Probabilistic Modeling and Analysis of Nanocell based Molecular Memory

Renu Kumawat, Malaviya National Institute of Technology Jaipur, India
Vineet Sahula, Malaviya National Institute of Technology Jaipur, India
Manoj Singh Gaur, Malaviya National Institute of Technology Jaipur, India

This manuscript investigates the aspects of designing a nanocell based molecular memory. An empirical model for molecular device is developed, based on circuit behavior of nitro-substituted Oligo (Phynylene Ethynylene) molecule (OPE). This device model is subsequently used to design nanocell based 1-bit memory cell and verified using HSPICE. The approach is extended to train the nanocell for multi-bit storage capability using external voltage signals. It is observed that to successfully train a 2-bit molecular memory, the number of control signals should be approx. one-fourth of total number of nanoparticles. A computational framework is proposed to compute the probability of retrieving the stored data bits correctly, at the output terminal of the nanocell buffer. This nanocell configuration is simulated by systematically varying number of nanoparticles as well as molecular switches. It has been observed that the probability of the existence of at least one path from input to output approaches close to unity with presence of 20 or more nanoparticles in a nanocell. During memory model validation, one thousand samples of 1-bit memory cell (consisting of 20 nanoparticles) were generated and verified for read and write memory operations. The model verification results obtained for this memory cell closely match those obtained using analytical solution of probabilistic graph model. Omnipotent training is used to train the multi-bit molecular memory device using Genetic Algorithm. Later, the same device is also explored under mortal training, with equally encouraging results.

Categories and Subject Descriptors: B.3.3 [**Memory Structures**]: Performance Analysis and Design Aids; B.3.4 [**Memory Structures**]: Reliability, Testing, and Fault Tolerance

General Terms: Design, Algorithms, Device Modeling, Probability

Additional Key Words and Phrases: Nanocell, molecular device model, nano-electronics, reliability, self-assembly, nanoparticles.

ACM Reference Format:
Kumawat, R., Sahula, V., Gaur, M.S. 2012. Probabilistic Modeling and Analysis of Nanocell based Molecular Memory ACM J. Emerg. Technol. Comput. Syst. 8, 4, Article 40 (June 2010), 16 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Performance and reliability of nanoscale memory and logic devices is determined by few electron-phenomena. In this context, the organic molecules [Chen et al. 1999; Chen et al. 2000; Reed et al. 2001] may offer some advantages for future memory applications. Such molecular devices may be economically processed through

This work is supported by the Ministry of Communication and Information Technology, India, in the form of grant provided under the sponsored project "Special Manpower Development Project for VLSI Design and related software", phase II.
Author's addresses: R. Kumawat and V. Sahula, Department of Electronics and Communication Engineering, Malaviya National Institute of Technology Jaipur, India; M.S. Gaur, Department of Computer Engineering, Malaviya National Institute of Technology Jaipur, India.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

chemical-self assembly and synthesis. These molecular devices are constructed by harnessing the electrical properties of individual molecules or a group of molecules. Researchers have demonstrated the switching behavior of the molecule and have constructed simple logic functions [Cui and Lieber 2001; Huang 2001; Zhong et al. 2003] and memory [Chen et al. 2003; Wu et al. 2005; Green et al. 2007] by using such programmable molecules. These bistable molecules form active molecular switches, which are used for bit storage in crossbar molecular memory. The area required for 1-bit storage is defined by the diameter of the molecule and the intersecting nanowires. Emerging molecular crossbar technology offers high density, regular array-like and non-volatile memory structure [Dehon 2005; Stan et al. 2003]. These devices consume low power, offer low programming voltage and high switching speed. However, the bottom-up approach employed for device fabrication at nanoscale, lacks precision in molecular device ordering. The programmable nanocell based approach [Tour et al. 2002] mitigates this problem.

