Nanocell Logic Gates for Molecular Computing

James M. Tour, William L. Van Zandt, Christopher P. Husband, Summer M. Husband, Lauren S. Wilson, Paul D. Franzon, and David P. Nackashi

***Abstract*—Molecular electronics seeks to build electrical devices to implement computation—logic and memory—using individual or small collections of molecules. These devices have the potential to reduce device size and fabrication costs, by several orders of magnitude, relative to conventional CMOS. However, the construction of a practical molecular computer will require the molecular switches and their related interconnect technologies to behave as large-scale diverse logic, with input/output wires scaled to molecular dimensions. It is unclear whether it is necessary or even possible to control the precise regular placement and interconnection of these diminutive molecular systems. This paper describes genetic algorithm-based simulations of molecular device structures in a nanocell where placement and connectivity of the internal molecular switches are not specifically directed and the internal topology is generally disordered. With some simplifying assumptions, these results show that it is possible to use easily fabricated nanocells as logic devices by setting the internal molecular switch states after the topological molecular assembly is complete. Simulated logic devices include an inverter, a NAND gate, an XOR gate and a 1-bit adder. Issues of defect and fault tolerance are addressed.**

***Index Terms*—Circuit simulation, genetic algorithms, logic circuit fault tolerance, logic devices, molecular electronics.**

I. Introduction

MOLECULAR electronics seeks to build computational systems, both memory and logic, wherein individual or small collections of molecules serve as discrete device components. Potential advantages of molecular electronic systems could be many-fold including reducing the complexity and cost of current integrated circuit fabrication technologies, reducing heat generation by using only a few electrons per bit of information, and providing a route to meet the ever-continuing demand for miniaturization [1]–[3]. While molecules are approximately one million times smaller in area than their present-day solid state counterparts, this small size brings with it a new set of problems. In order to take advantage of the ultrasmall size of molecules, one ideally needs an interconnect technology that: 1) scales from the molecular dimensions; 2) can be structured to permit the formation of the molecular equivalent of large-scale diverse modular logic blocks as found in very large-scale interconnect (VLSI) architectures; and 3) can be selectively connected to mesoscopically (100 nm scale) defined input–output (I/O). Three architectures that have previously been described are the teramac, nanofabric, and the quantum dot cellular automata (QCA) [4]–[7]. Although these routes possess some distinct advantages, they are dependent upon precise molecular order and on building arrays of logic via exact arrays of nanostructures. Moreover, the I/O challenges remain enormous in those architectural models. The nanocell approach described here is not dependent on placing molecules in precise orientations or locations and the lithographic challenges of the I/O structure become trivial, however, programming issues become far more challenging.

A nanocell is a two-dimensional (2-D) [3-D models could also be considered] network of self-assembled metallic particles connected by molecules that show reprogrammable (can be turned ON or OFF) negative differential resistance (NDR) (or other switching and/or memory properties although these are not addressed in the simulations here) [8]. The nanocell is surrounded by a small number of lithographically defined access leads at the edges of the nanocell. Unlike typical chip fabrication, the nanocell is not constructed as a specific logic gate and the internal topology is, for the most part, disordered. Logic is created in the nanocell by training it postfabrication, similar in some respects to a field-programmable gate array (FPGA). Even if this process is only a few percent efficient in the use molecular devices, very high logic densities will be possible. Moreover, the nanocell has the potential to be reprogrammed throughout a computational process via changes in the ON and OFF states of the molecules, thereby creating a real-time dynamic reconfigurable hard-wired logic. The central processing unit of the computer would be comprised of arrays of nanocells wherein each nanocell would have the functionality of many transistors working in concert. A regular array of nanocells is assumed to manage complexity, and ultimately, a few nanocells, once programmed, should be capable of programming their neighboring nanocells, although the precise heuristic needed for the programming of neighbors has not yet been defined. Alternatively, arrays could be programmed one nanocell at a time via an underlying CMOS platform.

The primary goal of this paper is to demonstrate the initial algorithmic feasibility of programming a nanocell to perform a logic operation—creating functionality from disorder. First, a nanocell description and the molecular switch features are addressed. Then, with some simplifying assumptions, SPICE simulations describe the performance of the nanocell programming algorithms. Defect and fault–tolerance issues are studied. Some details of the genetic algorithm (GA) used are presented before making conclusions.

Manuscript received February 22, 2002; revised June 10, 2002. This work was supported by the Defense Advanced Research Projects Agency (DARPA) and the Office of Naval Research (ONR) under Grant N 00014-01-1-0657.

J. M. Tour, W. L. Van Zandt, C. P. Husband, S. M. Husband, and L. S. Wilson are with the Departments of Chemistry, Computer Science, Computational and Applied Mathematics, and Center for Nanoscale Science and Technology, Rice University, Houston, TX 77251-1892 USA (e-mail: tour@rice.edu).

P. D. Franzon and D. P. Nackashi are with the North Carolina State University, Engineering Graduate Research Center, Raleigh, NC 27695-7914 USA (e-mail: paulf@unity.ncsu.edu).

Digital Object Identifier 10.1109/TNANO.2002.804744

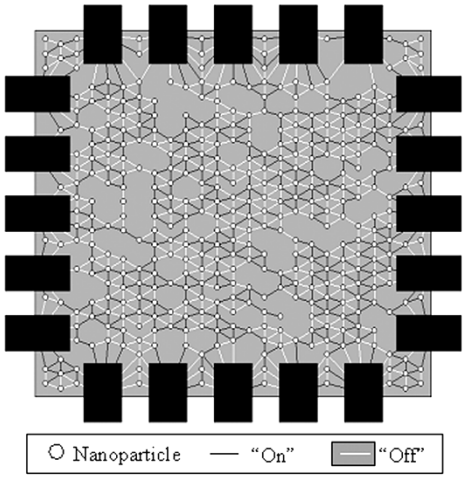

Fig. 1. Simulated self-assembled nanocell is depicted. The black rectangles at the edges are the I/O leads. The entire cell, excluding the outer portions of the contact pads, would be approximately 1 μm^2.

