be integrated into a NW. A field effect transistor (FET) can be created if a nanowire has a small section that contains fewer carriers than the rest of the wire [Gudiksen02]. Lowering the concentration of the dopant atoms in the growing atmosphere for a period of time can make this lesser-doped region. If another wire is placed over the top of this region, with an insulator separating the two wires, a FET is created. To control the current, a charge is place on the top wire to deplete the carriers in the FET regions of the lower wires [Huang01b]. The rest of the wire is not affected because its concentration of carriers is high enough that it is not depleted. Another way to create a device, which doesn't require another controlling wire, is to create a p-n junction diode. This can be done in two different manners; the easiest is to simply cross one p-type and one n-type semiconducting NWs, creating a connection [Cui01b]. Where the two wires contact each other, a p-n junction is formed. The other way to create a p-n diode with a nanowire is to create one on a single wire [Gudiksen02]. This is done by growing part of the nanowire with a p-type dopant, and then switching to an n-type dopant for the remainder of the nanowire growth.

There have also been experiments using nanowires as the channel in a more conventional FET, similar to what is done with CNTs [Cui03] (as in Figure 5). NW FETs have a few advantages over CNT FETs. One is that NWs will remain n-type and p-type when exposed to oxygen, while CNTs will revert from n-type to p-type. A much larger advantage is the ability to control the doping, and therefore the semiconducting properties of the NW during construction. Recall that with CNTs, the conduction is dependent on the chirality of the tube, which cannot currently be controlled during fabrication.

The ability to grow NWs hundreds of micrometers long makes them attractive as interconnect wires as well as devices. Due to their size, nanowires show unusual electrical properties. Unlike CNTs, which exhibit ballistic conduction, nanowire conduction is influenced by edge effects. The tube structure of carbon nanotubes dictates that all atoms are fully bonded to other atoms (in a defect-free structure). However, NWs are a solid wire, and therefore atoms on the edge are not completely bonded. While the core of the NW is metallic, and thus conducting, the atoms on the outside of the wire lower the conductivity of the wire because they often contain defects in the crystalline structure. As the nanowire shrinks, the atoms on the surface of the wire represent more and more of the overall structure. The edge effects become more prominent, worsening the overall conduction of the NW.

At first glance, NWs and CNTs seem to be very similar. Both are capable of forming active devices and interconnect wires with dimensions of a few nanometers. However, there are some differences that make NWs more promising than CNTs. While CNTs are physically strong, and their metallic form has excellent conduction properties, the inability to grow CNTs with desired properties is a major obstacle to their large-scale usage. Current methods for creating CNTs produce both semiconducting and metallic structures, and their semiconducting characteristics even vary from tube to tube. On the other hand, the doping levels of NWs, and thus their conduction properties, can be very tightly controlled. The doping levels can also be varied along the length of a NW, while a CNT is either all semiconducting or all metallic. As discussed previously, this control provides many more active device possibilities for NWs. Also, techniques for creating regular arrays are much more developed for NW's (section 3.1) than those for CNTs.

2.3 Molecular Devices

Even though NWs and CNTs can be used as active devices as well as wires in nano-electronics, there is also a set of molecules that could be used as the active devices. These molecules behave as diodes or programmable switches that can make up the programmable connections between wires. Chemists have designed these carbon-based molecules to have electrical properties similar to their solid-state counterparts. Molecular devices have one huge advantage over solid-state devices: their size. Thousands of molecules can be sandwiched between two crossing micro-scale wires to create an active device that takes up very little area. Current VLSI crosspoints made of pass transistors are 40-100 times larger than a wire crossing or via [Butts02]. Since molecular devices fit between the wires, large area savings could be achieved. For example, it has been estimated that the use of nanowires and molecular switches could reduce the area of an FPGA by 70% over a traditional SRAM based design at a 22nm process [Gayasen05]. In addition to being very small, molecular devices tend to be non-volatile: the configuration of the molecules remains stable in the absence of electrical stimulation. In the presence of electrical stimulation, programmable molecular device can be turned "on" and "off", which can be used to perform logic.

Molecular diodes

Diodes are devices that generally act as a one-way valve, allowing current to flow in only one direction. Modern diodes are built by mating n-type and p-type semiconducting material. Diodes are generally not used as logic devices because they are static devices that consume lots of power. Static devices cannot be turned "on" and "off"; they simply conduct under a positive voltage bias and do not conduct otherwise. If a diode could be turned "off" so it does not conduct even with a positive voltage bias, they would have greater use. This is essentially what researchers have developed with diodes made out of molecules.

One such diode is a molecular resonant tunneling diode (RTD). Molecular RTDs exhibit negative differential resistance (NDR) that can be turned on and off [Chen99, Chen00]. Devices that exhibit NDR have a region of their I-V curve that has a negative slope known as the NDR region (see Figure 10). A negative slope indicates that the current reduces as the voltage increases. The I-V curve has two important voltage points: the peak and the valley. The peak voltage is the point of highest current value, and the valley voltage is the point where the current is the lowest when the voltage is above the peak voltage. The important metric of an RTD is the ratio of the peak current versus the valley current (PVR). The larger the ratio, the easier it is to differentiate between the two states. PVRs of 1000:1 have been observed at cold temperatures, but at room temperature, the PVR decreases to 1.5:1 [Chen00].

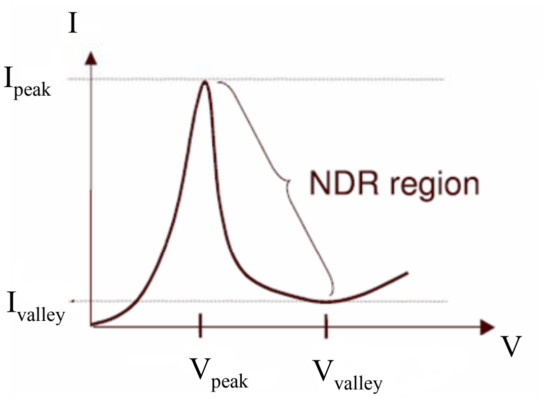

Figure 10: I-V curve of a molecule that exhibits negative differential resistance. [Husband03]

The NDR region is important because it allows negating logic to be built with these devices [Ellenbogen00]. Figure 11 shows an example of an XOR gate implemented with molecular rectifying diodes and a resonant tunneling diode. If one of the inputs (A or B) is high, while the other is low, the voltage across the RTD will be at the peak of the I-V curve. Thus, a high current will be present in the RTD, and the output will be high. If both of the inputs are high, the voltage will put the operating region of the RTD in the valley of the I-V curve, where there is very little current conduction. The lack of current will result in a low voltage on the output. Likewise, if both inputs are low, there will be no current through the RTD so the output will be low.

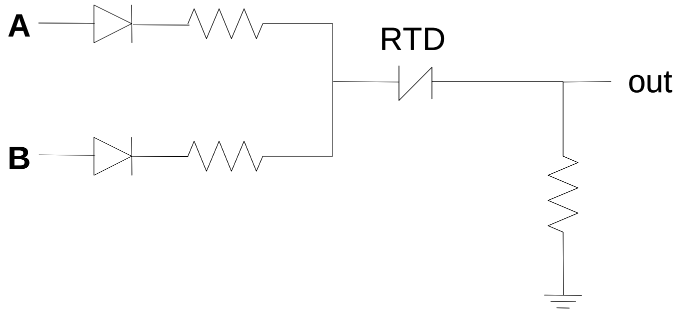

Figure 11: Molecular XOR gate made up of molecular rectifying diodes and a molecular resonant tunneling diode (RTD). [Ellenbogen00]

As previously mentioned, an important characteristic of molecular RTDs is that they can also be turned "on" and "off". The molecule has two different stable configurations. In its "on" state it conducts electricity, while in the "off" state it has a very high resistance and conducts very little current, even if a large voltage is place across the diode. Applying a voltage above a certain threshold changes the configuration of the molecule. For the RTD molecule in Figure 10 the thresholds are 1.75 and -1.75 Volts to turn the molecule "on" and "off" respectively. Once the molecule is configured, it is operated with a voltage (~.5V) [Whitaker88] that is less than the threshold to avoid switching the configuration. The configuration of this molecule has been shown to be stable for up to 15 minutes [Reed01].

Molecular RTDs have also been used to build a latch [Goldstein02, Mathews99]. A molecular latch uses two RTDs; one designated the drive RTD and the other the load RTD (see Figure 12). There are three "equilibrium" values of V_{ref} where the current through the two RTDs is equal. This means that in the absence of any input current, the data node between the two RTDs will be in equilibrium. Two of these states are stable. These are indicated as "0" and "1" in Figure 12, and represent the state of the latch accordingly. The third equilibrium state is not stable. When the latch is in this state any shift left or right will result in a large increase in current through one RTD, and a decrease in the current through the other, thus changing the data node voltage.

To store a new value in the latch, V is lowered to V_{mono}, as shown in Figure 12. Once the latch has reached the new steady state, V is raised back to V_{ref} while the input current is applied to the in node. If the input current is above a certain threshold, V_{out} will be high and the latch will stabilize in the "1" state. Likewise, if the input current is low, V_{out} will be low, and the latch will settle in the "0" state. These latches have been used to make memory cells where V_{ref} is a clock signal and the latch is refreshed on each clock cycle [Rose03].

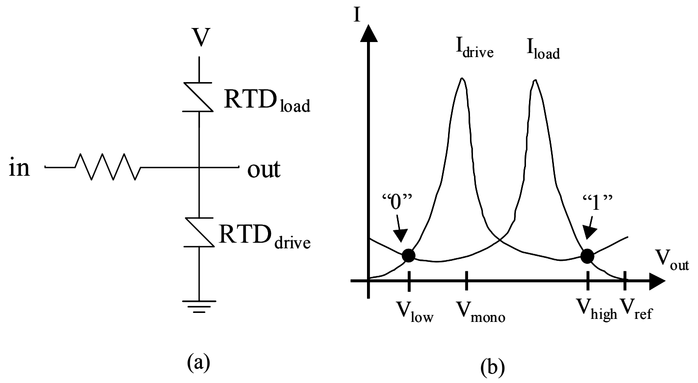

Figure 12: (a) Circuit diagram of a molecular latch using molecular RTDs. (b) Line load current diagram for molecular latch as a function of the output node voltage. [Goldstein02]

While the previous diodes have non-linear current-voltage characteristics, in is important in some architectures to restrict current to one direction. In semiconductor electronics, this is achieved by a rectifying diode. Researchers have been able to build rectifying diodes with molecules as well [Metzger97]. Although these devices cannot be switched "on" and "off" like the RTDs discussed above, they still may have a place in molecular electronics. They can be used in logic, as shown in Figure 11. They also can be used to control the direction of current flow in an array structure (section 3.1).

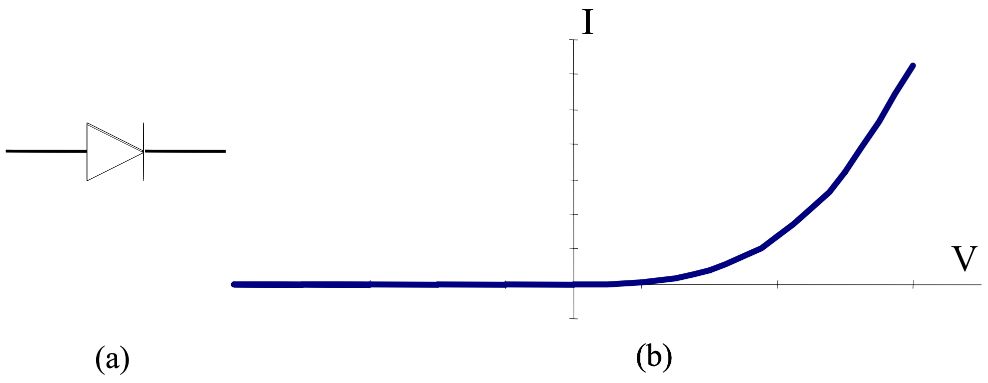

Figure 13: (a) Schematic symbol of a rectifying diode and its (b) I-V curve.

Molecular switches

In addition to molecular diodes, there are also molecules that behave like simple switches. The most widely-known molecular switches are from a group of molecules called rotaxanes and catenanes. Rotaxanes and catenanes are molecules that are made up of two or more components that are mechanically linked [Kay03]. This means that the components can move in relation to one another without breaking covalent bonds. Catenanes are made up of two or more interlocking rings, as shown in Figure 14a. Rotaxanes consist of at least one ring (called a macrocycle) that is trapped on a rod that has two bulky ends, which prevents the ring from "sliding" off (see Figure 14b).

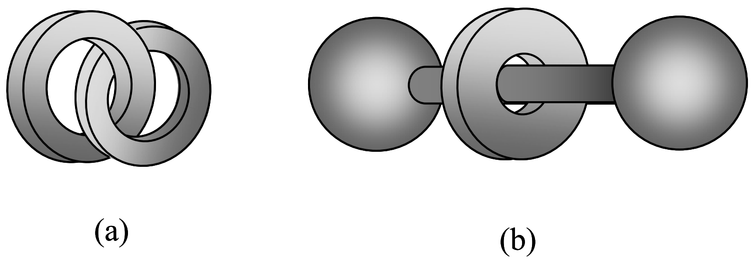

Figure 14: Illustration of (a) [2]catenane and (b) [2]rotaxane molecules. [Kay03]

Certain rotaxane and catenane molecules have been shown to behave as molecular switches that can be programmed on and off [Collier99, Collier00, Brown00]. [2]Rotaxane [Collier99, Stewart04] (the "[2]" is common chemistry nomenclature for the number of components in the molecule) and [2]catenane [Collier00, Brown00] are molecules that have been fabricated and shown to exhibit hysteretic I-V characteristics (see Figure 15) with two stable states. Hysteresis in this case means that the device turns on and off at different voltages. This is illustrated in Figure 15 where the molecule starts conducting (turns on) at about 1 volts and stops conducting (turns off) at about -1.5 volts. The molecules are switched "on" and "off" with high voltage, and operated with lesser voltages. For example, [2]catenane is switched on with 2 volts, off with -2 volts, and is read with ~0.1 volts [Collier00]. These molecules are mechanically switched "on" and "off" when one component is moved in relation to the other by either oxidation (removal of electrons) or reduction (addition of electrons). For [2]catenane, one ring rotates through the other, and for [2]rotaxane the ring slides back and forth on the rod. These molecules are essentially variable resistors that can be switched between two resistance values. For example, [2]rotaxane has a 200X difference in resistance between its "on" and "off" state. Since they are resistors, current can pass both ways through the molecule, as shown in Figure 15, as opposed to the diode discussed above. Note that some of the references show molecular switches behaving as diodes, but this is an artifact of the material that the molecules are connected to for testing, rather than the molecular switch itself [Collier00].