In contrast to molecular crossbar devices, a nanocell [Tour et al. 2002; Husband et al. 2003; Skoldberg et al. 2007] consists of conducting nanoparticles connected via randomly placed molecules and addressed by relatively small number of leads located at the edges. These molecules exhibit reprogrammable negative differential resistance (NDR) characteristics. The spacing between nanoparticles and the insertion of molecules between nanoparticles is controlled by chemical self assembly. A monolayer of alkanethiols that coats each nanoparticle, prevents them from coalescing into a multi-particle array. The electrical contacts between adjacent nanoparticles and between nanoparticles and I/O leads is established via molecule-metal chemical bonding. A typical nanocell would contain 250-1000 nanoparticles and 750-$10,000$ molecular switches approximately. Tour *et al.* used gold nanoparticles of diameter $60\ nm$ and spacing of $3\ nm$. The size of a typical nanocell is approximately $1\ \mu m^2$. The postfabrication training of a nanocell (omnipotently or mortally [Tour et al. 2002]) can be formulated as an optimization problem. The optimization process would require as an input the nanocell to be trained, the target logic in the form of a truth table and the I-V characteristics of the $'ON'$ and $'OFF'$ states of the molecular switch. The search space for omnipotent training would include all possible combinations of $'ON'$ and $'OFF'$ molecules and all possible assignments of input and output pins. The in-built defect tolerance, small size, postfabrication programmability through mortal training and the lack of requirement of precise molecular ordering features makes it a good choice for future nanoscale devices.

1.1. Our Contribution

In this manuscript, the device model of nitro-substituted Oligo (Phynylene Ethynylene) (OPE) molecule or $2'$-amino-4,$4'$di(ethynylphenyl)-$5'$-nitro-1-benzenethiol [Chen et al. 1999; Chen et al. 2000; Reed et al. 2001] is captured in Verilog-A and used for designing molecular memory. The model is based on empirical equations that depict the I-V characteristics of a molecule [Ziegler and Stan 2002; Rose et al. 2004; Rose et al. 2007]. The device model of OPE molecule follows the I-V characteristics as shown in Fig. 1(b) at very low temperatures, which almost matches the curve for molecule as shown in Fig. 3 of [Tour et al. 2002]. It is expected that molecules, exhibiting such I-V characteristics at room temperatures, would be synthesised in the near future [Tour et al. 2002].

A probabilistic framework is proposed in this paper for modeling molecular memory based on the nanocell approach. An algorithm is proposed to compute the probability that at least one path is present from the input to the output of a nanocell buffer. The nanocell based memory cell has been designed using device model of gold nanoparti-

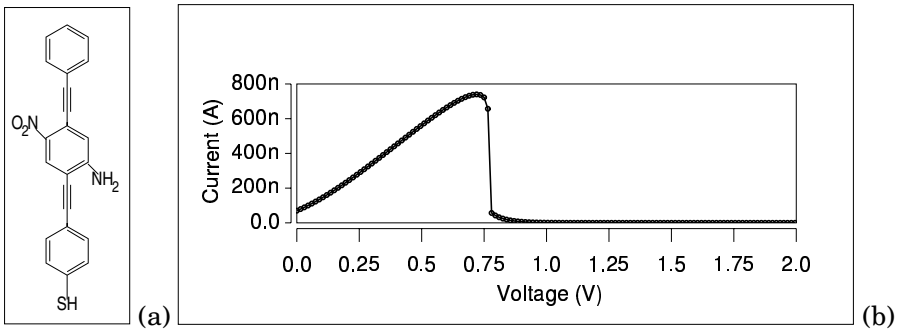

Fig. 1. (a) Nitro substituted Oligo (Phynylene Ethynylene) molecule (b) I-V curve obtained for molecular device model used in molecular memory design. In $'OFF'$ state, approximately zero current flows.

cles and OPE molecular switches. The 1-bit molecular memory cell is verified for 1000 random configurations generated using Monte Carlo Simulation. It has been observed that such a memory cell can successfully perform read and write operation for more than 99.5% of the samples. Also, the presence of multiple paths from input to output of a nanocell increases the probability of receiving correct output, even in presence of soft transient errors.