II. Description of the Nanocell

A nanocell could be $\sim$1 μm^2 possessing 20 I/O leads on the edges that can be contacted by standard lithographic wiring. A simulated self-assembled nanocell is depicted in Fig. 1. The black rectangles at the edges are the I/O leads. The entire cell, excluding the outer portions of the contact pads, would be approximately 1 μm^2. A 2-D array of the metal nanoparticles (gray circles) is deposited on an oxide surface (gray background) with a 90% density in this simulation. A molecular self-assembled monolayer coating each nanoparticle would control the spacing between nanoparticles. Molecular switches would insert into the inert self-assembled monolayer barrier around each nanoparticle via processes that have previously been demonstrated, and thereby inter-link adjacent nanoparticles [9], [10]. For the simulation, a Poisson distribution with an average of five molecular switches per nanoparticle is used. This distribution is used because we will eventually move to multiple switches between pairs of nanoparticles. Each molecular switch could be set into an ON state or an OFF state. Although the figure shows nanoparticles in a random subset of a regular grid, such placement is not essential and it merely eases simulation choices. The dimension of 1 μm^2 with 20 access leads was chosen to relax fabrication constraints. As shown in Fig. 1, this network of molecules and particles can be modeled as a planar graph where the nodes are the nanoparticles and the edges are the molecular switches. Based on a nanoparticle diameter of 60-nm and 3-nm spacings by the bridging molecules, each nanocell will contain approximately 200–250 nanoparticles, well within the range of experimental fabrication. In our simulations, the nanoparticles are laid out on a regular hexagonal grid with some probability that each grid location contains a nanoparticle. Initial experiments on nanoparticle depositions on an oxide surface indicate that they do form a fairly regular grid [9]. This regular grid in the simulations has no effect on the programming of a nanocell. It only determines the possible connections within the nanocell. Every nanoparticle could be moved slightly, and the same results would be obtained. It is estimated that an average of five molecular switches will have the proper orientation between adjacent particles' facets to join adjacent nanoparticles. However, for the sake of simplicity, initial simulations assumed that at most one molecular switch bridges any pair of adjacent particles, while later simulations in this paper consider multiple connections. Both the oxide surface attachment and the nanoparticle segregation have been experimentally demonstrated. Our preliminary experimental results on 2-D gold nanoparticle/conjugated molecule assemblies have shown that we can have current flow over the 1-μm distance [9]. The nanoparticles are kept from coalescing into multiparticle arrays through the use of short alkanethiols self-assembled onto the gold nanoparticles [10]. A series of functional, electrically settable molecular switches will then be introduced. Each molecular switch is terminated on both its ends with molecular alligator clips, such as thiols, and allowed to insert between adjacent nanoparticles via self-assembly with molecule-metal chemical bonding to establish the electrical contacts between the adjacent nanoparticles and between the nanoparticles and nearby I/O leads. We have already demonstrated molecular switch insertion and bridging between nanoparticles, and through these structures re-settable enhanced conductivity and resistivity states have been established using voltage pulses at nearby lithographically defined contact pads [9], [11], [12].

Once the physical topology of the self-assembly is formed in the nanocell, it remains static; there is no molecule or nanoparticle dynamic character (other than bond rotations or vibrations) to the highly crosslinked network. The only changeable behavior is in the molecular states: conducting ON or nonconducting OFF, as set by voltage pulses from the periphery of the cell, or as defined by the search algorithms in these simulations.

Several types of room temperature-operable molecular switches have been synthesized and demonstrated in nanopores and atop silicon-chip platforms [4], [5], [9], [11], [12]. The functional molecular switches can be reversibly switched from an OFF state to an ON state, and/or the reverse, based on stimuli such as voltage pulses. The number of nanoparticles (usually metallic or semiconducting) and the number of the interconnecting molecular switches can vary dramatically based on the chosen size of the nanocell and on the dimensions of the nanoparticles and molecules chosen.

Within the fabricated nanocell, the input and output leads could be repetitively interchanged based on the programming needs of the system, thereby demonstrating the pliability of the architecture. Naturally, issues of gain will eventually have to be addressed through either an underlying CMOS layer or clocked circuits programmed into the nanocell [13]–[16]. Even if one CMOS transistor was used for gain at the output from each nanocell, enormous space savings could be attained since a nanocell could possess the functionality of numerous transistors working in concert to produce a specified logic function. Furthermore, by capitalizing on the NDR properties of the molecular switches, internal gain elements based upon NDR/nanoparticle/NDR stacks (Goto pairs) could be efficacious [17], [18].

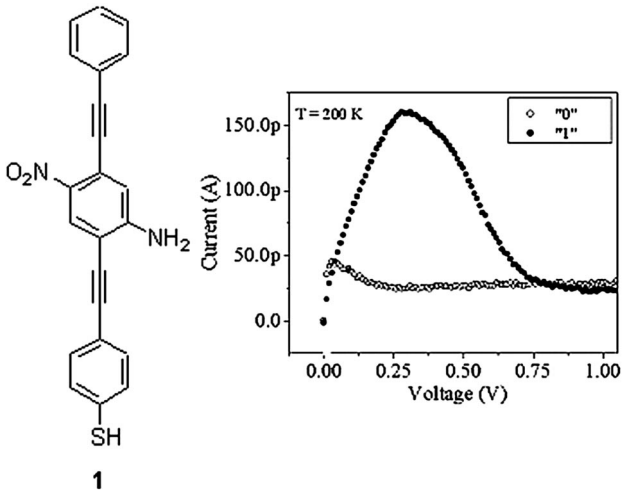

Fig. 2. Shown is our first experimentally obtained $I(V)$ curve of a self-assembled monolayer of **1** between two metallic contacts [11], [12]. Initially, the $I(V)$ response is in the "0" state (open circles). Once application of a 1.75-V pulse takes place, the molecule sets into a new state, "1" (black circles), that exhibits NDR behavior wherein the current rises then falls with increased voltage. Initial simulations used this $I(V)$ curve.

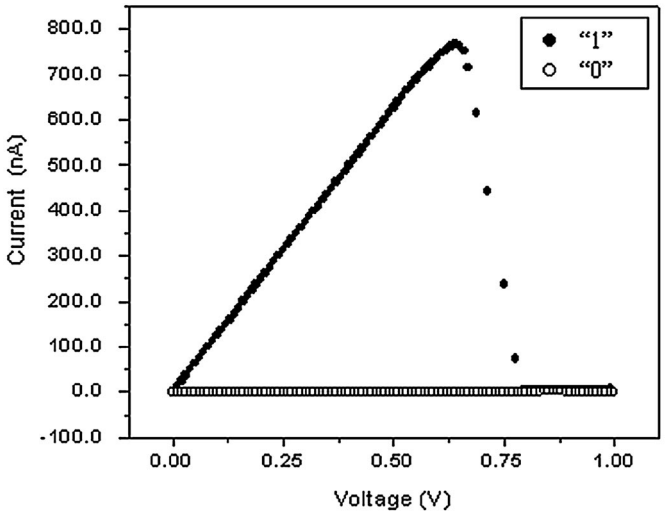

Fig. 3. Shown are the simulated $I(V)$ curves used in logic gates demonstrated later in this paper. The ON-to-OFF ratio is 1000:1. Although these precise curves have not been obtained experimentally, similar NDR behavior with very large ON-to-OFF ratios has been observed at low temperatures. We expect to obtain these characteristics at room temperature in the near future.