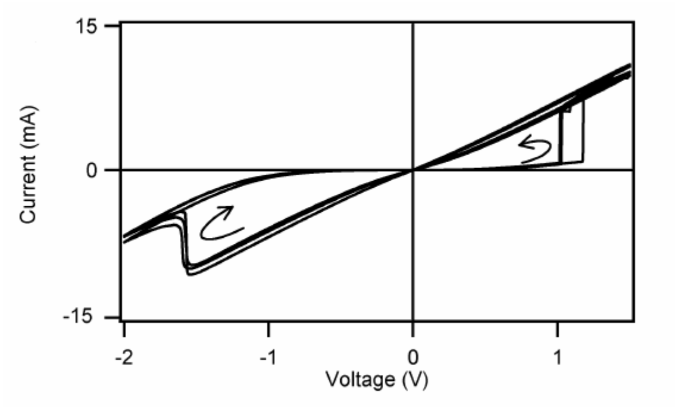

Figure 15: I-V curve of a [2]rotaxane molecule cycled on and off multiple times. The curve is linear when the molecule is "on", but the current drops off when it reaches the "off" threshold, around -1.5 volts. It turns back "on" at about 1 volt. [Stewart04]

Since these molecules conduct current in both directions, this may limit the applications which molecular switches can be used in. Although architectures based on these molecular switches have been proposed [Snider05b], resistors alone are not ideally suited for performing logic because of signal degradation. These switches will have to be incorporated with other devices to create logic (section 3). Molecular switches are probably better suited for memory devices where only one transistor is encounter per memory read.

Even though these molecules conduct in both directions, it is important to orient these molecules. This is important because if molecules are arranged in both "directions", an attempt to turn "off" the molecules will turn "off" some but will also turn "on" others. This is accomplished by engineering different characteristics for each end of the molecule. For example, making one end hydrophobic (repels water) and the other end hydrophilic (attracts water) could be used to help align the molecules in that same direction during assembly.

3. Architectures

All of the devices discussed above have been fabricated and tested to differing degrees. However a crucial step is to integrate these devices into an architecture that takes advantage of their strengths and overcomes their limitations. An efficient architecture is strongly dependent on the available devices and manufacturing capabilities. With current transistors and lithography, essentially any circuit can be created and manufactured with high reliability. This level of control is unlikely to be possible for nano-electronics. Because of their small size, nano-electronic devices will likely not be able to be deterministically placed. Researchers have been able to manipulate components with atomic force microscopes, but this will be impractical for full chips. Even if advances in manufacturing allow other ways to manipulate at this scale, the tolerances required will likely make the costs prohibitive. Current approaches to these problems include three major ideas:

1) Let the circuits assemble themselves (self-assembly).
2) Manipulation at a higher level (i.e. guide a group of wires so they line up in the same direction).
3) Use a totally random process (i.e. place enough elements until statistics imply that things should work).

This is a significant departure from micro-scale fabrication, where carefully-controlled lithographic processes dictate the placement of each individual element. There are some consequences to this bottom-up approach to assembly [Stan03, Goldstein01].

1) Defects are inevitable and must be handled (section 4).
2) Three-terminal devices will be hard to fabricate. While a two-terminal connection can be established merely by overlapping two wires perpendicularly, the stochastic nature of the assembly means that the probability of aligning three things will be very low. Two-terminal devices such as nanowire FETs, diodes, and molecular switches will be preferred.
3) Wire to wire connections will need to be achieved by orthogonal overlapping of the two wires. The inability to manipulate individual wires means that it will likely be impossible to assure two parallel wires will line up end-to-end or even overlap.
4) Nanoscale to microscale connections will have to be sparse, and should be done with orthogonal overlapping. Similarly to point 3 above, it will likely be impossible to assure an end-to-end connection of microscale and nanoscale wires. Even if connection could be assured, the microscale wire pitch would greatly spread out the nanowires, negating the area savings of nanowires. Also, because of the size difference, nanoscale elements will be slow when driving microscale devices.

3.1 Array-based

Currently, the most popular architecture for nano-electronics is array-based design with nanowires or nanotubes overlapped to make a grid. The reason for their popularity is that techniques for creating them are well established, and such arrays address many of the issues discussed above. It is impossible to select individual junctions of an array to contain switches, so arrays will likely be full crossbars. This full crossbar nature makes it easy to avoid defects since any line can be replaced by another line with the same orientation (horizontal or vertical) [Naeimi04]. The positions where two wires overlap can create many two-terminal devices, including ohmic contacts[Rueckes00], programmable switches[Collier99, Stewart04, Collier00, Brown00], and diodes[Chen99, Chen00, Metzger97]. Finally, as we will discuss, parallel nanoscale wires attach more easily to microscale wires than trying to line up two wires end-to-end [DeHon03a, Williams01, Snider05a].

Array creation

There are several methods that can align nanowires and nanotubes into parallel rows several nanometers apart; the Langmuir-Blodgett flow is one such technique [Whang03a, Huang01a, Whang03b]. Irving Langmuir and Katherine Blodgett discovered this technique in the early 1900s for depositing a single layer of molecules on a film. The nanowires or nanotubes are suspended in a liquid that flows over a substrate. As the liquid flows over a Langmuir-Blodgett trough the wires are compressed in order to line them up (see Figure 16).

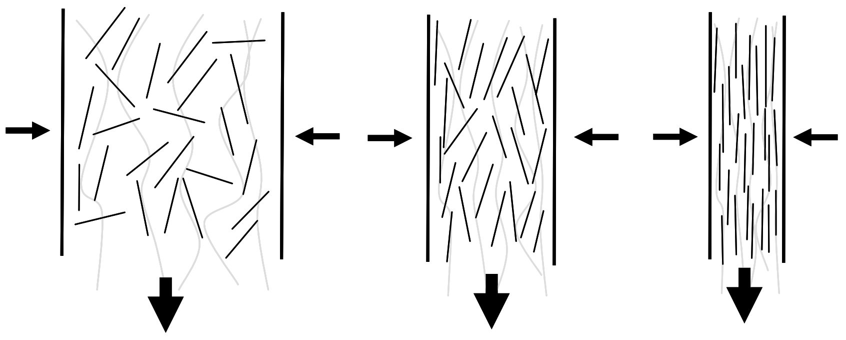

Figure 16: Illustration of the Langmuir-Blodgett technique to create parallel nanowires or nanotubes [DeHon05b]. A random assembly of nanowires (dark lines) is progressively squeezed in the x-direction, while fluid is flowed in the y direction, to create a parallel set of wires.

A layer of oxidation grown around the wires (see section 2.2) controls how closely the wires can be packed together (their pitch). Once one layer of wires is deposited on a substrate, another flow can be performed and deposited at a right angle to the first layer, thus creating a grid. Notice that while this technique can control the alignment of most of the wires in the direction of the flow to within a few degrees [Huang01a], there is no control over where the end of a wire will line up, or where any particular wire is deposited in the array. This lack of precise control will have large implications when circuits are fabricated with this technique. This technique has been able to deposit nanowires with an average pitch of 90nm [Whang03b], but it is believed that pitches 3-4x smaller will be achievable.

Nanoimprint lithography is also a well-established technique for producing aligned nanowires [Chou96, Chen03]. Nanoimprint lithography is similar to conventional lithography, except instead of using light to etch a pattern, this technique uses a mold. The first step is to create a mold in the pattern of the desired array. This is done with electron beam lithography. Electron beam lithography can achieve significantly smaller feature sizes than current lithographic processes. Current technologies use light and a mask to pattern VLSI circuits. The light is "shined" on the mask, which has cutouts of the desired pattern, and the light passes through the cutouts to create a pattern on the photoresist. This method is quick because the whole pattern is etched at once, but it has a resolution limit of about 45nm [ITRS05] due to the mask causing the light to diffract. Electron beam lithography uses an electron beam to pattern a circuit. The electron beam draws out the desired pattern on the photoresist directly instead of using a mask. This method can achieve a resolution of about 10nm [Chou96], but it is a slow process. This time consuming process is acceptable for nanoimprint lithography though because it only has to be done one per mold. That mold can be copied countless times [Schulz00], and those copies can be used to make imprints very rapidly.

The second step in nanoimprint lithography is to press the mold into a layer of resist over a substrate (see Figure 17a). After the mold is removed (Figure 17b), some resist remains in the compressed channels. This extra resist is etched away with a process called reactive ion etching (RIE) to reveal the substrate (see Figure 17c). The final step is to deposit metal onto the pattern to fill in the channels and create the wires. Even though this process has been used to create a one-kilobit memory at a 30nm half-pitch (half the distance between two lines) [Wu05], there is some doubt about how small a pitch is achievable, and whether multiple molds will be able to be aligned [Chou96, Stan03].

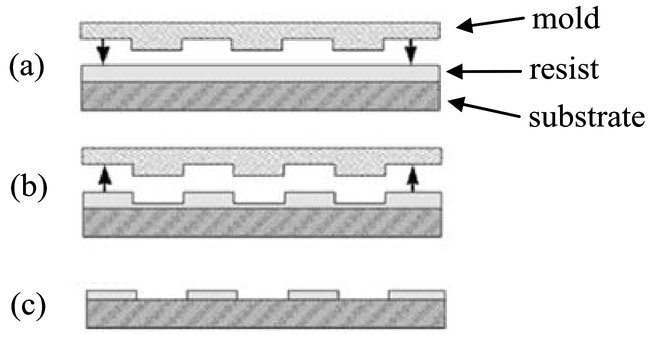

Figure 17: An illustration of the nanoimprint lithography process [Chou96]. A mold is pressed into resist that sits on top of a substrate (a) and then removed (b). An etching process is then used to remove the compacted resist and create the pattern (c).

Array Based Logic

An array of nanowires or nanotubes alone is not enough to create most electrical circuits. These nanowires or nanotubes need to be integrated with computation elements for the array to do something useful. There are several ways to integrate these devices into an array. One is to place molecule devices (section 2.3) between two wires at junction points where one wire crosses over another. Another method to create a device is to selectively dope a section of a nanowire (section 2.2) to create a field effect transistor (FET), provided another wire is placed across the FET to control conduction (this cannot be done with CNTs currently).

One design that uses both molecular switches and FETs is the nanoPLA [DeHon03b, DeHon04b, DeHon05c, DeHon05d]. NanoPLA is a programmable logic array (PLA) that uses a combination of nanowires, configurable molecular switches, and nanowire FETs (Figure 18). A PLA is a reconfigurable logic architecture that directly implements a two level sum-of-products computation. It does this with a programmable AND plane that leads to a programmable OR plane (created by NOR-NOR planes in NanoPLA). The input NOR plane in nanoPLA is accomplished by programmable diode crosspoints (OR plane in Figure 18) that perform a wired-OR, followed by an array of nanowire FETs. The nanowire FETs provide signal restoration or signal inversion of the OR-terms. The signal restoration is accomplished by precharging the buffer array output lines to V_{dd}. If the input from the OR plane is a '1' the line will stay charged, and if the input is a '0' the output nanowire will be pulled down through the evaluation transistor. The signal inversion works the same way except the line is precharged low. The inversion is needed to create the NOR function because the OR function is not universal logic. The buffer plane is needed for signal restoration (when inversion is not needed for logic) because the molecular switches in the OR plane provide no gain. The outputs from the buffer or inversion planes then go through another OR plane, and then an inversion or buffer plane (not shown in Figure 18).

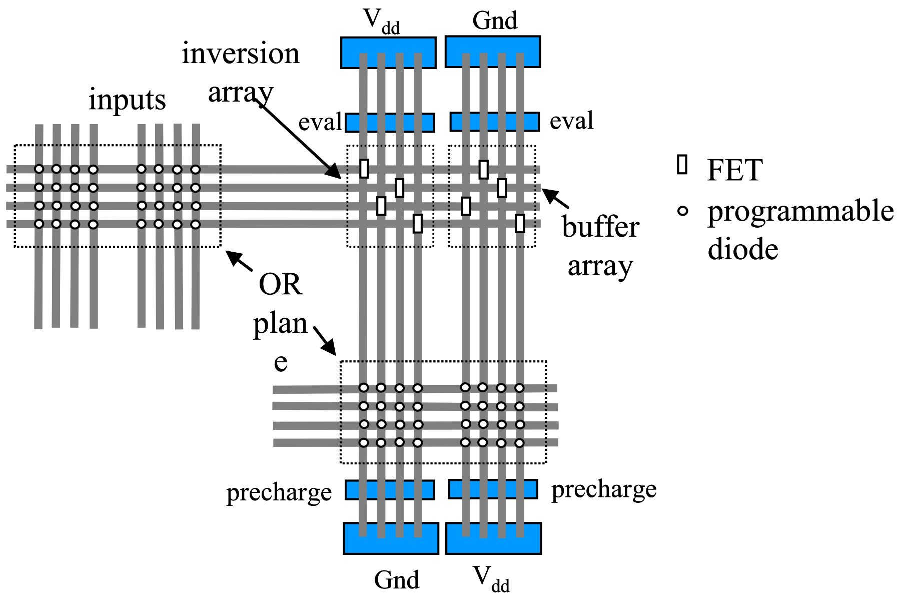

Figure 18: Organization of a nanoPLA block. [DeHon04b]

The nanoPLA uses nanowire FETs to achieve signal restoration. Another way to achieve gain is to use conventional CMOS gates, nanowire connections, and molecular devices all in a single circuit. One example, CMOL consists of an array of nanowires that is placed on top of a CMOS circuit (Figure 19) [Strukov05]. Nanoscale devices occur at each of the nanowire junctions. Notice that in Figure 19 the nanowire array is patterned at an angle to the microscale array. This angle accomplishes two things. First, this compensates for the inability to place the nanowires exactly over the interface pins to the microscale circuit. If the two grids were parallel and misaligned (which is likely), only a few pins would be connected. The angle means that a slight shift in the nanowire grid, due to assembly imperfections, allows nanowires to contact the microscale pins. The angle also compensates for the difference in pitch of the two arrays. Even if the nanowire array could be placed over the microscale grid perfectly, the pitch of the microscale wires would have to be a multiple of the nanoscale wire.

The CMOS cells can contain a variety of circuits in order to make the CMOL circuit behave as a memory or as a logic circuit. In order to harness the logic density of nano-electronics, the main tasks of the underlying CMOS elements/circuits are to address the nanowires and accompanied devices, configure the nanoscale devices, and to provide the signal restoration and inversion that cannot be done with these nanoscale devices.