The concept is further extended and applied for (i) omnipotent training as well as (ii) mortal training of a multi-bit memory, using Genetic Algorithm (GA). As discussed in [Tour et al. 2002], in case of omnipotent training of a nanocell, the location of each molecular switch is known to the search algorithm and their states can be switched to $'ON'$ or $'OFF'$. However, in case of mortal training, the internal topology of the nanocell is not known to the search algorithm and switching of internal molecular switches is controlled by the external voltage pulses. Henceforth, in the proposed approach for mortal training, the external control voltage signals (CVS) are applied to some of the nanoparticles in the nanocell. All possible combinations of high and low voltage values are applied to these (CVS) and a set of CVS voltage values are obtained for which the nanocell behaves as a n-bit memory. Since, the search space exponentially increases with the increase in number of nanoparticles and number of CVS, the proposed Genetic Algorithm based mortal training algorithm can be applied to successfully train a nanocell in polynomial time. However, a considerable amount of noise margin is present in the trained 2-bit memory device. The proposed methodology is flexible and easily scalable to train the nanocell for multi-bit storage functionality.

The rest of the paper is organized as follows. In Section 2, the 1-bit memory cell design approach and verification results are presented. The approach is extended to omnipotently train a multi-bit memory. The mortal training of a multi-bit memory is discussed in Section 3. The proposed algorithm that searches the complete design space for training the multi-bit memory is also discussed in Section 3. Further, in Section 4, the probabilistic analysis of a single-input, single-output nanocell buffer is explained. Also, the lower and upper bounds of this probability are calculated. The experimental setup for the probabilistic analysis of nanocell based molecular memory is explained in Section 5. The conclusion and further works are discussed in Section 6.

2. EXPERIMENTAL SETUP FOR OMNIPOTENT TRAINING

As an initial attempt, a 1-bit molecular memory cell model has been designed using nanocell based approach. Fig. 2(a) shows schematic of a nanocell based memory, which has an Address (A), Write/Read (W/R), Data_In (In) and Data_Out (Out) ports. As explained earlier, the nanoparticles are connected to the molecules via metal-molecule

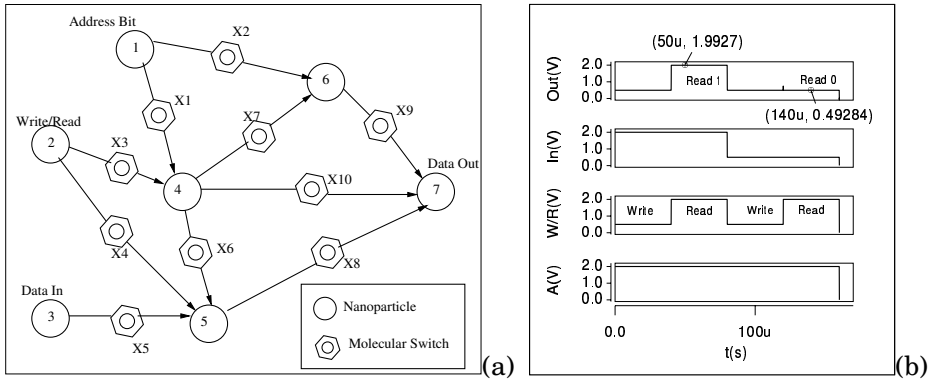

Fig. 2. Nanocell based 1 - bit molecular memory cell (a) Schematic (b) Simulation results for Address (A), Write/Read (W/R), Data_In (In) and Data_Out (Out) terminals. Here, $2V$ and $0.5V$ represent logic $'1'$ and logic $'0'$, respectively.

chemical bonding. When Address bit is set to high voltage ($2V$) and the memory is in write mode ($A = 2V$ and $W/R = 0.5V$), the data on the Data_In port is stored to the memory. Now, when the memory is switched to read mode ($A = 2V$ and $W/R = 2V$), the data stored in the memory can be read out from Data_Out port. Initially, the nanocell is designed in HSPICE using Verilog-A model of the Oligo (Phynylene Ethynylene) molecule. The molecular connections in this nanocell are done using the directed acyclic graph generated using Nanocell Reliability Prediction Algorithm(NRPA) (to be explained in Section 5, Algorithm 3). For molecular connections, the following assumptions are made:

(1) A certain/fixed number of nanoparticles are present inside the nanocell.
(2) Molecules are spatially distributed following the Gaussian Distribution within the nanocell.
(3) Between two nanoparticles N_i and N_j, only single molecular switch M_{ij} is present.
(4) When voltage ($\geq V_{threshold}$) is applied on N_i, the conformational changes occur in the molecule M_{ij}. This changes its resistance value and current flows through M_{ij} to the nanoparticle N_j, (as shown in Fig. 1(b) of [Reed et al. 2001]). Thus, we say that, the molecules in the nanocell are unidirectional i.e. directed in the direction of current flow (from the set of input nodes to the output node).