III. Molecular Switches

The functionality of a nanocell depends largely on the $I(V)$ (current as a function of voltage) characteristics and placement of its molecular switches with respect to the nanoparticles. We have demonstrated NDR with a large ON-to-OFF ratio from several types of molecular switches based upon nitro-containing oligo(phenylene ethynylene)s, such as one, that are sandwiched between metallic contacts [11], [12]. A characteristic $I(V)$ curve that we have obtained from **1** is shown in Fig. 2, and these devices have been functioning for over one year with no signs of degradation over nearly 10^9 switching events [11], [12]. We will exploit this NDR behavior (rise then decline in the current with increased voltage) in order to build logic devices that exhibit negating functionality such as NAND or XOR responses from these two-terminal devices since two voltage inputs that are high could set the device into an OFF state (right side of the $I(V)$ curve). Switches that do not exhibit the NDR characteristic cannot provide the negating functionality needed for the approach described here [4], [5]. Early simulations were run with the $I(V)$ curve derived from one (Fig. 2). Although NAND gates and inverters were trained, the ON-to-OFF ratios were low (approximately 2:1). However, more recent simulations have been run with a simulated $I(V)$ curve displayed in Fig. 3. This curve has NDR behavior and an ON-to-OFF ratio of 1000:1. Although, it has not been obtained experimentally, similar effects have been observed at low temperatures, and modified systems are being synthesized to optimize the room temperature effects [9], [11], [12]. We expect to obtain room temperature $I(V)$ curves of this sort in the near future.

IV. General Programming

The object in programming or training a nanocell is to take a random, fixed nanocell and turn its switches ON and OFF until it functions as a target logic device. *The physical position of each molecular switch is first fixed; i.e., the internal topology of the nanocell is static. The nanocell is then trained postfabrication by changing the states, ON or OFF, of the molecular switches.*

Here, the terms omniscience, omnipotence, and mortal switching in relation to the programming algorithms used are introduced. By omniscience we mean that the connections within the nanocell and the location and state of each switch are known. Omnipotence means that the search algorithm knows the location of each molecular switch and has precise and selective access to reversibly set its ON or OFF state. Naturally, the definition of omnipotence includes omniscience. Finally, with mortal switching, the algorithm does not know the connections within the nanocell or locations of the switches, and switching is limited to voltage pulses applied to the I/O pins. The actual physical nanocell will be programmed in a mortal fashion and switching will occur only through voltage pulses between contact pads along the periphery.

When we eventually program nanocells through mortal switching, we will treat them as black boxes. However, in exploring this mortal problem it is useful to approach it in a different manner. Programming a nanocell with only mortal switching involves solving two problems: 1) finding switch states such that the given nanocell functions as the target logic device and 2) finding a series of voltage pulses (applied to the I/O pins) that give rise to these desired switch states. In the simulations presented here, we address this first problem and assume omnipotent control over switch states. Given a certain density of nanoparticles and molecular switches, the goal is to determine whether any random nanocell can be trained as some target logic device, with the assumption of absolute control over switch states. We do not plan on actually reaching into a physical nanocell and turning switches ON or OFF. Rather, after thoroughly investigating, via simulations, the first problem of finding switch states where a nanocell functions as a target logic device, we will explore strategies for training a nanocell with voltage pulses at the I/O pins. The simulations here only address the first problem, namely a proof-of-concept that a

nanocell can be trained omnipotently. If it cannot be trained omnipotently, there is no hope for mortal training. However, we present here some results which suggest that mortal training could be efficacious.

However, some preliminary strategies for mortal training include taking advantage of the capacitances of the nanoparticles to better access individual switches.[1] Theoretically, a line of molecular switches between two I/O pins, where there is some capacitance between two nanoparticles, can be set to any pattern of ON and OFF states by using these capacitances. While the network of molecular switches and nanoparticles within a nanocell is much more complicated than a simple line of switches between I/O pins, simulations indicate that the solution space for some logic gates is quite dense. This implies that it will not be necessary to uniquely access every individual molecule. In fact, if there are multiple switches between two nanoparticles, then every switch oriented in one direction will switch states simultaneously. However, this should not be a problem because toggling groups of molecules is most likely sufficient.

Here, the model used to simulate nanocells and the algorithm with which they are trained are presented.

V. Modeling a Nanocell

Given the density and dimensions of the nanoparticles and the average density of the molecular switches, a random nanocell is generated as a hexagonal grid of metallic particles with the specified chosen density. Molecular switches connecting adjacent nanoparticles are distributed following a Poisson distribution based around the given average density (Fig. 1). After the creation of a nanocell, the settings on 20 surrounding I/O pins (five pins occupying each of the four sides) are specified. Each I/O pin can be set to input, output, or to float, and thus behave as a nanoparticle.

A random nanocell must be trained to function as some useful logic device. In order to train a nanocell, the output current resulting from the voltage traces applied to input pins must be evaluated. With individual molecules modeled as nonlinear resistor circuit elements, Intusoft's ICAPS/4 Windows version of SPICE was used to compute the current through each output pin [19]. Achieving convergence in SPICE was resolved by including the parasitic capacitance expected between the nanoparticles. The added capacitance prevents abrupt changes in the current from occurring during simulations, which more realistically models the nanocell architecture and helps with convergence [13].

The nanocell training problem with omnipotence is a combinatorial optimization problem where the search space is the set of all possible switch states for some fixed nanocell. We estimate than an actual nanocell would contain ca. 250–1000 nanoparticles (depending on the size of the nanoparticles chosen) and ca. 750–10 000 molecular switches in the proper orientation between proximal nanoparticle facets contained in a nanocell of 1 μm^2 [9]. In that case, the size of this search space is minimally 2^{750} (as a size comparison, the number of elemental particles in the universe is estimated at 2^{300}). A GA is used to search this space [20].

[1]The authors thank P. Lincoln of SRI for suggesting this approach.

VI. GAs

GAs work by taking a population of individuals, represented as strings of "1s" and "0s," quantifying their fitness, then recombining them to generate a new population of children. Usually the first generation is randomly created, and then three operators are used to produce each subsequent generation: selection, crossover and mutuation. First two parents must be selected. They are either selected randomly or more fit individuals are given preference. The most simplistic method of giving preference is to use roulette wheel selection. Each individual is given a portion of a roulette wheel that is proportional to its fitness. The wheel is spun to select each parent. Hence, the most fit individual is most likely to be selected as a parent, and the least fit individual is the least likely to be chosen. Another method of selection is tournament selection. In tournament selection, a subset of n individuals is chosen, and the two most fit of this group are chosen to reproduce next. Tournament selection has the advantage of varying the degree to which the most fit individuals are favored. Favoring them too much can cause the problem of too little population diversity [20].

Once two parents are selected, they must be recombined to form two new children. Single point crossover is the simplest recombination method. If the length of the chromosome (the string of "1s" and "0s" representing each individual) is m, then some point p between 1 and $m - 1$ is chosen as the crossover point. To create the first child, the first p bits of the first parent are combined with the last $m - p$ bits of the second parent. The second child is created by attaching the first p bits of the second parent to the last $m - p$ bits of the first parent. Alternatively, with n-point crossover, n crossover points are selected and the children are produced analogously. In uniform crossover, a coin is flipped for each bit of the chromosome. If it is heads, the first child gets this bit from the first parent, and the second child gets this bit from the second parent. If it is tails, then the first child gets this bit from the second parent, and the second child gets this bit from the first parent. Hence, uniform crossover is similar to n-point crossover, except the number of crossover points changes with each new pair of children. The two parents are both automatically replaced by their children, or of the four individuals, the two most fit are kept and the other two discarded [20].