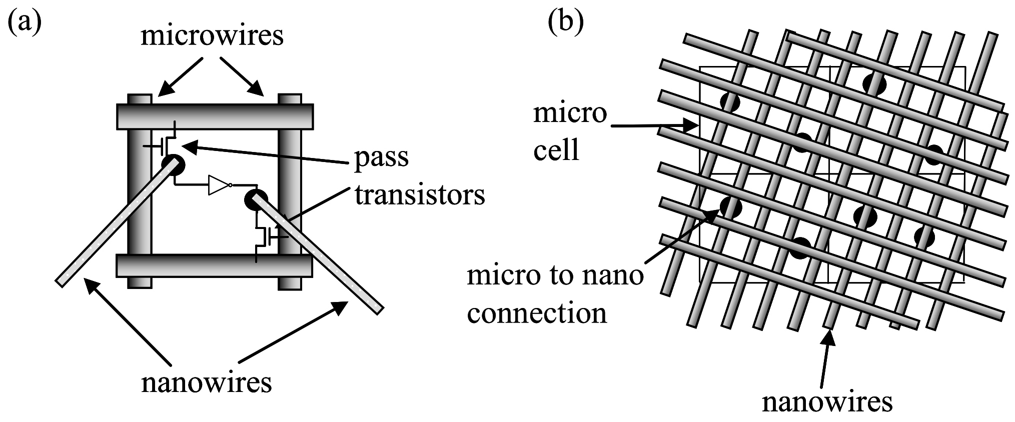

Figure 19: CMOL architecture (a) basic cell with CMOS inverter and pass gages (only two nanowires are shown for clarity) and (b) nanowire interconnect with four basic cells. [Strukov05]]

Array Based Memories

While the logic circuits proposed above pose interesting possibilities, the first viable nano-electronic devices will probably be memories. This is because memories are well suited to very regular arrays, which are the only structures to be fabricated to date (2006). In fact, HP Labs has already built a 64-bit [Chen03] and a 1-kilobit [Wu05] memory using the method shown in Figure 20. In these devices after the bottom set of wires are created using nanoimprint lithography (see Figure 17), a Langmuir-Blodgett flow process, similar to the one used to align nanowires, can be used to deposit a single layer of rotaxane molecules [Wu05]. To protect the molecules during the construction of the upper layer wires, a layer of metal is deposited over the whole chip. To create the top layer of wires, resist is deposited over the metal, followed by the imprint steps show in Figure 17. After the top wires are produced, they can be used as a mask to etch the metal layer protecting the molecular devices. Even though rotaxane molecules are deposited over the whole bottom layer, only those molecules sandwiched between two metal wires will be used.

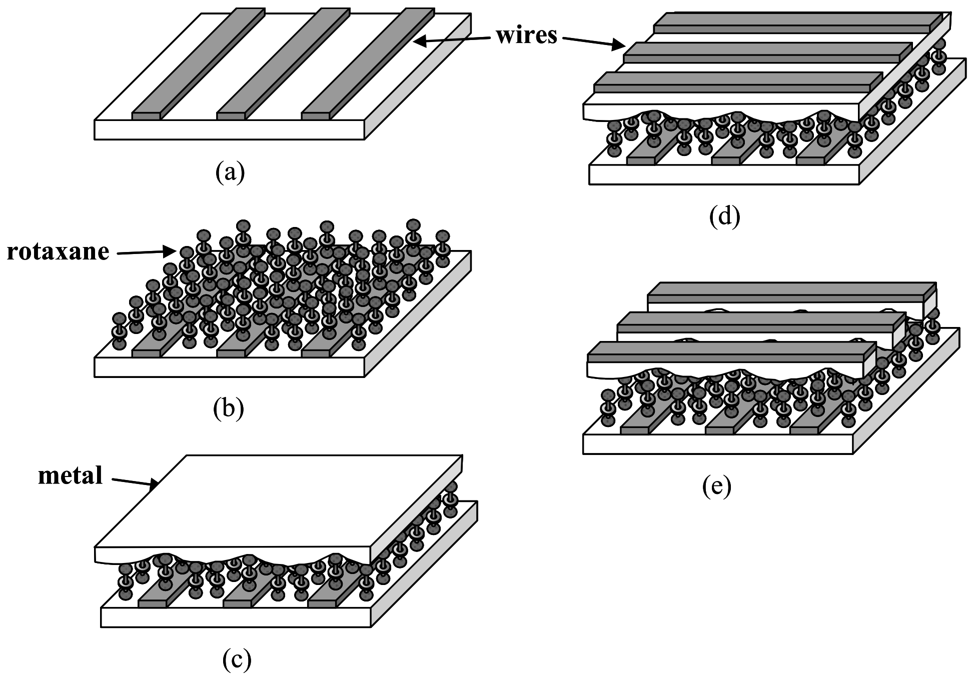

Figure 20: Fabrication process of a nano-scale memory built by HP. (a) Step one uses nanoimprint lithography to create nanowires. (b) The nanowires are then covered with a rotaxane molecule. (c) Metal is then deposited over all of the molecules to protect them. (d) Nanoimprint lithography is used again to create a set of orthogonal nanowires. (e) The nanowires are then used as a mask to etch away the protecting metal.

Another proposed memory is shown in Figure 21 [DeHon05b]. The memory contains row and column decoders that are used for memory addressing. Inside the memory array itself there is a configurable diode at each nanowire junction, similar to the HP memories discussed above. When the diode is conducting it means it holds a '1', and when it is off it means it holds a '0'. To read the memory, a high voltage is placed on a column via the column decoder. The row decoder will select the row; if current is detected, then a '1' is read, while no current means a '0'. Notice that when a column is charged, it charges all rows to which it has a '1' stored. This is why it is important that the devices are diodes. If the devices were switches that conducted in both directions, rows that are charged could then charge other columns that were not charged with the column decoder. These other charged columns could then charge the output row erroneously.

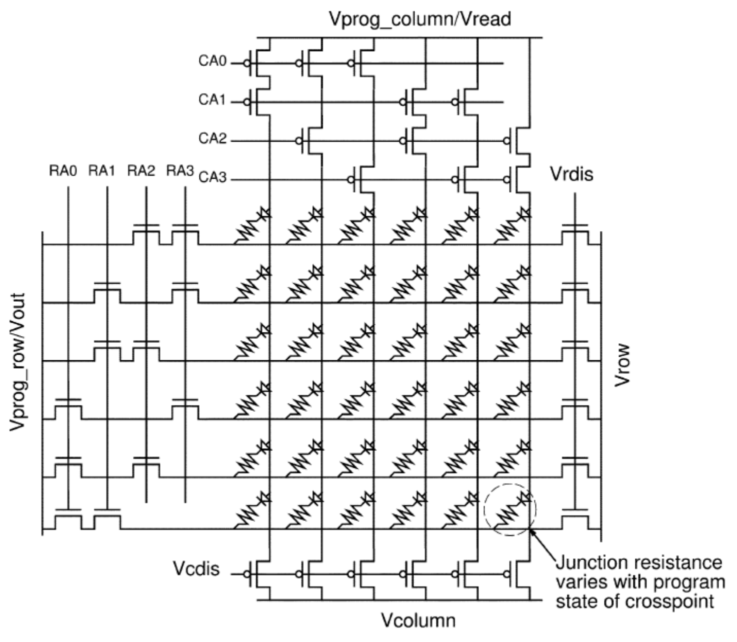

Figure 21: Circuit diagram of a nanoscale memory [DeHon05b].

When the CMOL circuit in Figure 19 is fabricated as a memory, nanoscale devices are used as the memory cell to achieve a high density, while the microscale infrastructure is used to configure the molecular switches (write operation) and to read the memory. While this has a density advantage over standard lithographic memories, since the memory bits themselves are essentially free in area, the use of lithographic wires inside the array make this device much larger than most potential nanowire-based devices. However, the simplicity makes these devices much easier to fabricate.

Nanoscale memories, if realized, will drastically change the current memory model. The memories will be very compact, because the area required to store a bit is only the cross point of two wires. This memory would also be non-volatile because the molecular switches do not lose their state without power. This means that it may be possible to have memory as fast as DRAM, but non-volatile like FLASH.

Interfacing Nano-scale to Micro-scale

For at least the first couple generations of nano-electronics, lithography-based nano-scale electronics will probably be required for things such as I/O, signal restoration and V_{dd}/Gnd distribution. The importance of this is illustrated by two memory designs discussed above. HP memories attached microscale wires directly to each nanoscale wire to access the memories. While this is adequate for proof-of-concept for the small array, this is impractical for large memory or logic devices since the microscale wires will dominate the area. Thus, there cannot be an input/output pin for each wire in the array. This problem is addressed in the memory in Figure 21 with a decoder to connect microscale wires to nanoscale wires [DeHon03a]. If microscale wires and nanoscale wires were attached end to end, the pitch of the microscale wires would set the pitch of the nanoscale wires, eliminating many of the area gains of nanoscale electronics. The alternative is to cross the two sets of wires at a right angle to one another (see Figure 22).

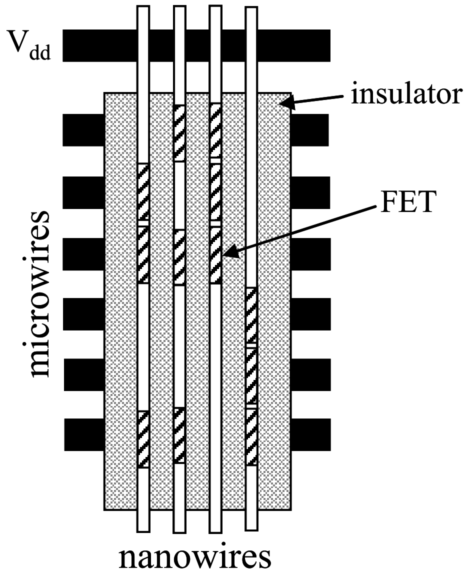

Figure 22: Decoder to interface microscale wires to nanoscale wires [DeHon03a].

DeHon's proposed decoder uses microscale wires to control the FETs in the nanowires. The nanowires are coded with a (n/2)-hot scheme, where n is the number of possible controllable regions (number of microscale wires). This means that each nanowire has (n/2) FETs that are controllable, and there can be $\binom{n}{n/2}$ uniquely coded nanowires. For example, in Figure 22 n equals six, so each nanowire has three controllable p-type FETs, and there can be 20 unique codes. In order to make a nanowire conduct, 0's are driven over each FET of that wire, while 1's are driven on all other (n/2) wires to stop all other nanowires from conducting. There are many issues that make this decoder complicated. One concern is that there is no way to control how the FETs of a nanowire will line up with the microscale wires. In fact, there is no assurance that the FETs will line up with the microscale wires at all - they may all end up above or below the microscale wires. To address this, the codes (sequence of FETs) are repeated over the length of the nanowire. By placing many copies of the code along the length of the nanowire, a shift up or down will simply place different FETs over the microwires. This works because if a FET is uncontrolled (not covered by a microscale wire) it will always conduct. The uncontrolled FET conducts because the nanowires are made of all p-type or n-type semiconductor, and the controllable regions are also the same type (p-type or n-type), but with a lower doping level, so they are easier to deplete than the rest of the wire.

A second concern with the decoder is that there is no way to individually place specifically coded nanowires on the decoder. This is because all of the nanowires with all of the needed codes are placed in a solution which is used in a Langmuir-Blodgett flow to deposit the nanowire over the microwires. Unfortunately, it is impossible to create a solution that contains exactly one wire of each needed code, and even it that was possible, the inability to control vertical alignment would essentially change codes. For example, the two nanowires on the right side in Figure 22 have the same code, but because they are not "placed" in the same location, they have different addresses. If a solution with many copies of the needed codes was created, the likelihood that multiple copies of the same code being deposited on the decoder is very high. To overcome this uncertainty, DeHon proposes using more codes than are required to address the set of nanowires [DeHon03a]. With enough different codes, and many copies of each code, the required number of nanowires needed for the decoder will be randomly "selected" with a high probability that each wire's code will be unique. This means that not all of the addresses will be used in each decoder, and there will have to be more microscale wires than would be required if an ideal decoder could be created. DeHon showed that $\lceil 2.2\log_2(N) \rceil + 11$ address (micro) wires are needed to address N nanowires with a 99% chance that all nanowires will have a unique code. This represents a substantial overhead over an ideal decoder. For example, if the decoder had 20 microwires, it could address 184,756 nanowires using and ideal (n/2)-hot scheme decoder. To address this many nanowires with the non-ideal decoder would require 50 microwires. Finally, since not all addresses will be present in each decoder, the working addresses must be discovered before the decoder can be used.

An alternative method was proposed to randomly scatter gold nanoparticles over the decoder region [Williams01]. Ideally, half of the possible junctions will get a nanoparticle deposited on them to create connections between the nanowires and the microscale wires. This is similar to the (n/2)-hot scheme used in the decoder discussed previously. The main difference is that the gold particles act as a resistor instead of a FET. This means that the wire to be selected will have all contacts conducting, and the output will be at full voltage. Other lines will have some 1's and some 0's, making a voltage divider circuit, so they will be at an intermediate voltage. This is acceptable as long as the two voltages can be differentiated. With this method, addresses will also have to be discovered in the same manner as the decoder discussed above.

A problem with both of these decoders is that the addresses are stochastic. In an ideal memory, addresses are sequential and all possible addresses work. With the above proposed decoders, only a small subset of possible addresses work, and the working addresses are randomly placed in the address space. This means that 2^n addresses need to be checked for functionality, since any combination of connections is possible. Also, once a working memory is discovered, it must be stored in order to be subsequently used. To deal with storage of working addresses, an addition to the decoder has been proposed to generate deterministic addresses [DeHon05e]. If programmable FETs are added to the decoder, stochastic addresses can be converted to deterministic addresses (see Figure 23). To discover working stochastic addresses, the programmer starts at address zero (on microwires a-d in Figure 23) and increases the address until one that is working is found. Working means that one of the microwires is charged up. When a working stochastic address is discovered through the stochastic decoder, it is assigned a deterministic or known address by configuring the programmable FETs. The FETs are programmed to be either controlling or not controlling, using a (n/2)-hot scheme again. For example, in

Figure 23, if the nanowire was p-type, when the address on micorwires a-d is 0101, the nanowire will be discovered to conduct. If the next deterministic address to be programmed is 0011, the programmable FETs over microwires 1 and 2 will be configured as controlling while microwires 3 and 4 will be configured non-controlling. Now only the address 0011 will allow the nanowire to conduct because any other address in a (n/2)-hot scheme will have a '1' on microwire 1 or 2. Once the FETs are programmed, only the deterministic addresses are used and the controlling wires for the stochastic decoder are set such that the nanowires conduct through that region. This converts addresses that are randomly placed in the address space into know addresses. Two suitable technologies for the programmable FETs have been proposed [Salvo01, Huang01b].

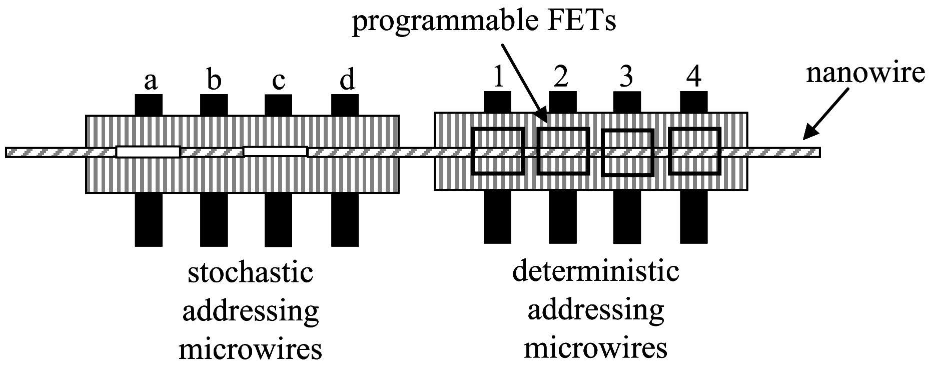
programmable FETs
a
b
c
d
1
2
3
4
nanowire
stochastic addressing microwires
deterministic addressing microwires

Figure 23: Programmable address line to create a deterministic address decoder. The nanowire corresponds to a nanowire in Figure 22. [DeHon05e]

Another proposed decoder also uses a (n/2)-hot scheme, but approaches the need for deterministic addressing and fault tolerance very differently [Snider05c]. The decoder in Figure 24 uses a combination of a CMOS address encoder and an nanowire crossbars to "turn on" a single nanowire with a deterministic address. The deterministic addresses are expanded to a sparse, redundant code by the address encoder. The sparse code has three parameters that can be tuned to change the number of output lines and the ability to tolerate errors. These parameters are the code length, the weight, and Hamming distance (6, 3, and 4 respectively in Figure 24). The code length is the number of lines coming out of the address encoder. The weight of the code is the number of 1's in the code. Each output of the encoder always has the same weight. The Hamming distance is the number of bit flips required to go from one address to another. This limits the number of possible addresses, but it also is the mechanism for fault tolerance. Notice in Figure 24 that if the decoder is defect-free then one output nanowire is at full voltage, while the rest are at some lesser voltage. Assuming only stuck-open defects, a fault in a molecular switch or wire for the targeted line will still result in a full voltage output. If the fault is on a 0 V line for an output line that is not intended to be charged, the voltage will increase, but there is still enough of a difference between the two output lines to determine which line is being selected as it takes multiple faults to create an error.