Considering these assumptions, the nanocell is omnipotently trained to behave as a 1-bit memory cell using the Genetic Algorithm. By omnipotence, we mean to say that, the internal topology of the nanocell is priori known to us and individual molecules can be switched to $'ON'$ or $'OFF'$ state. The set of molecular switches and their locations within the nanocell is passed as initial population to the algorithm. The algorithm switches these molecules to $'ON'$ or $'OFF'$ state, to find the optimal set of $'ON'$ molecular switches for which the nanocell behaves as 1-bit memory. Fig. 2(b) depicts the simulation results for a 1-bit molecular memory cell, which consists of 7 nanoparticles and 10 molecular switches. The spice file has been simulated for $\{(w_1, r_1), (w_0, r_0)\}$ operations. The high and low input voltage levels are defined as $2.0V$ and $0.5V$, respectively. One thousand samples of proposed 1-bit molecular memory cell with 20 nanoparticles are generated using Monte Carlo simulation. These samples are not genetically trained for correct functionality. The input signals for $\{(w_1, r_1), (w_0, r_0)\}$ operations are given on the input terminals and output is observed on the Data_Out terminal. Also, the simulation results for Read 1 (r_1) and Read 0 (r_0) are plotted in Fig. 3. The output voltage range for r_1 operation falls between $1.989\,V$ and $1.994\,V$. It is inferred that even

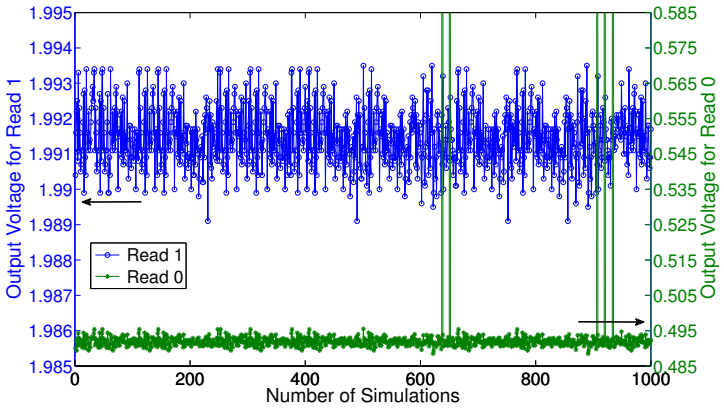

Fig. 3. Simulation results for 1000 samples of 1-bit memory cell. The left and right y-axises represents Data_Out voltage for Read 1 and Read 0, respectively

for an untrained memory cell, bit 1 is always read correctly and bit 0 is read correctly for more than 99.5% cases in the range of $0.490\ V$ to $0.495\ V$. Thus, we can conclude that even an untrained nanocell, consisting of 20 or more nanoparticles and a monolayer of OPE molecules, behaves as a reliable memory device. This observation is theoretically proved in the Section 4 and 5.

2.1. Omnipotent training of multi-bit memory

The concept of omnipotence is further extended to train multi-bit molecular memory device. A n-bit memory device consists of n Address lines ($Ai, \forall\ i\ =\ 1\ to\ n$), a Write/Read ($W/R$), a Data_In ($In$) and a Data_Out ($Out$) ports. Whenever i^{th} Address line ($Ai = 2V$) is high, the Write (w_a^i) or Read (r_a^i) operation is performed for the i^{th} bit ($a\ =\ 0\ or\ 1$). The Fig. 4 shows the simulation results of an omnipotently trained two-