After crossover, each of the two new children is mutated. Each bit of each new child is flipped with some probability p. Mutation keeps the GA from losing certain chromosomal information. For instance, without mutation, if every individual in the current generation has a "0" in some particular bit, then it is impossible for subsequent generations to have a "1" in this bit. Hence, potentially beneficial genetic information is not lost when mutation is used [20].

VII. Training Nanocells With a GA

The goal in training a nanocell is to find configurations within a randomly assembled nanocell that will perform as some given logic gate. Genetic algorithms work well for this problem. First a random nanocell is generated and a target logic device is defined (such as NAND). Next, some of the pins are set to input or output. The nanocell is currently a voltage-in, current-out device, so high and low input voltages, V_{IH} and V_{IL} are deter-

mined, as well. When the truth table value of an input is 1, V_{IH} volts are applied to this pin. A truth table value of zero indicates that V_{IL} volts are applied. Similarly, we set I_{OL} and I_{OH} as the low and high output current thresholds, respectively. If the current through an output pin is at or below I_{OL}, that pin is considered OFF, and if the current is at or above I_{OH}, the pin is considered ON.

The states of the nanocell's switches are stored as a "chromosome" of "1s" and "0s." An initial generation of random chromosomes is produced. Each chromosome corresponds to a different set of switch states for the nanocell with fixed locations of nanoparticles and molecular switches. For the results shown here, each generation contains 25 individuals. Next, the fitness of each individual in the generation must be evaluated. The nanocell simulator was written in Windows, so Microsoft's COM platform was used to interface through OLE to Intusoft's ICAPS/4 Windows SPICE variant and, thus, determine the output current for each configuration of switches. We start with the fitness, $f = 0$. Next by parsing the output from SPICE, we determine the output of the individual at each clock step. We then compare these readings to I_{OH} and I_{OL} to determine if the output pin is ON, OFF, or neither (between the discrete threshold settings). If the output at clock step i, x_i is supposed to be ON and is greater than I_{OH}, then nothing is added to f. Otherwise, replace f with $f + (I_{\mathrm{OH}} - x_i)$. If x_i is supposed to be OFF and is less than I_{OL}, then nothing is added to f. Otherwise, replace f with $f + (x_i - I_{\mathrm{OL}})$. In this way, the fitness of each individual is quantified. Note that a fitness of zero indicates that the individual successfully functions as the target logic device; therefore, the GA stops when some individual receives a score of zero.

This fitness formula does not work well for training all logic gates. For instance, in training NANDS, we found that individuals whose current was always high received a better score than those whose output current had the correct shape but was on the wrong scale. To alleviate this problem, we began to scale the fitness score based on the slope of the output current. Hence, the score of an individual whose current decreased when both input pins were at high voltage was scaled down. Alternatively, the fitness of an individual whose current increased or remained the same was scaled up. This technique proved to be extremely effective at quantifying the true fitness of an individual.

After the fitness of a generation is evaluated, tournament selection, uniform crossover, and a mutation probability of $1/(2m)$ are used to create subsequent generations. Children always replace their parents, but the two most fit individuals of the older generation are copied into the younger generation. In the following section, the results of this training process are presented.

VIII. Results of GA Training

Inverters, NAND gates, half-adders, and 1-bit adders have been discovered using the nanocell simulator. For the inverters, NANDS, and half-adders, the simulated $I(V)$ curve displayed in Fig. 3 was used to characterize the ON and OFF states of the molecular switches. A simulated $I(V)$ curve with rectifying diode behavior was used for the 1-bit adder.

Twelve nanocells were randomly generated, and all 12 were successfully trained as inverters. In Fig. 4, the output of one of the inverters is shown. The pin labeled "A" is set to input, and the pin labeled "1" is set to output. Note that the input and output pin are adjacent. This is due to the fact that training negating logic gates is easier with shorter paths from input to output. Shorter paths give rise to higher voltage drops across molecules attached to the output pin, which in turn makes it easier to get past the peak in the $I(V)$ curve. High-input voltage is set at 2 V and low-input voltage is set at 0.5 V. The output pin is considered OFF if there is <7 nA recorded, and ON if >700 nA are recorded. Hence, the ON-to-OFF ratio is 100:1. If the thresholds are allowed to vary with each nanocell, an ON-to-OFF ratio of 1000:1 is obtained. The accompanying plots show the voltage as a function of time $(V(t))$ for input A. The corresponding output current through the output pin is also plotted as a function of time. Note that Output "1" is high when Input "A" is low. Likewise, the Output "1" is low when the Input "A" is high, which is the proper truth table sequence for an inverter (Fig. 4). It took an average of four generations to train each inverter. The simulation time depends primarily on the number of molecular switches in the nanocell. To run a generation of 25 individuals it takes approximately 10 s if there are ten switches, 25 s if there are 100 switches, and 250 s if there are 1000 switches. Hence, four generations took about 160 s on an 800-MHz desktop PC, virtually all of which was simulation time for SPICE to operate. In actual physical training time we estimate that this would take on the order of 1 ms since the nanocell and test electronics can operate at a rate of 100 MHz, thus, 100 000 trials can be performed in 1 ms. Encouragingly, three of the inverters were found in the initial random population, before the GA began to converge upon a solution. This indicates that the solution space is enormous, which will be helpful when the move is made toward more realistic mortal switching and when considering defect- and fault-tolerant needs wherein multiple solutions are necessary.

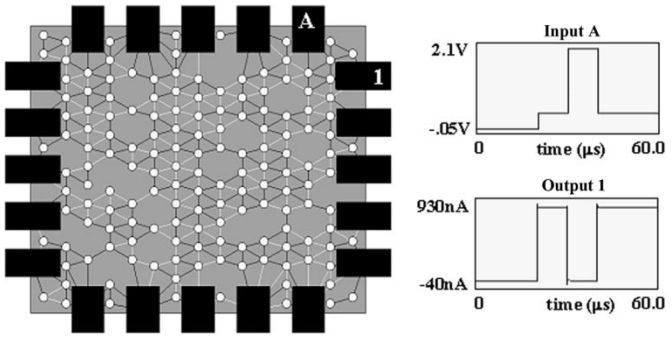

Fig. 4. Inverter is demonstrated in this nanocell using the SPICE model and the $I(V)$ curve shown in Fig. 3. The plots show the input and output voltage as a function of time.

In addition to the inverters, I/O settings for NAND gates were discovered. NANDS are particularly attractive since they constitute a functionally complete logic set meaning that any logic function could be created from NAND sets.[2] The settings that were found to yield NAND gates are shown in Fig. 5. The logic

[2]Note that all functions can be computed from any logical function that generates NOT and AND or OR. Thus, NAND is functionally complete because $\mathrm{NOT}(X) = \mathrm{NAND}(X, 1)$ and $\mathrm{AND}\ (X, Y) = \mathrm{NOT\ NAND}((X, Y))$. AND and XOR are functionally complete because $\mathrm{NOT}\ (X) = \mathrm{XOR}(X, 1)$. The authors thank M. Hill, Univ. of Wisconsin for sharing this with them.