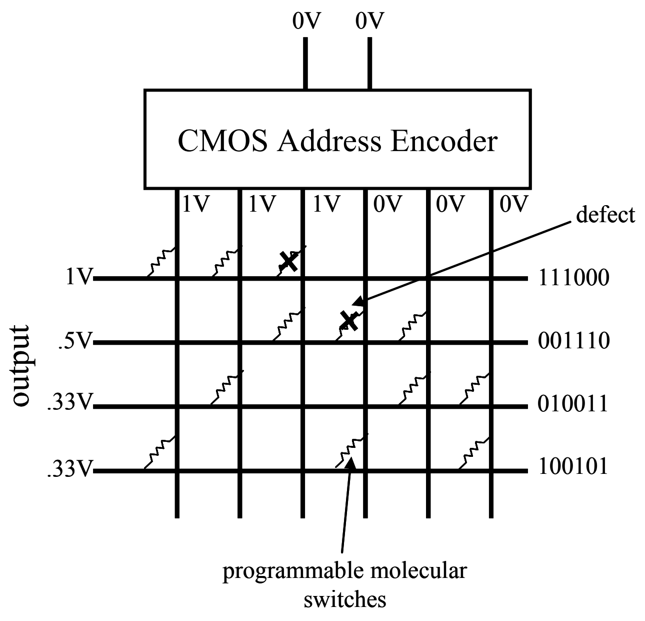

Figure 24: Micro-scale wire to nano-scale wire decoder [Snider05c]. The 2-bit input address is expanded by the address encoder to a redundant 6-bit address that selects one output nanowire.

Although in Figure 24 it appears that it would be better to just connect the output lines directly to the CMOS circuits, there are several reasons that this is not the case. One is that many more output lines can be controlled as the number of address lines out of the address encoder increases. For example, 237 output lines can be controlled with only 22 address lines [Snider05c]. Another advantage is that if one of the address nanowires is broken, this scheme still works. This decoder does have some pieces that will not be easy to fabricate. One difficult part would be the need to place the programmable molecular switches at certain nanowire-nanowire junctions. Current techniques for depositing molecules, such as the Langmuir-Blodgett flow, distribute a layer of molecules, and can't place individual molecules. There is also no discussion of how the vertical nanowires would be connected to the CMOS address encoder.

Each of these three encoders has their strengths and weaknesses. While the first decoder has a digital output (ideally only one output has any voltage), it has many different parts, which may be difficult to achieve in nano-electronics. The last two are simpler approaches, but they have outputs that are controlled by voltage divider circuits. This means that the connecting circuitry has to be designed to handle intermediate voltage values properly. The last two decoders also have the advantage of not having to use high impedance semiconducting nanowires. Finally, the last decoder does not require the discovery of addresses, as the first two decoders do.

3.2 Random structures

In contrast to the regular structure of an array, there are a few architectures that are much more random. One example, Nanocell [Husband03] is a block that has a random network of molecules that act as negative differential resistors (NDR) (see Figure 25). The network is created by randomly depositing very small conductive particles (nanoparticles) of gold or platinum onto a substrate. The NDRs are then put onto this substrate, and the ends of each molecule attach to a nanoparticle, creating a random network. The NDRs are put in a random state (on/off), and a genetic algorithm is used to program the device. The genetic algorithm starts with several nanocells with random configurations of NDRs, some of which are 'on' and some that are 'off'. The configuration of each nanocell is encoded as a binary string that represents the state of each switch in the cell, with '1' indicating the switch is 'on', and '0' indicating it is 'off'. The two fittest "parent" cells are then selected to "mate" by recombining their binary strings. A cell is more fit than another if it behaves more like the desired function. The recombining is done by uniform crossover, which goes though both parents' binary strings one bit at a time and determines with a coin toss which of the two offspring get which parent's bit. The offspring then replace the parents in the population. This continues until a cell functions as desired (OR, NAND, etc).

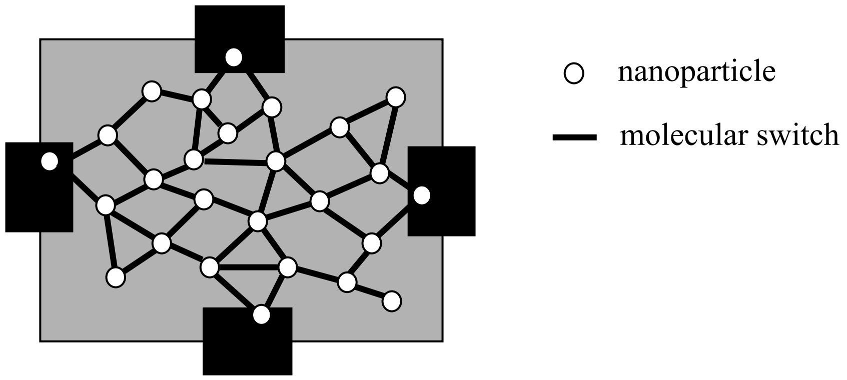

Figure 25: Diagram of a nanocell block. The black boxes on the periphery are microscale I/O terminals for connecting to other blocks [Husband03]

This network functions as a current device, meaning that a cell functions correctly by having a current level below a certain threshold for a '0' and higher than a certain threshold for a '1'. For example, an OR function would produce "high" current levels when a '1' is placed on any of the inputs, and a "low" current when a '0' is placed on the inputs. With omnipresent control over switches (able to configure switches individually without using I/O pins), an inverter, NAND and a 1-bit adder were able to be programmed. The intent is to configure a block to perform a certain function such as AND or NOR and then tile the blocks together to implement a circuit.

While random architectures are interesting because they have inherent fault tolerance and it harnesses the random nature of nanotechnology instead of trying to create some order with it, this and other current example have drawbacks. One obvious disadvantage is the amount of resources required to implement a small circuit such as a NAND gate. Also, because each cell needs to be configured separately, the circuit would probably not be reconfigurable once it is built. The configurations so far have all been done in simulation, so it will be interesting to see if the programming of the switches can be accomplished from the I/O blocks, which would be required in a real system. Scalability will also be a challenge for this design because it is a current output device. Blocks cannot be simply placed together because the signal will be degraded to useless levels in just a few blocks.

4. Fault Tolerance

Even though nano-electronics device fabrication is in its infancy, it is clear that defect and fault levels will be much higher than current CMOS technology. The exact level of defect densities is unknown, but it is assumed that 1-15% of the resources on a chip (wires, switches FETs, ect.) will be defective [Chen03, Huang01a, Misha03]. These high number of defects are largely a consequence of how extremely small the devices will be. Although their size makes nano-electronics an attractive successor to VLSI, it also makes reliable manufacturing difficult for three main reasons: stochastic assembly, fragility due to small numbers of atoms, and less random skew tolerance. First, the ability to deterministically place all of the parts of a circuit will likely no longer be possible for nano-electronics devices. This means that stochastic assembly will be necessary. The use of a stochastic process means that things such as the proper alignment of wires for connections over an active FET, or the population of molecular switches, cannot be guaranteed. The second issue resulting from the use of devices that will be a few atoms in at least one dimension is that these devices will now be much more fragile. They will be susceptible to wires breaking during manufacturing and during their lifetime. They are also expected to be less tolerant to radiation, high temperatures, and electromagnetic interference. Finally, as pointed out in [DeHon04a], these devices will no longer be able to rely on the Law of Large Numbers at the device level. In current MOSFETs, even though electrons behave randomly, statistics tell us what the majority of electrons will be doing at any moment, and thus we can be assured of a certain amount of current flow. This will no longer be a proper assumption when dealing with only a few electrons, since only a few "stray" electrons could drastically change the timing of a gate. We can probably only rely on the Law of Large Numbers above the device level for nano-electronics.

Before we review the many proposals on how to handle faults and defects in nano-electronics, these two terms should be defined. "Defects" include both manufacturing and post-manufacturing errors that are permanent, such as broken wires or missing switches. These permanently change the behavior of the device. "Faults" are errors in computations as a result of either defects or transient faults. "Transient faults" occur due to events such as electromagnetic noise or radiation, and are not always reproducible. This distinction is important because some remediation techniques only handle faults due to defects, while some can remedy any fault.

With defect rates for current VLSI processes in the range of 1 part per billion [Stan03], manufacturers can afford to discard any chip that is found to be defective. Some FPGA and memory manufacturers add a small amount of redundancy that can replace a small amount of defective devices to increase yield. But this strategy still relies on very small defect rates. With defect rates of around 1-15% of all individual transistors and wires, neither of these techniques will prove effective for nano-electronics. Because reliability will have to be handled at the architecture level instead of the device level, what was once a process engineering problem will now become an architectural dilemma. Novel techniques and architectures will have to be devised in order for nano-electronics to become a viable replacement for current VLSI processes. Current research on fault tolerance has followed three tracks to solve this problem. These are configuring around the defects, masking faults with redundancy or designs that are inherently fault tolerant.

4.1 Configure around defects

The first track relies on reconfiguring the device around the faults. Since nano-electronics devices will generally have to be built with a bottom up approach, they will likely have to be homogeneous at some level. Since heterogeneous components such as ALUs and signal processing units may be impossible to build, configuration of the homogenous structure will have to be employed to realize such devices. This configuration process can be leveraged to map around the defective parts of a chip. For example, consider the Teramac configurable computing machine [Amerson95]. Teramac is a configurable hardware system with 1728 FPGAs built by Hewlett-Packard Laboratories in 1995. To keep costs down, the designers of Teramac allowed for some of the devices to be defective. Despite having a defect rate of almost 3%, Teramac was able to function properly by configuring around the defects.

Defect discovery

Before one can configure around the faults, they must first be discovered. Since nano-electronic researchers expect to be able to integrate 10^{12} devices on a single chip [Huang04], this will not be a trivial task. Even though the number of devices for Teramac is six orders of magnitude less, the developers still faced many of the same issues [Culbertson 97]. These included not being able to probe each individual device and not knowing a priori which devices are functioning. Each individual gate cannot be economically tested because of the low bandwidth for chip I/O and the large number of devices. Also, since the testing circuitry is made up of the same defective fabric, there is no assurance that the testing circuitry is fault-free. To handle this issue, devices are tested in clusters (see Figure 26) by configuring a signal generator into the units. If the output pattern is correct, all of the devices in that test are deemed working. Otherwise, all of the devices are assumed defective until another test says otherwise. Each device is tested many times with separate devices to increase the probability that a good device, will eventually be tested with a set of other good devices and the bad devices will be isolated. To facilitate faster testing, the testing on Teramac was bootstrapped. That is, the system tests one part of the machine, maps its defects, and configures that part to test its neighbors. Repeatedly applying this process eventually tests the whole machine.

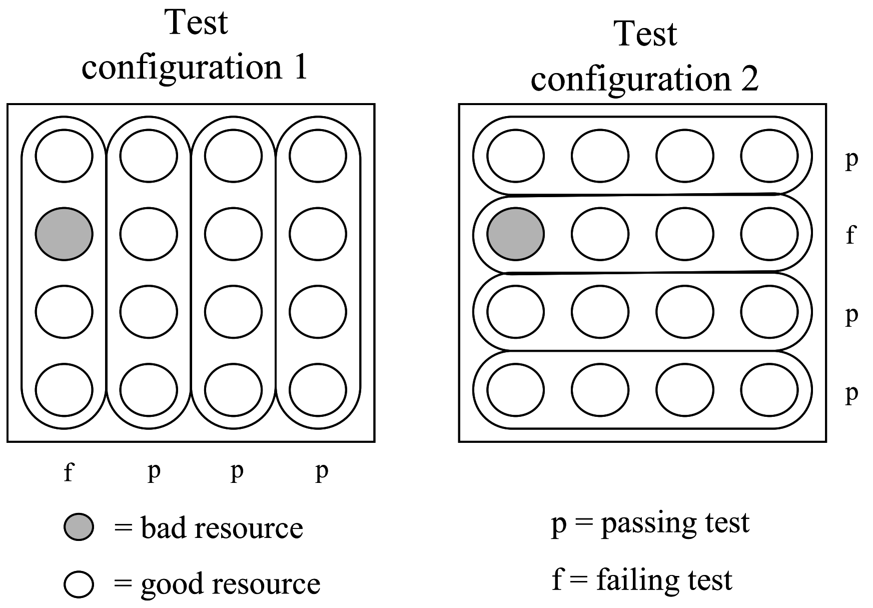

Figure 26: Resources are grouped together to be tested in the Teramac computer [Culbertson 97]. Multiple orthogonal tests are run, and the intersection of failing tests with common resources points out bad resources. Note that this is just an illustration; many tests are run to isolate each defect.

Unfortunately, this scheme will not translate directly to nano-electronics [Brown 04]. One issue is the large number of devices foreseen. A larger issue is the greater predicted defect density for nano-electronics. The problem is that when there are so many defective devices, a good device has a low chance of ever being paired with a set of only good devices. A simple solution to this would be to group fewer devices together for the test, but this increases the testing time to an unacceptable level. A solution to the problem of too many devices and defects is to first determine the probability of a device being defective, and not testing devices with a high probability of being defective [Mishra03, Goldstein03]. Instead of running enough test vectors to isolate defective devices, a smaller set of test vector returns either none, some, or many faults. To accomplish this, a block is configured into a LFSR. If the LFSR returns the correct signature, then the result is none. If the signature is incorrect, then the block is broken up into smaller LFSRs and rerun. If more than half of the smaller LFSRs are defective, then many defects is the result. Otherwise, the result is some. Once this first step is completed, each device is analyzed to determine how many circuits it was in and how many faults were detected in these circuits. Bayesian analysis is then used to determine the probability that a device is defective. All of the devices with a high probability of being defective are discarded. The defect rate of the remaining devices should now be low enough to do defect detection similar to Teramac.