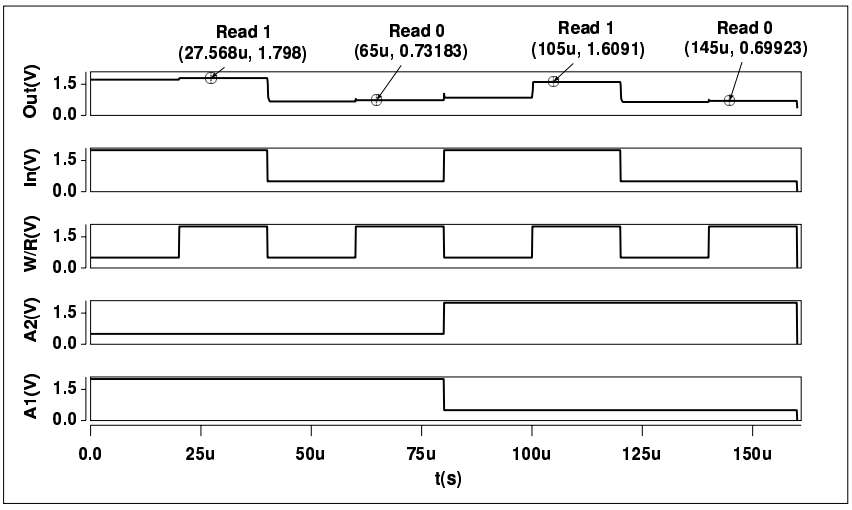

Fig. 4. Nanocell based 2 - bit molecular memory cell: Simulation results for Address 1 (A1), Address 2 (A1), Write/Read (W/R), Data_In (In) and Data_Out (Out) terminals. Here, $2V$ and $0.5V$ represent logic $'1'$ and logic $'0'$, respectively.

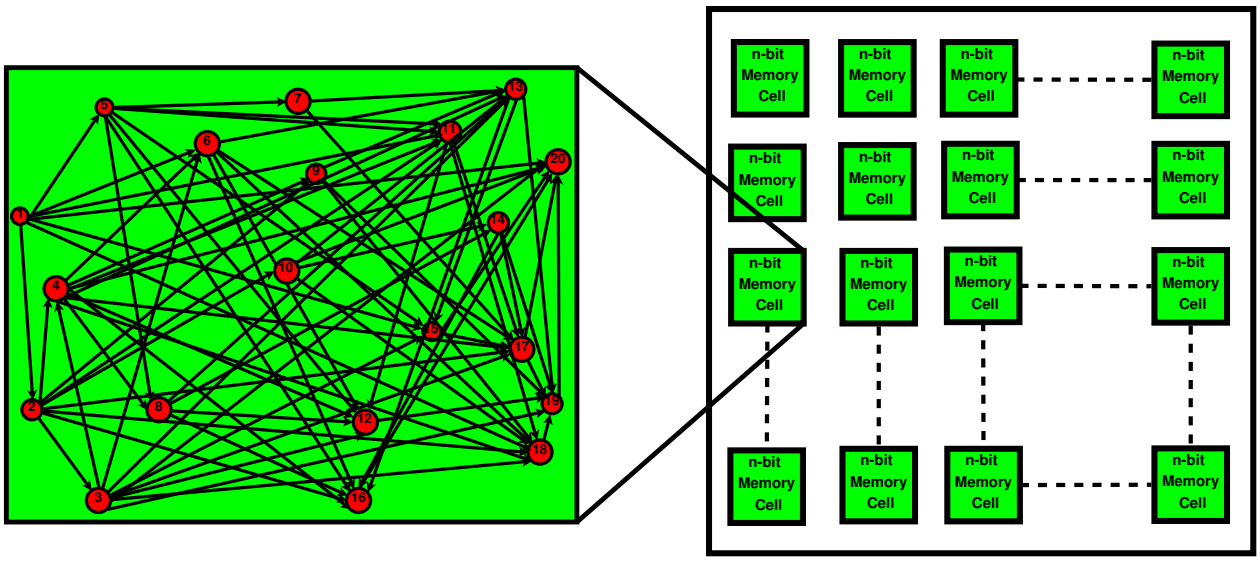

Fig. 5. Schematic to design multi-bit memory by using multiple n-bit Memory Cells (represented by green square box). Each Memory Cell consists of N nanoparticles (red circles) and M molecular switches (black arrows).