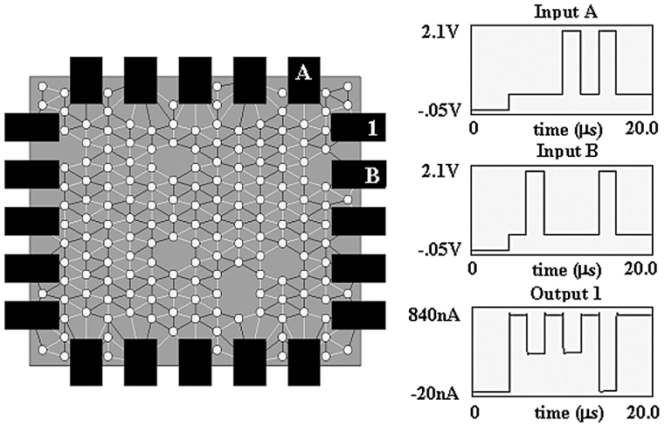

Fig. 5. NAND gate is demonstrated in this nanocell using the SPICE interface model and the $I(V)$ curve shown in Fig. 3. The plots show the $V(t)$ and $I(t)$ with the accompanying NAND Truth Table logic.

TABLE I
TRUTH TABLE FOR A NAND GATE

NAND Truth Table		
Input	Input	Output
0	0	1
1	0	1
0	1	1
1	1	0

for a NAND requires two distinct inputs. The pins labeled "A" and "B" are the input pins, and the pin labeled "1" is the output. High-input voltage is set at 2 V, while low-input voltage is set at 0.5 V. The output pin is considered OFF if there is <6.9 nA recorded, and it is considered ON if >345 nA are recorded. Hence, there is a 50:1 ON-to-OFF ratio. As with the inverters, an even better ON-to-OFF ratio (100:1 or 1000:1) was obtained when the thresholds vary from nanocell to nanocell. The accompanying plots show the $V(t)$ for the inputs "A" and "B" and the corresponding output current over time. Note that output 1 is high when either Inputs "A" or "B" are high, but low when both Inputs "A" and "B" are high, in concert with the accompanying NAND truth table in Table I.

Twelve nanocells were randomly generated, and all 12 were successfully trained. However, the pin settings had to be adjusted to get one of the NANDS to converge since it was too sparse around the original input pins, so the locations of the pins were changed from the upper right corner of the nanocell to the lower left corner. After this change, the nanocell was successfully trained as a NAND gate. On average it took about nine generations for each NAND to converge. This took about 6 min to run due to the SPICE simulations. Again, this should take merely milliseconds in actual physical training time (*vide supra*). As with the inverters, two NAND gates were found in the initial random population. Once again, this indicates a vast solution space, which will be extremely helpful when the assumption of omnipotence is dropped.

To test the robustness of the NAND gates, input "B" is held to high voltage while input "A" is swept from OFF-to-ON-to-OFF. Next, "A" is set to constant high voltage, and "B" is swept from OFF-to-ON-to-OFF. For both of these tests, each NAND gate functions as an inverter with the ON and OFF thresholds given for the NANDS. As anticipated, this implies that the NAND gates are robust.

Finally, a 1-bit adder has been trained (Fig. 6) with a 70-nanoparticle, 1000-molecular switch nanocell, where the molecules exhibit rectifying diode behavior as displayed in Fig. 6. A molecule with this precise $I(V)$ curve has not yet been synthesized; however, a mononitro oligo(phenylenenethynylene) that the authors synthesized does have a similar shape [11], [22]. In Fig. 6, the pins labeled "A" are set to the first input, those labeled "B" are set to the second input, and those labeled "C" are set to the third input. The output pins are labeled "1" and "2." High-input voltage is set at 1.8 V, while low-input voltage is set at 0 V. The input voltages are different from those used for NANDS and inverters because of the fundamental difference in the truth tables. For a NAND or inverter, the output should be high for low inputs, whereas for an adder, the output should be low. One can use identical input voltages by using high rails on the NANDs and inverters. The output pin is considered OFF if there is <50 pA recorded. It is considered ON if >100 pA are recorded. The accompanying plots show the $V(t)$ for the inputs and the corresponding output currents over time, while the truth table for a 1-bit adder is displayed in Table II. Improved search techniques and the exploitation of different molecules and pin settings should improve upon the 2:1 ON-to-OFF ratio exhibited here. This result is significant in that it demonstrates that a nanocell can be trained as a complex logic device with multiple outputs.

IX. DEFECT- AND FAULT-TOLERANCE

More explicit evidence of the size of NAND gate solution spaces was found using SPICE to *exhaustively evaluate every possible combination of switch states for 50 nanocells*. Small cells of 5–16 switches were used because the number of switch state combinations is 2^n, where n is the number of switches. The NAND I/O settings from previous trials were used. The scores of every possible set of switch states for a cell with 14 molecular switches (16 384 possible states) is displayed in Fig. 7. The x axis represents every combination of switch states and the y axis, the corresponding score. The combinations were evaluated in Gray code order; hence, two adjacent combinations differ by exactly 1 bit [20]. For the nanocell results displayed in Fig. 7, 13% of the possible switch states functioned as a NAND. In the 50 test nanocells, 3%–19% of the switch states functioned as NAND. This implies that it will probably not be difficult to mortally train a NAND. It also indicates that a nanocell trained as a NAND is defect tolerant because the performance of the logic gate does not depend on a single set of switch states.

Furthermore, as a check of defect tolerance, large, multiple-switch nanocells were tested for defect tolerance through the SPICE interface. With all switches in the ON position, the cell showed NAND logic with ON-to-OFF output current thresholds of approximately 20:1. Switches were then chosen at random and set to the OFF position and the cell was evaluated periodically (data not shown). The average nanocell tested had 1826 switches, and $>60\%$ of these switches could be turned to the OFF nonconducting state before the cell lost NAND functionality with the minimum output ON-to-OFF ratio set-point of 10:1. This