Another method to detect the faults eliminates the need to store the fault map externally [Brown 04]. This is a significant advantage given the number of devices predicted to be integrated on a chip and the time required to get the fault map off chip. Built in self test (BIST) is a classic VLSI testing technique that uses dedicated on-chip testing circuitry to test the chip, saving time by removing the I/O overhead of external testers. Of course, the testing circuit must be defect-free, so BIST will not directly work with nano-electronics. However, the reconfigurable nature of the architecture can be leveraged to facilitate the testing. Testing of the NanoFabric architecture (section 3.1) involves a modified BIST algorithm that uses the reconfigurable blocks to do the testing. This is accomplished by first testing block 1 in Figure 27 with and external tester.

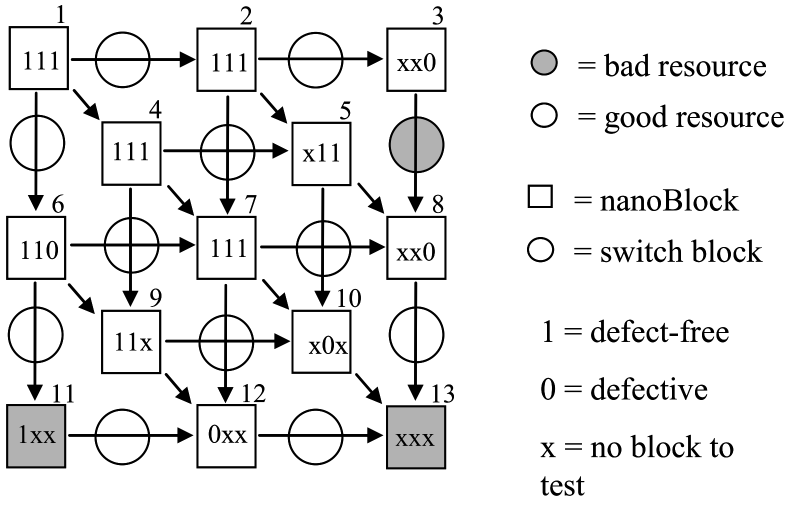

Figure 27: NanoBlocks are tested by each other in groups of threes [Brown 04]. An arrow indicates one block testing another. Three bits in each block are for its three neighbors in the order of right, diagonal and down.

Block 1 is then configured from an external source to test blocks 2, 4 and 6. Blocks 2, 4 and 6 are all given the same test vector, and the blocks under test are configured so that the output patterns are identical to the input patterns. This means that each testing block only has to compare the output to the input test vector to see if they are identical. Whether a nanoblock is capable of implementing a k-bit comparator (where k is the number of vertical or horizontal wires) is not clear [Wang 05]. Block 1 then stores one bit for each block it tested, indicating whether it has a fault or not. This testing continues in a wave diagonally across the chip until all blocks are tested. This scheme will also indirectly test the switch blocks and wires. Notice in Figure 27 that block 8 is tested by blocks 3, 5 and 7. Block 3 indicates block 8 is defective while block 5 and 7 indicate that block 8 is defect-free. This indicates that the resources between blocks 3 and 8 are defective. All blocks are considered defective until proven otherwise. This could lead to underutilized blocks. For example, if block 6 is a defective block, 11 must be assumed defective since block 11 is only tested by block 6 and therefore cannot be proven otherwise. Because the configuration stream always goes through blocks that have been tested, the defect map does not need to be exported to the external tester. A block that has a defective neighbor configures the switch block not to route to the defective block. It was shown that more than 99% of the defect-free blocks could be identified if such a "scan" was performed from each of the four corners.

It may also be possible to use BIST to test the entire fabric in parallel instead of the wavelike pattern proposed above [Wang 05]. Blocks are grouped together in clusters of three with one block being the test pattern generator (TPG), one being the block under test (BUT), and the last being the output response analyzer (ORA). The clusters are regrouped three times such that each block is configured as a TPG, BUT, and ORA so that a BUT is not marked as defective because the TPG or the ORA is faulty. The testing is set up to test for stuck-at faults, so that the defect-free response from the BUT is all 1's or 0's, which can be checked with a simple AND or OR gate (see Figure 28). Testing could possibly be done quickly if a chip could configure itself, but how this would be done is not clear since operating voltages are much less than programming voltages.

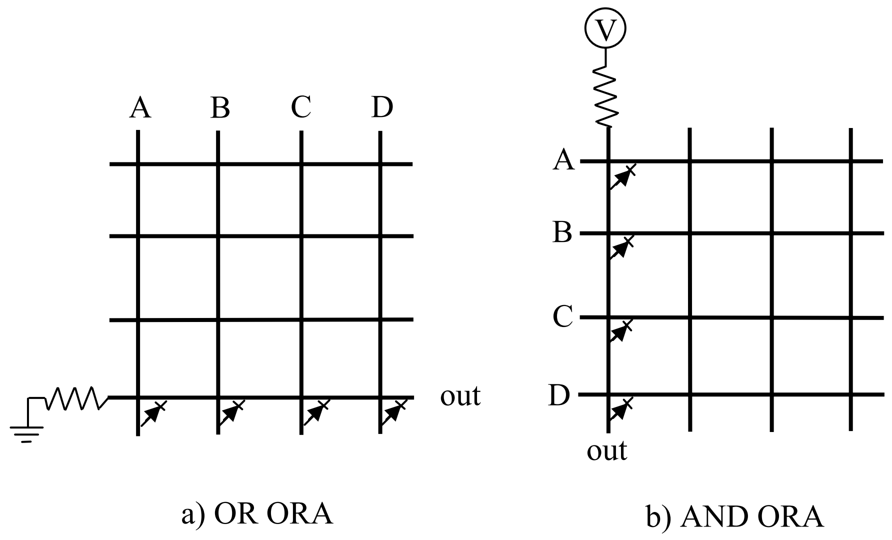

Figure 28: Configuration of a nanoBlock to perform an AND and OR output response analyzer (ORA) for a built in self test of nanoFabric [Wang 05].

For the nanoPLA [DeHon03a, DeHon04, DeHon05a] and nanoscale memory [DeHon05b], the working resources are discovered instead of the defective resources [DeHon03a]. Because the decoder used to address nanowires from microwires (section 3.1) is stochastically built, it is not known a priori which addresses are working. All possible addresses must be tested to see if they can configure a cross point. With the memory in Figure 21, first the row output microwires are all turned on so that if any column is energized by a particular address it will be detected. Once the column addresses are discovered, then the row addresses can be discovered with the known column addresses. This is accomplished by first configuring all of the crosspoint junctions for a known column address to be on. Then all of the output or row addresses can be read. Any address that produces a high output is a working address. Note that every row address has to be checked every known column address as it can't be assumed that all crosspoints are functional. An address may not function because it does not exist or the nanowire it addresses has a fault somewhere (broken wire, missing switch, etc.). One drawback for stochastic addressing is that the working addresses may have to be stored. For the nanoscale memory, this will require O(N log(N)) bits of storage for a memory that can store O(N^2) bits. A monolithic memory to store the table of working addresses is not compact enough to store in lithographic memories, so future work is needed to develop multistage nanoscale address mapping architectures.

An algorithm to detect the faults in nano-electronics is going to be dependent on the architecture, but there will probably be some common characteristics among all of them. The first characteristic is the use of an external tester or a reliable CMOS circuit to do the initial testing [Culbertson97]. This is done because none of the resources in a nanoelectric device can be assumed to be defect-free. Once some portion of the chip is found to be defect-free, the testing will need to be bootstrapped. That is, some portions of the chip that are determined to be defect-free are used to test the rest of the chip. This is essential to speed up the testing so that the large number of devices can be tested in a reasonable amount of time. One large issue remaining to be thoroughly addressed is the vast amount of data a defect map will contain, and how to efficiently handle it. If a chip has 10^{12} devices on it with a defect rate of 10%, then there will be 10^{11} defects to map.

Mapping around defects

When defects are discovered, the defect map must be used to avoid defects on the chip in order to create a functioning circuit. This will likely add a step to the CAD flow for configuring the device. Considering that 10^{11} defective components have to be handled, this will not be a trivial problem. One simple solution is to simply remove the defective device from the list of available resources, just as if it was pre-assigned to logic in the placement phase [Amerson95]. This solution worked for the Teramac computer because of the sparseness of the defects. When a LUT (computing block) was determined to contain a defect, it could be thrown out because there were many other LUTs that were defect-free. It will probably be impossible to build a defect-free LUT with nano-electronics, so a solution will have to be found that maps around defects at a fine-grain level.

To map around faults in a crossbar-based architecture one can use a technique based on a bipartite graph B = (U,V,E) [Huang04], where the vertices U and V represent the input and output nanowires respectively, and the edges E represent the programmable switches between the wires (see Figure 29). The input vertices make up one partition of the graph, while the output vertices make up the other partition. A maximum flow algorithm can be used to find a maximum matching of the inputs to outputs, but first the graph must be modified to incorporate the errors. Stuck open faults, where a switch cannot be programmed to close, means that the affected edge is removed from the graph. For stuck closed faults, where the switch cannot be configured to be open, all of the affected resources are removed from the graph. However, it should first be noted that the removed group can be used to route one signal. For nanowire break faults, the corresponding vertices are simply removed from the graph (notice in Figure 29 that it appears that i4 can still connect to o1 and o2, but the authors are assuming that the wires will be precharged, so all of i4 is defective). For nanowire bridging faults, where two nanowires are shorted together, all but one of the affected wires is removed from the graph. One thing to note is that some of the errors such as bridge and stuck closed faults might not be useable because they are not a reliable connection. In this case all of the affected resources would need to be removed from the graph. Once the graph is constructed, it can be used to represent the crossbar to configure around the defects.

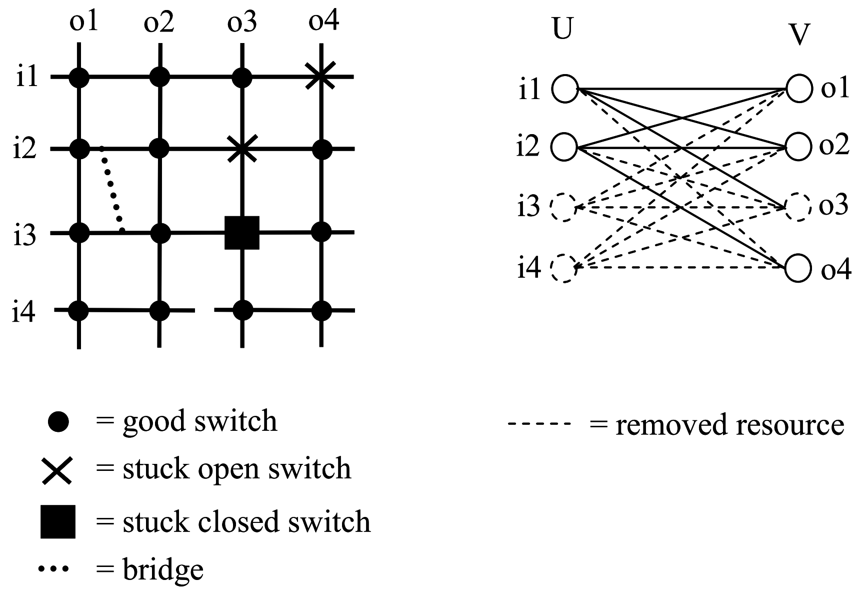

Figure 29: How a bipartite graph is built with defects to facilitate mapping circuits to a defective crossbar architecture.

Another use of the bipartite graph is to construct the graph based on whether a certain output wire will support a certain function given a set of faults in a crossbar architecture [Naeimi04, DeHon05a]. Given a set of product terms (Figure 30a) and a crossbar with all defects mapped (Figure 30b), a bipartite graph can be constructed that represents all possible assignments of functions to output wires (o_1-o_4) (Figure 30c). Once the graph has been constructed, a maximum flow algorithm can be used to come up with a solution (Figure 30d).

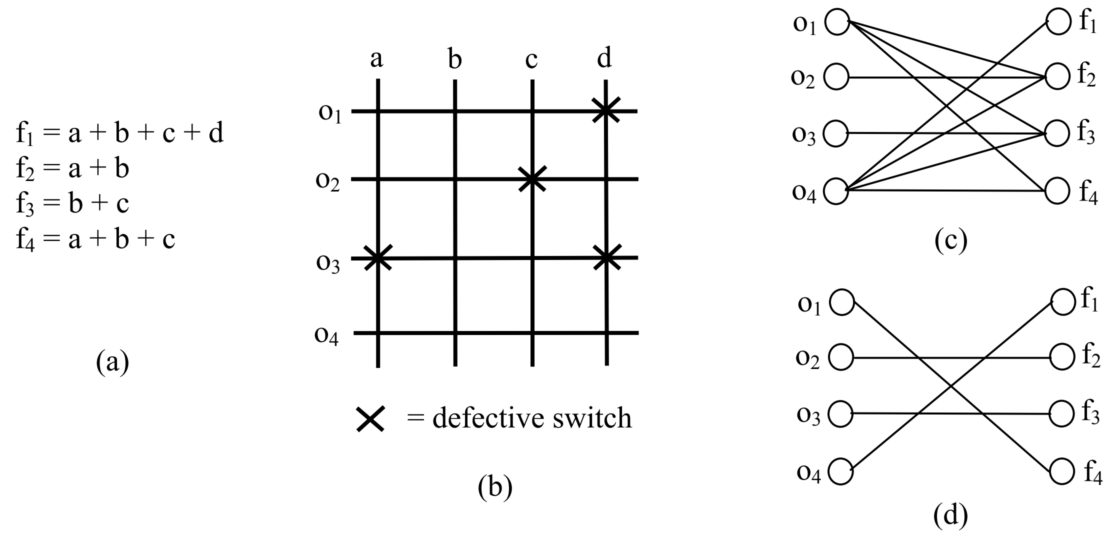

Figure 30: (a) OR term functions to be mapped to crossbar. (b) Crossbar with defective switch points. (c) Bipartite graph of how functions can map to output wires. (d) Possible assignment. [Naeimi 04].

One problem with this solution for nano-electronics is that the large number of resources on a chip will make constructing this graph a very time consuming process that will require an enormous amount of data. However, if a greedy algorithm is used instead of a maximum flow algorithm, then the graph does not have to be built [Naeimi 04]. The greedy algorithm is a two-step process. Conceptually, step one is to select the function that has the least number of options for output lines to use (f_1 in Figure 30a). This is represented in the graph (Figure 30c) as the node f_i with the least number of edges or least degree. Then select a random output node (o_i), until one that has an edge to the function is found. To avoid building the graph, the first step can be accomplished by noticing that the functions that have the least degree have the most inputs. Also, without a graph it is not possible to check for an edge between a function and a random output wire. This is circumvented by selecting a random output wire and only testing the cross points that will be needed for the given function.

Since it appears that nano-electronics will be assembled with homogeneous structure and configured post-fabrication to perform a given circuit, it seems logical that the reconfigurable nature should be leveraged to handle the defects. Even though reconfiguring around defects has been shown to give the greatest chip reliability for a given redundancy level [Nikolic 02], there are some deficiencies intrinsic to this method. The main drawback is that the fault map is static. This means that it cannot handle transient faults because they often will not show up during defect testing. Defects that arise during use will also cause errors until the faults are remapped and the device reconfigured. One other drawback is that the configuration stream for each device must be unique due to the unique nature of a chip's fault map. This means that it will no longer be possible for a design to be compiled once and shipped out. Designs will have to be shipped and compiled by the end user after they have a fault map for each device.