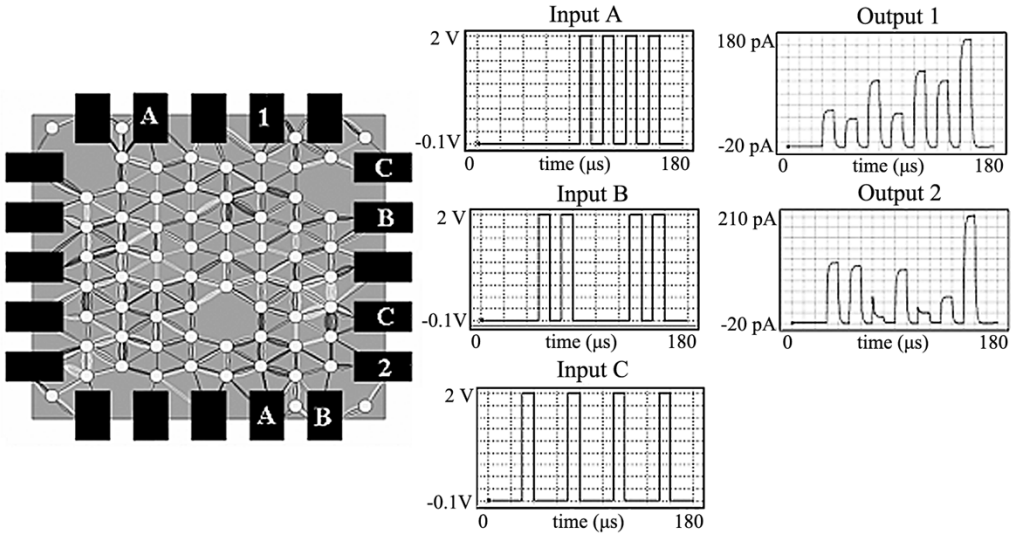

Fig. 6. A 1-bit adder is demonstrated on a randomly assembled nanocell using the SPICE interface model. The plots show the $V(t)$ for the inputs, $I(t)$ for the outputs and the $I(V)$ curve used for the molecules in the ON state. The OFF state is the same as in Fig. 3. The truth table for a 1-bit adder is displayed, as well.

TABLE II
TRUTH TABLE FOR A 1-BIT ADDER

1-Bit Adder				
Input A	Input B	Input C	Output 1	Output 2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

indicates a high tolerance for numerous faults in the nanocell architecture.

In addition, a NAND gate was tested for variation in the $I(V)$ curve of the molecule used in the nanocell. The nanocells were trained with a peak current at 0.625 V. The NAND retained its functionality for peak voltages of 0.61 V–0.77 V. If the peak is less than 0.61 V and the high-input voltage is lowered, the NAND gate also works. Similarly, for peak voltages greater than 0.77 V, the high-input voltage must be increased. Hence, the nanocell maintains its NAND functionality for a wide range of molecular peak voltages. Additionally, if the peak voltage is not in this range, the high-input voltage can be adjusted.

The GA trials and exhaustive tests indicated that to train a nanocell as a NAND, one must simply have enough molecular switches in the ON state near the input and output pins. Additionally, the number of ON molecules between input A and the output should be approximately the same as the number between input B and the output. This conclusion indicated that a single nanocell could be trained as several independent NANDS. This hypothesis was tested on a very large nanocell—approximately 900 nanoparticles and 9000 molecular switches, with multiple switches between adjacent nanoparticles. The nanocell is displayed in Fig. 8 where each two-letter set is the independent NAND input, i.e., "A" and "B," working in concert with the nearby output designated by a numeral, i.e., "1." The molecular switches in each of the four corners are in the ON state while the molecular switches in the middle of the cell are OFF. This establishes a barrier between the four corners. Each of the four corners of this single cell functions as an independent NAND gate with minimally a 15:1 ON-to-OFF ratio. This nanocell might be straightforward to train mortally by initially adding all the molecules into the cell in an OFF state. Next, applying enough voltage to the input pins in each corner to turn ON most of the molecules in that region, a NAND should be afforded in that corner.

X. DISCUSSION

Although configuration of the nanocell was achieved through omnipotent control of each molecule, ultimately the goal is to eliminate this assumption and move toward mortal switching where access should be limited to the surrounding input and output pins. However, before assuming this limited knowledge, it is important to demonstrate the proof-of-principle possibilities for a nanocell using omnipotence. As shown here, the nanocell is a practical approach to molecular computing under these constraints. It does not require molecules to be placed into precise locations, rather it is self-assembled with only short range order, and, therefore, eases fabrication demands, and it addresses the interconnect concerns of molecular computing. Its postfabrication training suggests other exciting possibilities. Nanocells might be reconfigurable "on-the-fly" throughout a calculation process, thereby permitting dynamic hard-wired logic changes, similar to FPGAs or even biological systems. Furthermore, the molecular switches described here that show NDR behavior have been used for memory applications [22], [23]. Thus, the same nanocell could be used for memory in some circumstances and logic in others while nanocells that cannot be trained as high-level logical devices could be used as memory or low-level devices.

The results presented here are for voltage in–current out nanocells. Clearly, uniformity of signal is necessary before nanocells are wired together. However, bistable latches provide a means for obtaining a voltage output driven by current [13],

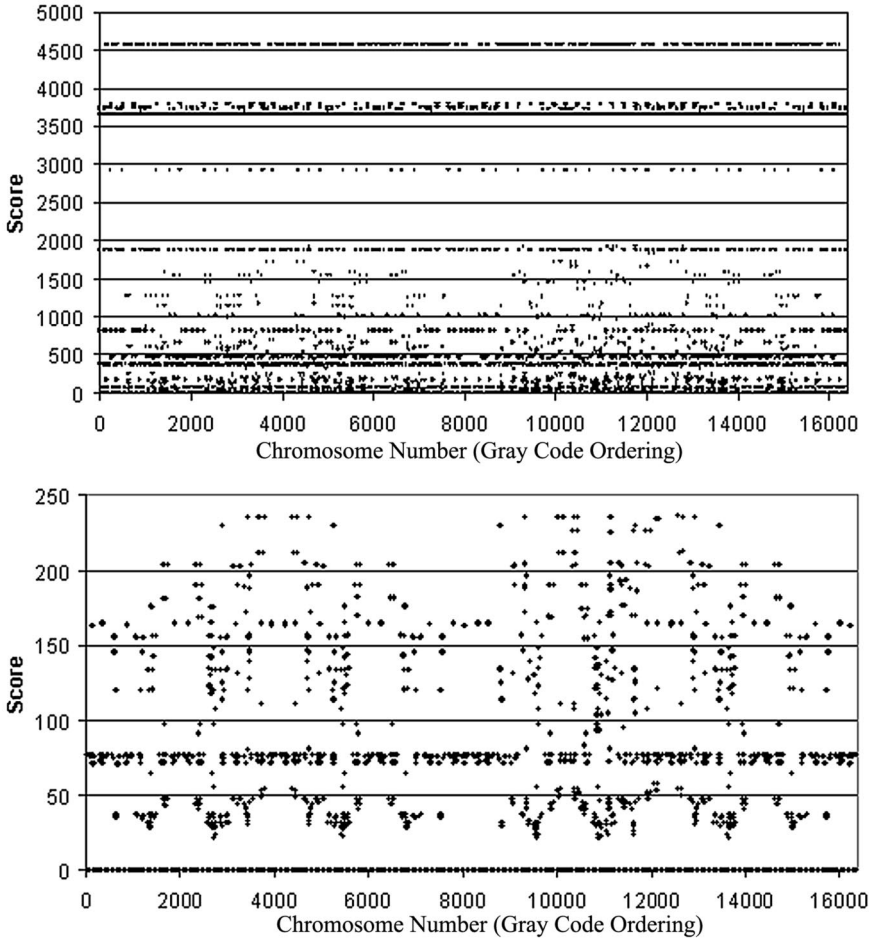

Fig. 7. Displayed is the solution space for configurations of a 14-switch nanocell that function as a NAND gate. The x axis represents each combination of switch states. The y axis represents the corresponding scores, where a score of zero indicates that the nanocell functions as a NAND with these switch states. The bottom graph magnifies the range from 0–250. The space of solutions is clearly dense.