4.2 Run time fault tolerance

There are some fault tolerance techniques that can handle both defects and transient faults. This is accomplished mostly through hardware redundancy. This approach means that defects are characterized on a statistical level instead of specifically mapped. In other words, for a given technique, it is known that a given percentage of faults can be mitigated whenever they may occur. In the 1950's computers were unreliable in large part because the valves that made up the switches were prone to burning out. This prompted John von Neumann to study the task of building reliable computers with unreliable components [von Neumann 56]. Specifically, von Neumann developed two hardware redundancy schemes, NMR and multiplexing, to achieve reliability. With the invention of the silicon transistor and the subsequent advances in manufacturing, these two techniques were somewhat forgotten. However, they are still used in very critical hardware that cannot afford an upset due to radiation or some other random event. Now that it is evident that nanotechnology will also be unreliable, these ideas have received renewed interest.

Triple Modular Redundancy

Triple modular redundancy (TMR) is a technique where each computational unit is duplicated three times, run in parallel, and the outputs are "voted" on with dedicated circuitry to determine the correct output (see Figure 31). The idea is that at least two of the outputs will be correct, and that one may be incorrect. The voting circuit will pick the two correct outputs, essentially ignoring any errors. NMR is a general form of TMR where there are N copies (N must be odd to avoid ties) instead of three.

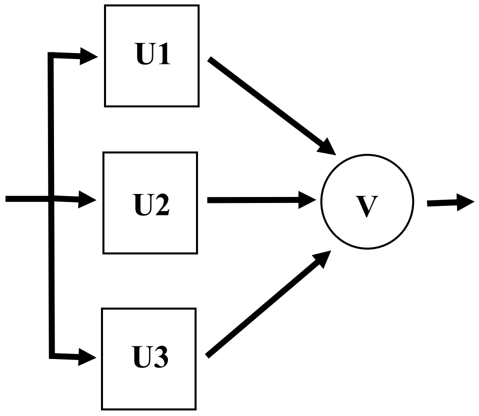

Figure 31: Triple module Redundancy with three copies of logic (U1-U3) and a voting circuit (V) [Graham04].

TMR is often used in current VLSI designs to assure correct circuit functionality when errors have drastic consequences or when circuits will be subject to high levels of radiation. Current implementations guard against transient faults such as those caused by radiation, rather than against defects since defective chips are discarded. There are two issues with this technique. The first issue is that TMR assumes that at most one of the three (or (N/2 –1) for NMR) will contain a fault. If more units contain faults, then the faulty response may appear to be the correct response to the voting circuits. For nanotechnology, where fault levels will be quite high, NMR would have to be done at a very fine grain level or with lots of redundancy (large N) to assure a high probability that less than half of the logic units will be defect-free. This means that NMR will probably not be a solution to whole chip fault tolerance. In fact, it has been shown that in order to get a 90% probability of a chip with 10^{12} devices working with NMR, that even with a defect probability (probability of an individual device being defective) of 10^{-7}, that N would have to be 1000 [Nikolić02]. A defect probability of 10^{-9} is about what is being achieved with the state-of-the-art (2006) VLSI processes [ITRS05]. The second issue is the requirement that the voting circuit must be fault free. As discussed previously, there should not be any single component in nano-electronics that must be fault-free in order for a chip to be functional.

Von Neumann multiplexing

Von Neumann multiplexing (see Figure 32) is very similar to NMR, but it removes the reliance on a single voting circuit. As in NMR, the logic units and inputs are duplicated

N times, but instead of voting on the outputs to produce a single output, all of the outputs are kept. If all of the inputs are correct and all of the logic gates are defect-free, then all of the outputs will be correct. However, if some of the inputs are incorrect (errors from a previous stage) or there are faults in the logic, some of the outputs will be incorrect. The outputs are considered true if $(1-\Delta)N$ or more signals are true, and false if ΔN or less signals are false, where Δ is a predetermined threshold $(0 < \Delta < .5)$. If the distribution of the outputs lie between these two values then the circuit is considered to have malfunctioned. Figure 32 shows that there are two stages in a multiplexing circuit: the execution and restoration stages. The execution stage is where the desired function is performed. The purpose of the restoration stage is to correct any "degradation" of signal caused by the execution stage. A degraded signal is when the output bundle from the execution unit contains more errors than any of the input signals. An example will show how a multiplexer can resolve faults.

Von Neumann gave examples using NAND and majority gates (this is why this technique is often called NAND or majority multiplexing). A majority gates example is shown in Figure 32. If N = 5 and Δ=1/5 then at least four lines need to be true for a bundle of lines to be considered true. The inputs come in as bundles that consist of the same signal duplicated N times. In Figure 32 two of the input bundles have only four of the five lines stimulated (logic '1') due to an error in a prior stage, and the other bundle has no stimulated lines (all 0's). The output of the execution stage has only three stimulated lines because the random permutation unit (block U in Figure 32) groups inputs to the majority gates (block M in Figure 32) so each majority gate has a random input from each of the three input bundles. In this case, the execution stage degraded the signal because it outputs two 0's and only three 1's. If you follow the signals through the restoration unit, and again assume that the "random" permutation unit does not pair wires from the same bundle together or create the same permutation for more than one majority block (ie. pairing the output from the first three execution majority blocks into the first two restoration majority blocks), then four of the five lines will be stimulated and sent on to the next stage as a "true" bundle. It is also possible for a given permutation to output a "false" bundle, but This scheme can be done with NAND or NOR gates in place of the majority gates in Figure 32, except two restoration stages are needed because of the inverting nature of NAND and NOR. Since NAND and NOR are universal logic gates, any circuit could be transformed to be multiplexed. One issue with this scheme is what happens to the final output. Conceivably, the bundled signals could be used throughout the whole circuit and external circuitry such as CMOS could "vote" on the output.

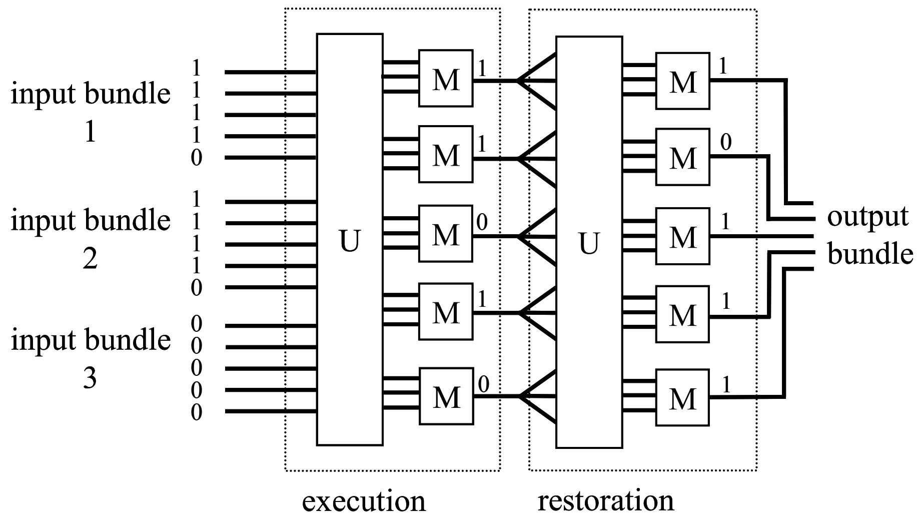

Figure 32: Majority multiplexing example that shows the function of the restoration stage. Block U is a random permutations unit and block M is a majority gate.

It has been shown that NAND multiplexing could be a viable fault tolerance technique for nano-electronics, but only at high levels of redundancy [Roy05, Bhaduri04, Han02, Norman04, Han05, Qi05]. Simulations have determined that with a fault rate of 10^{-3} per device, a redundancy level (number of times circuit is replicated) 10^5 is need to assure 90% of the chips will work correctly [Nikolić02]. This means that a chip with 10^{12} devices will actually function as a chip with 10^7 devices. It should be noted that some of the redundancy is in the form of extra stages in the restoration stage. Figure 32 was shown with only one restoration stage for simplicity but, it has been shown that multiple stages increases the reliability of a design [Bhaduri04, Han02, Norman04, Qi05].

4.3 Inherently fault tolerant architectures

Some proposed architectures are inherently fault tolerant because of the manner in which they are configured. Nanocells (section 3.2) use a genetic algorithm to "train" cells to perform a certain function [Husband03, Tour03]. By using a genetic algorithm to create the configuration stream on the actual hardware, any defects are avoided. This is very similar to reconfiguration, but the faults are not mapped. It is also less susceptible to transient faults because the output of a cell is determined by a current ratio. This means that if a switch malfunctions, it may not change the output current enough to change the output logic state

4.4 Fault tolerance conclusion

Fault tolerance has come full circle since the pioneering work by John von Neumann, but with a much different outlook and scope. During the time of von Neumann, the scale of the problem was much smaller and there was the expectation that reliable devices would soon be manufactured, making fault tolerance unnecessary. The likelihood that nanotechnology will become reliable enough to make fault tolerance obsolete is small. Even if high reliability becomes achievable, it will undoubtedly be very expensive, which goes against a main goal of nanotechnology. As can be seen from the previous discussions, each technique has its strengths and weaknesses. While reconfiguration requires the least amount of redundancy [Nikolić02] (see Figure 33) it cannot handle transient faults or defects that arise during use (at least until it is reconfigured). It is also time-consuming to detect and map around faults. The fault-tolerant techniques based on hardware redundancy (NMR and multiplexing) can, in theory, mitigate any faults. The main drawback is that much more redundancy is required at a given defect rate, as shown in Figure 33. These techniques are also based on probabilities, which means that some level of chips will not work. For example, in TMR the probabilities can indicate that there is only a 0.1% chance that two duplicated parts will contain faults, but when two parts do contain faults, the chip will fail. To further increase the fault tolerance problem complexity, nano-electronics will probably be susceptible to transient faults as well as manufacturing defects. Therefore any good fault tolerance solution will probably have to use both reconfiguration and hardware redundancy.

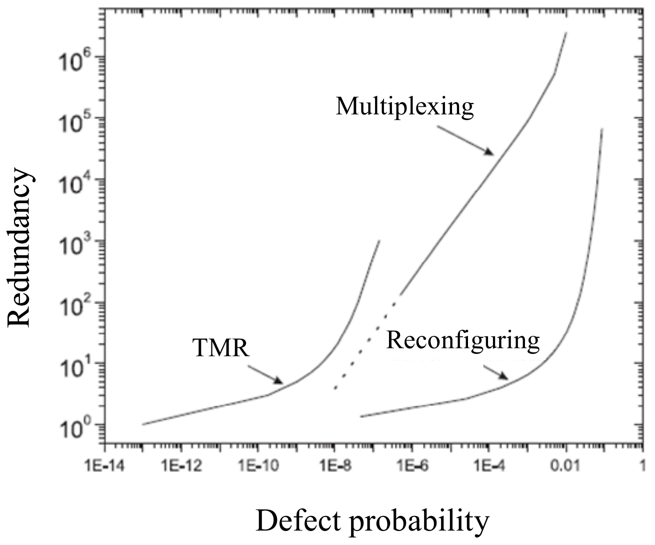

Figure 33: Simulation results for three fault-tolerant techniques applied to a theoretical chip with 10^{12} devices. The curves show the required level of redundancy required to ensure a 90% probability a chip will work for a given individual device defect probability. The level of redundancy is effectively the number of times a circuit has to be replicated. [Nikolić02]

5. Software tools

A transition to nano-electronics will require modifications and additions to current computer aided design (CAD) flows to accommodate the unique properties of nano-electronics. CAD software tools for integrated circuits have been the subject of intense research for many years. These tools leverage the power of computers to assist circuit designers in implementing a circuit. The level of assistance can vary from completely implementing a circuit from a high-level language description to merely providing a GUI for implementing a circuit by hand. CAD tools are becoming more important as designs increase in size and complexity. The number of devices and presence of defects expected in nano-electronics will only increase their importance. While many parts of current FPGA CAD flows can be directly migrated to tools for nanotechnology, there are unique challenges that will require additions and changes to the current tools. Before we discuss the unique challenges of CAD for nano-electronics, the traditional FPGA flow will be discussed to give an indication of what parts will be migrated and what needs to be added.

A CAD tool's primary goal is to create the most efficient implementation of a design within the constraints of the target technology. Thus, the steps shown in Figure 34 are largely dependent on that technology. FPGAs represent a premade chip technology, where a flexible chip architecture is programmed to implement a user's design. This is similar to how nano-electronics are expected to be used.

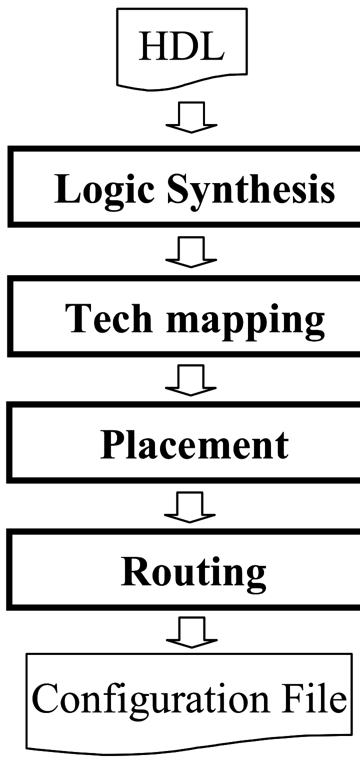

Figure 34. CAD flow for FPGAs.

The first step in the CAD flows is to convert the user's description of a circuit into a format that can be implemented on the targeted technology. This is done in two parts; logic synthesis and technology mapping. Logic synthesis is a general step that takes a high level description (often written in HDL) and converts it to a netlist of logic gates with interconnections. For example, if/else statements are changed into multiplexers and a+b is implemented with a specific adder. In theory, logic synthesis can create a generic implementation, not targeted towards any technology or architecture. In practice however, having knowledge of the underlying technology can provide better implementations.

The technology mapping step is responsible for implementing the circuit netlist in the components that are available in the target. The basic logic element in most modern FPGAs is an SRAM based look-up table (LUT) that can perform any arbitrary N-to-1 logic function (N is normally between 4 and 6, depending on the manufacturer). Most modern FPGAs contain dedicated resources such as multipliers, memories and carry chains for fast addition. Technology mapping for FPGAs takes the netlist produced by logic synthesis and decomposes it into LUTs and the dedicated resources. Optimizations for power, area, and performance can be made in technology mapping by making economic use of an FPGA's resources. For example, if many 4-LUTs are used for two or three input functions, the area will be larger than necessary.