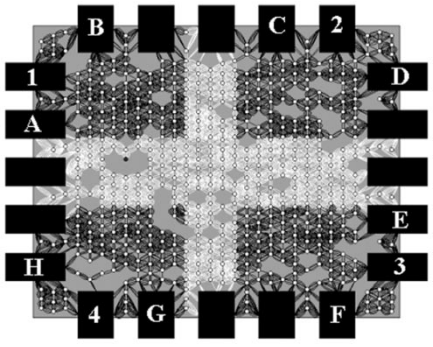

Fig. 8. Nanocell with approximately 900 nanoparticles and 9000 molecular switches that has been trained as four independent NANDS. The ON-to-OFF ratio is 15:1.

[17]. Hence, training voltage in–voltage out latched nanocells is essentially the same as training voltage in–current out nanocells.

More extensive tests remain with the SPICE model. Experiments with various types of molecular switches should be carried out to determine potentially enhanced characteristics. Search algorithms other than the GA should be tested, so as to permit exploration of omniscient and mortal programming. Among other search algorithms, the authors plan to explore simulated annealing, the amoebae algorithm, go-with-the-winners, and genetic hill-climbing [24]–[26]. Some may be best addressed in the actual experimental platforms. Reinforcement learning, as well as a version of GAS called classifier systems will also be investigated [20], [27]. These methods allow purely mortal programming of the nanocell because they treat the nanocell as a black box. Knowledge of the interior of the nanocell is limited to measurements taken at the I/O pins, and the states of molecular switches are altered by voltage applied to the pins. Binary decision diagrams will be explored as a way of learning about the interior of nanocells when the omniscience and omnipotence assumptions are dropped [28].[3]

Another question to answer is what level of logic device complexity is attainable and does the nanocell provide a significant advantage when compared to traditional solid-state dimensional requirements? Based on the International Technology Roadmap for Semiconductors, CMOS will be able to attain simple logic gate functions, such as AND and OR gates, in approximately 1 μm^2, within the next four years [29]. However, the authors are presently seeking to program circuit structures of at least the complexity of a 2-bit adder, with a carry, within a nanocell. In 2005, in solid state devices, a 2-bit adder at 0.1-μm feature sizes will require about 50 transistors over 15–50 μm^2 of chip real estate, depending on the wiring scheme used.

[3]P. Lincoln suggested this technique.

Therefore, significant scaling achievements at lower fabrication constraints might be attained with the nanocell described here.

Finally, in these simulations, the rapid convergence to solutions and exhaustive tests on small nanocells leave us encouraged with regard to the large solution space for a desired logic property, thereby providing a significant level of hard-wired tolerance.

XI. Conclusion

The simulations described here provide a proof-of-principle for the overall nanocell approach. It takes advantage of the small size of molecules within a nanocell and it appears to configure with an interconnect technology that can be structured to permit the formation of the molecular equivalent of large-scale ordered logic with diverse logic functions. Additionally, it can be selectively connected to lithographically defined I/O, and the size of the solution space lends itself to a defect- and fault-tolerance. The mode of programmability from disorder is an exciting approach to molecular computing that could prove essential as the authors move toward molecular-based computing machines, and it should be placed within the arsenal for molecular computing architectural considerations.

References

[1] J. M. Tour, "Molecular electronics. Synthesis and testing of components," *Accounts Chem. Res.*, vol. 33, pp. 791–804, Nov 2000.

[2] M. A. Reed and J. M. Tour, "Computing with molecules," *Sci. Amer.*, pp. 86–93, June 2000.

[3] R. M. Metzger, B. Chen, U. Höpfner, M. V. Lakshmikantham, D. Vuillaume, T. Kawai, X. Wu, H. Tachibana, T. V. Hughes, H. Sakurai, J. W. Baldwin, C. Hosch, M. P. Cava, L. Brehmer, and G. J. Ashwell, "Unimolecular electrical rectification in hexadecylquinolinium tricyanoquinodimethanide," *J. Amer. Chem. Soc.*, vol. 119, pp. 10 455–10 466, Oct. 1997.

[4] C. P. Collier, E. W. Wong, M. Belohradský, F. M. Raymo, J. F. Stoddart, P. J. Kuekes, R. S. Williams, and J. R. Heath, "Electronically configurable molecular-based logic gates," *Science*, vol. 285, pp. 391–394, July 1999.

[5] J. R. Heath, P. J. Kuekes, G. S. Snider, and R. S. Williams, "A defect-tolerant computer architecture: Opportunities for nanotechnology," *Science*, vol. 280, pp. 1716–1721, June 1998.

[6] S. C. Goldstein and M. Budiu, "NanoFabrics: Spatial computing using molecular electronics," presented at the *28th Int. Symp. ISCA*, Göteborg, Sweden, June June 30-July 4, 2001.

[7] C. S. Lent and P. D. Tougaw, "A device architecture for computing with quantum dots," *Proc. IEEE*, vol. 85, pp. 541–557, Apr. 1997.

[8] J. H. Smet, T. P. E. Broekaert, and C. G. Fonstad, "Peak-to-valley current ratios as high as 50:1 at room temperature in pseudomorphic $In_{0.53}Ga_{0.47}As$/AIAs/InAs resonant tunneling diodes," *J. Appl. Phys.*, vol. 71, pp. 2475–2477, Mar. 1992.

[9] D. L. Allara, M. A. Reed, and J. M. Tour, presented at the *DARPA Research Conf.*, Santa Fe, NM, Mar. 2001.

[10] T. D. Dunbar, M. T. Cygan, L. A. Bumm, G. S. McCarty, T. P. Burgin, W. A. Reinerth, L. Jones II, J. J. Jackiw, J. M. Tour, P. S. Weiss, and D. L. Allara, "Combined scanning tunneling microscopy and infrared spectroscopic characterization of mixed surface assemblies of linear conjugated guest molecules in host alkanethiolate monolayers on gold," *J. Phys. Chem. B.*, vol. 104, pp. 4880–4893, May 2000.

[11] J. Chen, W. Wang, M. A. Reed, A. M. Rawlett, D. W. Price, and J. M. Tour, "Room-temperature negative differential resistance in nanoscale molecular junctions," *Appl. Phys. Lett.*, vol. 77, pp. 1224–1226, Aug. 2000.

[12] J. Chen, M. A. Reed, A. M. Rawlett, and J. M. Tour, "Large on-off ratios and negative differential resistance in a molecular electronic device," *Science*, vol. 286, pp. 1550–1552, Nov. 1999.