For nano-electronics, logic synthesis and technology mapping will closely resemble the algorithms for FPGAs, because both nano-electronics and FPGAs contain very regular structures. The main difference will be customizing to logic resources such as crossbars and diodes instead of LUTs. Technology mapping may be easier for nano-electronics because of their homogeneous nature. Most of the proposed architectures have only one or two methods for performing logic, eliminating the need to target embedded multipliers, carry chains, etc. The impact of possible defects at this stage depends on the architecture and fault tolerance scheme used. For example, if the architecture is a PLA-based design, and a fault in the PLA is handled by reducing the amount of inputs, outputs, and/or product terms available, the technology mapper will need to be aware of the reductions. An algorithm may decompose the logic into a range of PLA sizes and let the next step find PLAs that meet the requirements. On the other hand, if the architecture is LUT based and entire LUTs are thrown out if a defect is present, then the technology mapper will not need to be concerned with defects.

Placement, the next step in the CAD flow, decides exactly what components on the chip will be used to implement each logic function. While technology mapping determines whether a function should go into a 4-LUT, the placer determines exactly which 4-LUT on the FPGA to use for that function. The goal of placement is to minimize the communication costs by keeping cells that communicate with each other close. This step can save area, power, and delay by minimizing the length of interconnect wires.

The most popular placement algorithm is simulated annealing [Sechen85], which was inspired from metallurgical annealing. The idea in metallurgy annealing is to slowly cool the metal so that crystals (which represent the lowest energy state) can form. In order for this to occur, molecules must gain energy in the form heat to escape from their local minimum configuration. In placement, this equates to accepting some moves that actually make the placement worse so that the placement can escape any local minimum created if a greedy algorithm was used. The first step of simulated annealing is to create a random feasible placement. The next step is to swap two random blocks of the circuit (LUTs in FPGAs). If the new placement is better than the previous, then the move is accepted. The unique feature of simulated annealing is that along with accepting the good moves, a certain percentage of bad moves are accepted. The rate at which bad moves are accepted is a function of the temperature, so that as the placement progresses and the temperature is lowered, fewer bad moves are accepted. The most difficult and important part of a simulated annealing algorithm is determining what “better” is. Better is determined by a cost function that takes into account parameters such as critical path delay, signal congestion, and overall wire length needed to route the placement.

For most proposed nano-electronics, the problem of placement will be similar to FPGA placement. However, while a placer will still try to minimize communications costs, it must now be aware that some parts are defective and cannot be used in a placement. There may also be additional constraints, depending on the architecture. For example, nanowires and nanotubes tend to be short, so long distance communications may be very costly.

After placement, routing selects the interconnect resources to carry signals from their source to their destinations within the chip. The main goal of routing is to reduce delay without overusing any routing channel. FPGAs have a fixed number of wires in the rows and columns between the logic, which means that congestion can occur. Congestion is when the optimum routing solution places more signals in a row or column than there are wires. The crux of most routing algorithms is to start a systematic search for paths from the signal source. For FPGAs, extra steps may be required because of congestion. To handle congestion, the signals are rated by their criticality (how close the signal delay is to the longest delay path) and the signals that are less critical are forced to route somewhere else.

Routing of nano-electronic circuits again will be similar to FPGAs, since most nanoelectronic systems have prefabricated routing channels that are customized to a specific application. However, the router must be defect-aware to avoid using defective interconnect resources. This can be accomplished by removing the defective resources from the routing graph.

In addition to the steps above, an additional step must be added to a nano-electronic CAD flow to handle faults. The most effective way of mitigating defects is to test and configure around them. This requires a step before the placement so that no defective parts are used. As discussed in section 4.1, this will be difficult and time intensive, given the number of possible devices. On the other hand, if software-iserted redundancy is used for fault tolerance, some additions to the front end of the CAD flow will be needed to add that redundancy. A step at the start of the flow will need to add the required redundancy automatically, along with any needed voting circuitry. The technology mapping will then be able to map the circuit, including the redundancy and voting circuits, just like any other circuit.

After a design is routed, it can be deployed. This is where FPGAs and nano-electronics are different from other chips. For custom integrated circuits, each chip is tested to check for fabrication errors, and any defective parts are thrown out, while the working chips are deployed. This is possible because the probability of a device being defective is about 10^{-9}, while there are on the order of 10^{8} devices per chip [ITRS05]. FPGAs must be tested by the manufacturer, but their homogeneity and reprogrammability mean defective chips might still be deployed with a few defects. Some manufacturers put in extra resources that can be permanently swapped in for any defective hardware [Altera06]. This is possible because the LUTs are identical and can be swapped, and defects are rare. It is safe to assume that once a chip is verified defect free, it will remain defect free for many years, and the transient faults will be too rare to cause concern.

While the FPGA manufacturer tests that the device can support any user design, the user actually configures the chip for their desired computation. Fortunately, once a design is finalized and compiled, the same configuration file can be used to configure many identical chips. If a design works on one chip, it is safe to assume it will work on all other chips of the same family and size with the same timing and power characteristics. This makes the configuration and deployment of large numbers of chips fast and inexpensive.

Unfortunately, the deployment method for nano-electronics chips will not be as straightforward. This is due to the random nature of faults that will be present. Like FPGAs, chips will be customized after fabrication. However, unlike FPGAs, each chip will have a unique set of defects. This means that it will be difficult to generate one configuration file to configure multiple chips. Defect detection and defect-aware configuration will likely be a part of fault tolerant schemes. This means that perhaps 10^{11} faults will need to be discovered and stored. This will be an enormous task if a manufacturer ships millions of chips, and could be a hurdle to wide spread use of nano-electronics. If a manufacturer shipped millions of functionally identical chips, each one would need to be tested, their fault map created, and the CAD flow run on each using that fault map. This would be prohibitive, in time and cost, for a device manufacturer. One possible solution is to bootstrap the devices to test and configure themselves. However, we must still handle faults that arise during use. If nano-electronics are to become a replacement for ASICs, the deployment method must become similar to that of an ASIC: chips must be shipped working, and require no user intervention to keep working.

If nano-electroinc circuits are going to be used more like FPGAs, with them configured by the end user, there are other possible methods for deploying the fault map. The map may be kept in software. This could put the responsibility of mapping the faults on the end user as a part of the normal CAD flow. Alternatively, since nano-electronic arrays seem to be well suited to memories, putting the map in memory on the chip may be a viable option.

Overall, many parts of the FPGA CAD flow will be useful for nano-electronics, but there will have to be some non-trivial additions. Technology mapping, placement, and routing will not be very different after the faults are discovered and mapped, except for the problem size. Given that current projections are for nano-electronics to be three or four orders of magnitude more devices than current ICs, their size cannot be ignored. Their size, along with the need for additional steps of defect detection and deployment, will make nano-electronics CAD difficult.

6. Conclusion

The invention of the transistor in 1947 is one of the most important inventions of the 20th century. Since its inception, the transistor has been reduced so that now modern devices are orders of magnitude smaller than their earliest counterparts. Unfortunately, the scaling down must eventually end. Increasing power, capital costs, and ultimately theoretical size limitations, are poised to halt the process of continually shrinking the transistor. Nano-electronics show promise as a technology to continue the miniaturization of ICs. However, whether nano-electronics will be a replacement for conventional ICs, or as a complimentary technology, is yet to be determined. What has already been shown is that components such as wires and molecular switches can be fabricated and integrated into architectures. It is also known that these devices will be prone to defects and that fault tolerance schemes will be an integral part of any architecture. Finally, the preliminary research indicates that while existing parts of the CAD tools will be useful for nano-electronics, there will need to be some additions and changes made.

The greatest progress has been made in the research of the components that may make up nano-electronics. Chemists have been able to fabricate molecules that have two states, such that the molecules can be switched "on" and "off". Some of these molecules have shown the functionality of diodes or variable resistors. Chemists have also been able to fabricate silicon nanowires and carbon nanotubes. Both of these technologies can be used as wires or devices, and in some cases both. Nanoimprint lithography, probably the most promising wire fabrication technique, has been used to produce working memories on the nanometer scale. While all of these devices have been demonstrated, more lot of research is required to reliably produce these devices, and to create better devices.

One of the big questions for the future nano-electronics is whether nano-scale devices can be reliably assembled into architectures. Some small-scale successes have been achieved, but the benefit of nano-electronics is the enormous integration levels they may be able to achieve. The most promising architectures to date are array based. This is because arrays have a regular structure which is easier to build with self-assembly. Arrays also make good use of the available devices (nanowires, carbon nanotubes, and molecular electronics), and they are easy to configure in the presence of defects. There are other more random architectures that would require even less stringent fabrication techniques, but there is some doubt about how they will scale to larger systems. Overall, it is difficult to evaluate architectures as the underlying components are not fully understood nor developed yet. One thing that seems clear is that nano-electronics will, at least for the first few generations, need the support of conventional lithography based electronics for things such as I/O, fault tolerance, and even simple signal restoration.

Fault tolerance is another big problem for nano-electronics. It seems evident that the manufacturing techniques may never be able to produce defect free chips, so fault tolerance will be key to the success of nano-electronics. For manufacturing defects, detecting and configuring around the defects is the most economical technique, since nano-electonics will be configurable devices. The hard problems are detecting the defects among 10^{12} devices in an economical manner, and how to best manage the large defect map. It also appears that transient faults will be a problem with nano-electronics due to their small size and low current levels. To handle transient faults, a hardware redundancy method such as multiplexing or NMR will have to be used to dynamically detect and repair faults. Unfortunately, these methods would require too much redundancy to handle the number of manufacturing defects expected.

One aspect of nano-electronics that resembles current technologies is their CAD flow. Much of the software for utilizing nano-electronics will resemble that of FPGAs. A nano-electronic CAD flow will still have technology mapping, placement and routing to produce configuration files, plus some additional steps. The additions will be a routine to detect the defects before placement, and some kind of backend step to handle the unique circumstances surrounding deployment. The big issue is how to deploy a circuit on a nano-electronic chip when each chip is unique. With current reliable devices, one design can be used to produce millions of chips. If nano-electronics are to become more than a niche computing tool, a deployment model must be developed that doesn't burden the end user or cost the manufacturer excessive testing time.

As can be seen, a substantial amount of research has been conducted on nano-electronics. Many working devices have been designed and fabricated, along with a number of small-scale memory chips, but there are some big hurdles to overcome. These hurdles include lowering defect levels to a point that reasonable redundancy levels can be used, integrating billions of devices, and developing software tools to complement the new technologies. However, the prospect of cheaply integrating 10^{12} devices per chip is a powerful incentive to overcome the challenges. With a little more than 10 years before the projected end of scaling for lithography based circuits, answers to these questions will hopefully come within the decade.

REFERENCES

[Altera06] Altera Corporation, http://www.altera.com/index.jsp

[Amerson95] R. Amerson et al., "Teramac – Configurable Custom Computing," *IEEE Symp. FPGAs for Custom Computing Machines Proceedings*, 1995, pp. 32–38.

[Appenzeller02] J. Appenzeller et al., "Field-Modulated Carrier Transport in Carbon Nanotube Transistors," *Physical Letter Review*, vol. 89, no. 12, 2002, 126801.

[Bachtold01] A. Bachtold et al., "Logic Circuits with Carbon Nanotube Transistors," *Science*, vol. 294, 2001, pp. 1317-1320.

[Bahar04] R.I. Bahar, J. Chen, J. Mundy, "A Probabilistic-Based Design for Nanoscale Computations," In *Nano, Quantum, and Molecular Computing: Implications to High Level Design and Validation*, Kluwer, 2004, pp. 133-156.

[Bhaduri04] D. Bhaduri, S.K. Shukla, "Reliability Evaluation of von Neumann Multiplexing Based Defect-Tolerant Majority Circuits," *4th IEEE Conf. Nanotechnology*, 2004, pp. 599 – 601.

[Brown00] C.L. Brown et al., "Introduction of [2]Catenanes into Langmuir Film and Langmuir-Blodgett Multilayers. A Possible Stategy for Molecular Information Storage Materials," *Langmuir*, vol. 16, no. 4 2000, pp. 1924-1930.

[Brown04] J.G. Brown, R.D. Blanton, "CAEN-BIST: Testing the NanoFabric," *Int'l Test Conference Proc.*, 2004, pp. 462-471.

[Butts02] M. Butts, A. DeHon, S.C. Goldstein, "Molecular Electronics: Devices, Systems and Tools for Gigagate, Gigabit Chips," *IEEE/ACM Int'l Conf. Computer Aided Design (ICCAD02)*, 2002, pp. 430-440.

[Chen99] J. Chen et al., "Large On-Off Ratios and Negative Differential Resistance in a Molecular Electronic Devices," *Science*, 1999, vol. 286, no. 5444, pp. 1550-1552.

[Chen00] J. Chen et al., "Room-Temperature Negative Differential Resistance in Nanoscale Molecular Junctions," Applied Physics Letters, 21 August 2000, vol. 77, no. 8, pp. 1224-1226.

[Chen03] Y. Chen et al., "Nanoscale Molecular-Switch Crossbar Circuits," *Nanotechnology*, vol.14, no. 4, 2003, pp. 462-468.

[Chou96] S.Y. Chou et al., "Nanoimprint Lithography," *J. Vacuum Science Technology B*, vol. 14, no. 6, 1996, pp. 4129-4133.

[Chou97] Y. Chen et al., "Sub-10nm Imprint Lithography and Applications," *Journal of Vacuum Science Technology B*, vol. 15, no. 6, 1997, pp. 2897-2904.

[Collier99] C.P. Collier et al., "Electronically Configurable Molecular-Based Logic Gates," *Science*, vol. 285, no. 5426, 1999, pp. 391-394.

[Collier00] C.P. Collier et al., "A [2]Catenane-Based Solid State Electronically Reconfigurable Switch," *Science*, vol. 289, no. 5482, 2000, pp. 1172-1175.

[Cui00] Y. Cui et al., "Doping and Electrical Transport in Silicon Nanowires," *J. Physical Chemistry*, vol. 104, no. 22, 2000, pp. 5213-5216.

[Cui01a] Y. Cui et al., “Diameter-Controlled Synthesis of Single-Crystal Silicon Nanowires,” *Applied Physics Letters*, vol. 78, no. 15, 2001, pp. 2214-2216.

[Cui01b] Y. Cui, C.M. Lieber, “Functional Nanoscale Electronic Devices Assembled Using Silicon Nanowire Building Blocks,” *Science*, vol. 291, no. 5505, 2001, pp. 851-853.

[Cui03] Y. Cui et al., “High Performance Silicon Nanowire Field Effect Transistiors,” *Nano Letters*, vol. 3, no. 2, 2003, pp. 149-152.

[Goldstein03] S. Goldstein et al., “Reconfigurable Computing and Electronic Nanotechnology,” *Proc. Application-Specific Systems, Architectures, and Processors*, 2003, pp. 132-142.

[Culbertson97] W.B. Culbertson et al., “Defect Tolerance on the Teramac Custom Computer,” *The 5th Ann. IEEE Symposium on FPGAs for Custom Computing Machines Proceedings*, 1997, pp. 116-123.

[DeHon03a] A. DeHon, P. Lincoln, J.E. Savage, “Stochastic Assembly of Sublithographic Nanoscale Interfaces,” *IEEE Trans. on Nanotechnology*, vol. 2, no. 3, 2003, pp. 165-174.