[13] D. P. Nackashi and P. D. Franzon, "Molectronics: A circuit design perpective," in *Proc. SPIE*, vol. 4236, Mar. 2001, pp. 80–88.

[14] C. M. Lieber and Y. Cui, "Functional nanoscale electronic devices assembled using silicon nanowire building blocks," *Science*, vol. 291, pp. 851–853, Feb. 2001.

[15] J. H. Schön, J. Meng, and Z. Bao, "Field-effect modulation of the conductance of single molecules," *Science*, vol. 294, pp. 2138–2140, Dec. 2001.

[16] P. G. Collins, M. S. Arnold, and P. Avouris, "Engineering carbon nanotubes and nanotube circuits using electrical breakdown," *Science*, vol. 292, pp. 706–709, Apr. 2001.

[17] E. Goto, K. Murata, K. Nakazawa, K. Nakagawa, T. Moto-Oka, Y. Matsuoka, Y. Ishibashi, T. Soma, and E. Wada, "Esaki diode high-speed logical circuits," *Proc. IRE*, vol. 9, pp. 25–29, 1960.

[18] J. C. Ellenbogen and J. C. Love, "Architectures for molecular electronic computers: Logic structures and an adder designed from molecular electronic diodes," *Proc. IEEE*, vol. 88, pp. 386–426, Mar. 2000.

[19] A. Vladimirescu, *The SPICE Book*. New York: Wiley, 1994.

[20] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison Wesley, 1989.

[21] J. Chen, W. Wang, M. A. Reed, A. M. Rawlett, D. W. Price, and J. M. Tour, "Room temperature negative differential resistance in nanoscale molecular junctions," *Proc. Mater. Res. Soc.*, vol. 582, pp. H3.2.1–H3.2.5, 2001.

[22] M. A. Reed, J. Chen, A. M. Rawlett, D. W. Price, and J. M. Tour, "Molecular random access memory cell," *Appl. Phys. Lett.*, vol. 78, pp. 3735–3737, June 2001.

[23] D. T. Pham and D. Karaboga, *Intelligent Optimization Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, New York: Springer-Verlag, 2000.

[24] J. A. Nedler and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, pp. 308–313, Jan. 1965.

[25] D. Aldous and U. V. Vazirani, "Go with the winners algorithms," in *Proc. 35th Annu. Symp. Foundations of Computer Science*, 1994, pp. 492–501.

[26] D. H. Ackley, *A Connectionist Machine for Genetic Hill-climbing*. Norwell, MA: Kluwer, 1987.

[27] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[28] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, pp. 677–691, Aug. 1986.

[29] SEMATECH [Online]. Available: http://public.itrs.net/

James M. Tour received the B.S. degree in chemistry from Syracuse University, Syracuse, NY, in 1981 and the Ph.D. degree in synthetic organic and organometallic chemistry from Purdue University, West Lafayette, IN, in 1986.

He then pursued Postdoctoral training in synthetic organic chemistry at the University of Wisconsin, Madison, WI and Stanford University, Stanford, CA. After spending 11 years on the faculty of the Department of Chemistry and Biochemistry at the University of South Carolina, Columbia, he joined the Center for Nanoscale Science and Technology, Rice University, Houston, TX, in 1999 where he is currently the Chao Professor of Chemistry and Professor of Computer Science, and Mechanical Engineering and Materials Science. He is a co-founder of Molecular Electronics Corporation, Hilton Head Island, SC. His scientific research includes molecular electronics, chemical self-assembly, conjugated oligomers, electroactive polymers, combinatorial routes to precise oligomers, polymeric sensors, flame retarding polymer additives, carbon nanotube modification and composite formation, synthesis of molecular motors and nanotrucks, use of the NanoKids concept for K-12 education in nanoscale science, and methods for retarding chemical terrorist attacks.

Dr. Tour has won several national awards including the National Science Foundation Presidential Young Investigator Award in Polymer Chemistry and the Office of Naval Research Young Investigator Award in Polymer Chemistry.

William L. Van Zandt received the B.B.A. degree in finance from the University of Texas, Austin, in 1971, the M.S. degree in mathematics from the University of Houston, Houston, TX, in 1975, and the M.S. degree in software engineering from the University of Houston—Clear Lake, Clear Lake, TX, in 2000. He is currently pursuing the Ph.D. degree in computer science at Rice University, Houston, TX.

He has been actively employed in industry as a programmer, analyst, consultant, and teacher. In 2000, he joined the Tour Research Group as a Research Associate working on the molecular computer project.

Christopher P. Husband received the B.S. degree in mathematics and physics from Texas A&M University, College Station, in 1998 and the M.A. degree in computational and applied mathematics from Rice University, Houston, TX, in 2002, where he is currently working toward the Ph.D. degree in computational and applied mathematics.

His research interests include neuro-dynamic programming and other approaches to difficult optimization and control problems.

Summer M. Husband received the B.A. degree in mathematics from Texas A & M University, College Station, in 1997 and the Ph.D. degree in computational and applied mathematics from Rice University, Houston, TX, in 2002. Her Ph.D. thesis was entitled, "Programming the Nanocell, a Random Array of Molecules."

Currently, she is a Welch Postdoctoral Research Fellow at Rice University. Her research interests include modeling and simulations of nanoscale computer architectures, artificial intelligence approaches to discrete optimization, and graph theory approaches to applied problems.

Lauren S. Wilson is currently working toward the B.S. degree in mathematics at Rice University, Houston, TX.

In 2001, she joined the Tour Research Group as an Undergraduate Research Associate and is currently studying in Freiburg, Germany. Upon graduation in 2003, she plans to attend medical school and pursue an M.D. Ph.D. degree.

Ms. Wilson was the valedictorian of the 1999 graduating class of Sentinel High School, Missoula, MT and is a National Merit Scholar.

Paul D. Franzon received the Ph.D. degree from the University of Adelaide, Adelaide, Australia, in 1988.

He has worked at AT&T Bell Laboratories, DSTO Australia, Australia Telecom, and Communica Ltd., Australia. Currently, he is a Professor in the Department of Electrical and Computer Engineering at North Carolina State University, Raleigh. He has lead several major efforts and published over 80 papers in these areas. His current research interests center on the technology and design of complex systems incorporating VLSI, MEMS, advanced packaging, and molecular computing. Application areas currently being explored include novel advanced packaging structures, Network Processors, SOI baseband radio circuit design for deep space, on-chip inductor and inductance issues, RF MEMS, and moleware circuits and characterization.

Dr. Franzon received the National Science Foundation (NSF) Young Investigators Award in 1993 and was selected to join the NCSU Academy of Outstanding Teachers in 2001.

David P. Nackashi received the M.S. degree in computer engineering from North Carolina State University, Raleigh, in 1999 and the B.S. degree in electrical engineering from the Georgia Institute of Technology, Atlanta, in 1993. Currently, he is pursuing the Ph.D. degree in electrical engineering at North Carolina State University.

His research is in the area of molecular electronics and molecular computing.