[DeHon03b] A. DeHon, “Array-Based Architecture for FET-Based Nanoscale Electronics,” *IEEE Trans. Nanotechnology,* vol. 2, no. 1, 2003, pp. 23-32.

[DeHon04a] A. DeHon, “Law of Large Number System Design,” In *Nano, Quantum, and Molecular Computing: Implications to High Level Design and Validation*, Kluwer, 2004, pp. 213-241.

[DeHon04b] A. DeHon, M.J. Wilson, “Nanowire-Based Sublithographic Programmable Logic Arrays,” *Proc. 2004 ACM/SIGDA 12th Int’l Symp. Field Programmable Gate Arrays (FPGA04)*, 2004.

[DeHon05a] A. DeHon, H. Naeimi, “Seven Strategies for Tolerating Highly Defective Fabrication,” *IEEE Design & Test of Computers*, vol. 22, no. 4, 2005, pp. 306 – 315.

[DeHon05b] A. DeHon, et al., “Nonphotolithographic Nanoscale Memory Density Prospects,” *IEEE Trans. Nanotechnology*, vol. 4, no. 2, March 2005, pp. 215-228.

[DeHon05c] A. DeHon, “Nanowire-Based Programmable Architectures,” *ACM Jour. Emerging Technologies in Computing Systems*, vol. 1, no. 2, 2005, pp. 109–162.

[DeHon05d] A. DeHon, “Design of Programmable Interconnect for Sublithographic Programmable Logic Arrays,” *Proc. 2004 ACM/SIGDA 12th Int’l Symp. Field Programmable Gate Arrays (FPGA05)*, 2005.

[DeHon05e] A. DeHon, “Deterministic Addressing of Nanoscale Devices Assembled at Sublithographic Pitches”, *IEEE Trans. Nanotechnology*, vol. 4, no. 6, 2005, pp. 681-687.

[Ellenbogen00] J.C. Ellenbogen, J.C. Love, “Architectures for Molecular Electronic Computers: 1. Logic Structures and an Adder Designed from Molecular Electronic Diodes,” *Proc. IEEE*, vol. 88, no. 3, 2000, pp. 386-426.

[Gayasen05] A. Gayasen, N. Vijaykrishnan, M.J. Irwin, “Exploring Technology Alternatives for Nano-Scale FPGA Interconnects,” *42nd Proc. Design Automation Conference (DAC05)*, 2005, pp. 921-926.

[Goldstein01] S.C. Goldstein, M. Budiu, “NanoFabrics: Spatial Computing Using Molecular Electronics,” *Proc. 28th Ann. Int’l Symp. Computer Architecture*, 2001.

[Goldstein02] S.C. Goldstein, D. Rosewater, “What Makes a Good Molecular-Scale Computer Device?,” *CMU CS Technical Report , CMU-CS-02-181*, September 2002. http://reports-archive.adm.cs.cmu.edu/anon/2002/abstracts/02-181.html

[Graham05] A.P. Graham et al., “How Do Carbon Nanotubes Fit into the Semiconductor Roadmap?,” *Applied Physics A Materials Science & Processing*, vol. 80, no. 6, 2005, pp. 1141-1155.

[Gudiksen02] M. Gudiksen et al., “Growth of Nanowire Superlattice Structures for Nanoscale Photonics and Electronics,” *Letters to Nature*, vol. 415, no. 6872, 2002, pp. 617-620.

[Han02] J. Han, P. Jonker, “A System Architecture Solution for Unreliable Nanoelectronic Devices,” *IEEE Trans. Nanotechnology*, vol. 1, no. 4, 2002, pp. 201-208.

[Han05] J. Han et al. "Toward Hardware-Redundant, Fault-Tolerant Logic for Nanoelectronics", *IEEE Design & Test of Computers*, vol. 22, issue 4, 2005, pp. 328 – 339.

[Huang01a] Y. Huang et al., "Directed Assembly of One-Dimensional Nanostructures Into Functional Networks," *Science,* vol. 291, no. 5504, 2001, pp. 630–633.

[Huang01b] Y. Huang et al., "Logic Gates and Computation from Assembled Nanowire Building Blocks," *Science*, vol. 294, no. 5545, 2001, pp. 1313-1316.

[Huang04] J. Juang, M.B. Tahoori, F. Lombardi, "On the Defect Tolerance of Nano-Scale Two-Dimensional Crossbars," *Proc. 19th IEEE Int,l Symp. Defect and Fault Tolerance in VLSI Systems*, 2004, pp. 96-104.

[Husband03] C.P. Husband et al., "Logic and Memory with Nanocell Circuits," *IEEE Trans. Electron Devices*, vol. 50, no. 9, 2003, pp. 1865-1875.

[Iijima91] S. Iijima, "Helical Microtubules of Graphitic Carbon," *Nature*, vol. 354, no. 6341, 1991, pp. 56-58.

[Intel05] Intel Pentium D Processor Extreme Edition 955 Data Sheet, http://download.intel.com/design/PentiumXE/datashts/31030602.pdf

[ITRS05] International Technology Roadmap for Semiconductors, http://www.itrs.net/Common/2005Update/2005Update.htm

[Jaeger97] R.C. Jaeger, *Microelectronic Circuit Design*, Boston, MA, WCB/McGraw-Hill, 1997.

[Kanwal03] A. Kanwal, "A Review of Carbon Nanotube Field Effect Transistors," http://www.eng.auburn.edu/~vagrawal/TALKS/nanotube_v3.1.pdf.

[Kay03] E. Kay, "An Introduction to Rotaxanes and Catenanes," http://www.s119716185.websitehome.co.uk/home/rotcatintro.html, 2003.

[Krupke03] R. Krupke et al., "Seraration of Metallic from Semiconducting Single-Walled Carbon Nanotubes," *Science*, vol. 301, no. 5631, 2003, pp. 344-347.

[Kuekes04] P. Kuekes, "Molecular Electronics and Nanotechnology," unpublished talk

[Kuekes05] P.J. Kuekes et al., "Defect-Tolerant Demultiplexers for Nano-Electronics Constructed from Effor-Correcting Codes," *Applied Physics A Materials Science & Processing*, vol. 80, no. 6, 2005, pp. 1161-1164.

[Lauhon02] L.J. Lauhon et al., "Epitaxial Core-Shell and Core Multishell Nanowire Hetereostructures," *Nature*, vol. 420, n0. 6911, 2002, pp. 57-61.

[Luo02] Y. Luo et al., "Two-Dimensional Molecular Electronics Circuits," *Chemphyschem*, vol. 3, no. 6, 2002, pp. 519-525.

[Martel98] R. Martel et al., "Single- and Mulitwall Carbon Nanotube Field-Effect Transistors", *Applied Physics Letters*, vol. 73, no. 17, 1998, pp. 2447-2449.

[Mathews99] R.H. Mathews et al., "A New RTD-FET Logic Family", *Proc. IEEE*, vol. 87, no. 4, 1999, pp. 596-605.

[McEuen00] P.J. McEuen, "Single-Wall Carbon Nanotubes," *Physics World*, vol. 13, no. 6, 2000, pp 31-36.

[Metzger97] R.M. Metzger et al., "Unimolecular Electrical Rectification in Hexadecylquinolinium Tricyanoquinodimethanide," *J. Am. Chemical Society*, 1997, vol. 119, pp. 10455-10466.

[Mishra03] M. Mishra, S.C. Goldstein, "Defect Tolerance at the End of the Roadmap," *Int'l Test Conference Proceedings*, vol. 1, 2003, pp.1201-1210.

[Mishra04] M. Mishra, S.C. Goldstein, "Defect Tolerance at the End of the Roadmap," In *Nano, Quantum, and Molecular Computing: Implications to High Level Design and Validation*, Kluwer, 2004, pp. 73-108.

[Moore65] G.E. Moore, "Cramming more components into integrated circuits," *Electronics*, vol. 38, no. 8, 1965.

[Morales98] A.M. Morales, C.M. Lieber, "A Laser Ablation Method for the Synthesis of Crystalline Semiconductor Nanowires," *Science*, vol. 279, no. 5348, 1998, pp. 208-211.

[Naeimi04] H. Naeimi, A. DeHon, “A Greedy Algorithm for Tolerating Defective Crosspoints in NanoPLA Design,” *Proc. IEEE Int’l Conf. on Field-Programmable Technology (FPT04)*, 2004, pp. 49-56.

[Neumann56] J. Neumann, “Probabilistic Logic and the Synthesis of Reliable Organism from Unreliable Components,” *Automata Studies*, C.E. Shannon and J. McKarthy, Eds. Princeton, NJ: Princeton Univ. Press, 1956, pp. 43-98.

[Nikolić02] K. Nikolić, A. Sadek, M. Forsaw, “Fault-Tolerant Techniques for Nanocomputers,” *Nanotechnology*, vol. 13, no. 3, 2002, pp. 357-362.

[Norman04] Gethin Norman et al., “Evaluating the Reliability of Defect-Tolerant Architectures for Nanotechnology with Probabilistic Model Checking,” *Proc. 17th Int’l Conf. VLSI Design*, 2004, pp. 907-912.

[Nosho05] Y. Nosho et al, “N-Type Carbon Nanotube Field-Effect Transistors Fabricated by Using Ca Contact Electrodes,” *Applied Physics Letters*, vol. 86, no. 7, 2005, 073105.

[Raja04] T. Raja, V.D. Agrawal, M.L. Bushnell, “A Tutorial on the Emerging Nanotechnology Devices,” *Proc. 17th Int’l Conf. VLSI Design*,” 2004, pp 343-360.

[Raychowdhury04] A. Raychowdhury, K. Roy, “Nanometer Scale Technologies: Device Considerations,” In *Nano, Quantum, and Molecular Computing: Implications to High Level Design and Validation*, Kluwer, 2004, pp.5-33.

[Reed01] M.A. Reed et al., “Molecular Random Access Memory Cells,” *Applied Physics Letters*, vol. 78, no. 23, 2001, pp. 3735-3737.

[Rose03] G.S. Rose, M.R. Stan, “Memory Arrays Based on Molecular RTD Devices,” *Proc. IEEE Conf. Nanotechnology (IEEE-NANO03)*, 2003, pp. 453-456.

[Roy05] S. Roy, V. Beiu, “Majority Multiplexing-Economic Redundant Fault-Tolerant Designs for Nanoarchitectures,” *IEEE Trans. Nanotechnology*, vol. 4, no. 4, 2005, pp. 441-451.

[Rueckes00] T. Rueckes et al., “Carbon Nanotube-Based Nonvolatile Random Access Memory for Molecular Computing,” *Science*, vol. 289, no. 5476, 2000, pp. 94-97.

[Salvo01]	B.D. Salvo et al., "Experimental and Theoretical Investigation of Nano-Crystal and Nitride-Trap Memory Devices," *IEEE Trans. Electron Devices*, vol. 48, no. 8, 2001, pp. 1789-1799.
[Schulz00]	H. Schulz et al., "Master Replication into Thermosetting Polymers for Nanoimprinting," *J. Vacuum Science Technology B*, vol. 18, no. 6, 2000, pp. 3582-3585.
[Sechen85]	C.Sechen, A, Sangiovanni-Vincentelli, "The Timberworlf Placement and Routing Package," IEEE *J. Solid-State Circuits*, vol. SC-20, no. 2, 1985, pp. 510-522.
[Siewiorek92]	D.P. Siewiorek, R.S. Swarz, *Reliable Computer Systems: Design and Evaluation*, 2nd ed., Digital Press, Burlington, MA, 1992.
[Snider05a]	G. Snider et al., "Nanoelectronic Architectures," *Applied Physics A Materials Science & Processing*, vol. 80, no. 6, 2005, pp. 1183-1195.
[Snider05b]	G. Snider, "Computing with Hysteretic Resistor Crossbars," *Applied Physics A Materials Science & Processing*, vol. 80, no. 6, 2005, pp. 1165-1172.
[Snider05c]	G. Snider, W. Robinett, "Crossbar Demultiplexers for Nanoscale Based on n-Hot Codes," *IEEE Tran. Nanotechnology*, vol. 4, no. 2, 2005, pp. 249-254.
[Stan03]	M.R. Stan et al., "Molecular Electronics: From Devices and Interconnect to Circuits and Architectures," *Proc. IEEE*, vol. 91, no. 11, 2003, pp.1940-1957.
[Stewart04]	D.R. Stewart et al., "Molecular-Independent Electrical Switching in Pt/Organic Monolayer/Ti Devices," *Nano Letters*, vol. 4, no. 1, 2004, pp.133-136.
[Tour02]	J.M. Tour et al., "Nanocell Logic Gates for Molecular Computing," *IEEE Trans. Nanotechnology,* vol. 1, no. 2, 2002, pp. 100-109.
[Qi05]	Y. Qi, J. Gao, J.A.B. Fortes, "Markov Chain and Probabilistic Computation – A General Framework for Multiplexed Nanoelectronic Systems," *IEEE Trans. Nanotechnology*, vol. 4, no. 2, 2005, pp. 194-205.
[Wang05]	Z. Wang, K. Chakrabarty, "Built-in Self-Test of Molecular Electronic-Based Nanofabrics," *Proceedings of the European Test Symposium*, 22-25 May, pp.168 – 173.

[Ward04] J.W. Ward et al., “A Non-Volatile Nanoelectromechanical Memory Element Utilizing a Fabric of Caron Nanotubes,” *Non-Volatile Memory Technology Symposium*, 2004, pp. 34-38.

[Whang03a] D. Whang et al., “Large-Scale Organization of Nanowires Arrays for Integrated Nanosystems,” *Nano Letters*, vol. 3, no. 9, 2003, pp. 1255-1259.

[Whang03b] D. Whang, S. Jin, C.M. Lieber, “Nanolithography Using Hierarchically Assembled Nanowire Masks,” *Nano Letter*, vol.3, no. 7, 2003, pp. 951-954.

[Whitaker88] J. F. Whitaker et al., “Picosecond Switching Time Measurement of a Resonant Tunneling Diode,” *Applied Physics Letters,* vol. 53, no. 5, 1988, pp. 385.

[Williams01] S. Williams, P. Keukes, “Demultiplexer for a Molecular Wire Crossbar Network,” U.S. Patent 6 256 767, July 3, 2001.

[Wind02] S.J. Wind et al., “Vertical Scaling of Carbon Nanotube Field-Effect Transistors Using Top Gate Electrodes,” *Applied Physics Letters*, vol. 80, no. 20, 2002, pp. 3817-3819.

[Wu00] Y. Wu, P. Yang, “Germanium Nanowire Growth via Simple Vapor Transport,” *Chemistry of Materials*, vol. 12, 2000, pp.605-607.

[Wu05] W. Wu et al., “One-Kilobit Cross-Bar Molecular Memory Circuits at 30-nm Half-Pitch Fabricated by Nanoimprint Lithography,” *Applied Physics A Materials Science & Processing*, vol. 80, no. 6, 2005, pp. 1173-1178.

[Zeitzoff04] P.M. Zeitzoff, “MOSFET Scaling Trends and Challenges Through the End of the Roadmap”, *IEEE 2004 Custom Integrated Circuits Conference*, 2004, pp. 233-240.