bit memory device. The output of Out port for $\{(w_1^1, r_1^1), (w_0^1, r_0^1), (w_1^2, r_1^2), (w_0^2, r_0^2)\}$ operations on bit 1 ($A1 = 2V$ and $A2 = 0.5V$) and then on bit 2 ($A1 = 0.5V$ and $A2 = 2V$) have been plotted. The noise margin high and low are assumed to be $NM_H = NM_L = 0.4$. As shown in Fig. 4, the voltage value on Out port for Read 1 (r_1^i) and Read 0 (r_0^i) is within this noise margin for both the bits ($i = 1\ and\ 2$). Applying this methodology, we have successfully trained up to 4-bit memory. Fig. 5 illustrates how, in general, n-bit memory cells could be used in mesh formation to construct multi-bit memory. In this figure, each block (named as n-bit Memory Cell) represents a nanocell having 20 nanoparticles and it has been trained as 2-bit memory device. We have trained a 64-bit memory by using thirty two such 2-bit memory cells, as shown in Fig. 5. The higher order memory devices can also be trained using similar concepts.

3. EXPERIMENTAL SETUP FOR MORTAL TRAINING

As said earlier, for on chip training of the nanocell, it would be impossible to switch the individual molecules to $'ON'$ or $'OFF'$ state. This is due to extremely small size of the molecules and the nanoparticles. Hence, the nanocell must be trained by mortal assumption. By mortal training, we mean to say that, the nanocell is assumed to be a black box to the training algorithm and voltage signals are applied externally to switch the state of the molecules. This methodology for mortal training of the nanocell using Genetic Algorithm (GA) is proposed here. Consider a nanocell consisting of 50 nanoparticles and 644 molecular switches. Suppose this nanocell is to be trained mortally to exhibit the behavior of a 2-bit memory device. The schematic of a 2-bit memory is shown in Fig. 6. As depicted from this figure, the internal molecular connections of the nanocell are assumed to be unknown for mortal training. The $Address\ 1$, $Address\ 2$, $Write/Read$, and $Data\_In$ are the input ports and $Data\_Out$ is the output port. There are twenty control voltage signals (CVS) denoted as $C_i,\ \forall\ i = 1\ to\ 20$ in the figure. A set of high or low voltage signals are applied to these CVS ports which are used to externally switch the state of molecules to $'ON'$ or $'OFF'$. The set of CVS values

ALGORITHM 1: Mortal training of n-bit memory using Genetic Algorithm

Input: $untrained\_mem\_spice$: an untrained sample of a nanocell;
$initial\_cs$: initial population of randomly generated set of k control voltage signal (CVS) values (known as chromosomes). Each gene value in the chromosome is either low ($V_{low} = 0.5V$) or high ($V_{high} = 2.0V$);
$fitness\_mem()$: fitness function for n-bit memory which defines the expected output voltage levels for Write/Read modes of n-bit memory;
$num\_generations$: number of generations;
$max\_generations$: maximum number of generations;
Output: $trained\_nbit\_memory$: n-bit memory device which is mortally trained using k control signals;
1: The Genetic Algorithm (GA) is initialized with initial population $initial\_cs$;
2: **for** $num\_generations$ = 1 *to* $max\_generations$ **do**
(i): The set of new individuals are used to modify the k CVS values in $untrained\_mem\_spice$ file;
(ii): This modified file is simulated using HSPICE and output voltage values for Read and Write operations for all n bits are saved in an array, $voltage\_values$;
(iii): Fitness function $fitness\_mem()$ of GA evaluates the output voltage level of n-bit memory using $voltage\_values$ obtained in previous step;
(iv): The GA quantifies the fitness of each individual (based on fitness function) and recombines them to generate new population;
end
3: The GA outputs optimal set of control voltage signals for which the nanocell behaves as a n-bit memory and outputs $trained\_nbit\_memory$ by utilizing these CVS values;
return $trained\_nbit\_memory$;

for which the nanocell behaves as a 2-bit memory is generated by utilizing the Algorithm 1. In this way, the nanocell is mortally trained to behave as a 2-bit memory.
The Algorithm 1 discusses the Genetic Algorithm (GA) based mortal training of the molecular memory. Initially, the set of nanoparticles connected to the control signal ports are chosen randomly, by applying Gaussian distribution with $\mu = 25$ and $\sigma = 20$. A DC voltage (i) $V_{low} = 0.5V$, or (ii) $V_{high} = 2.0V$, is applied to all these con-

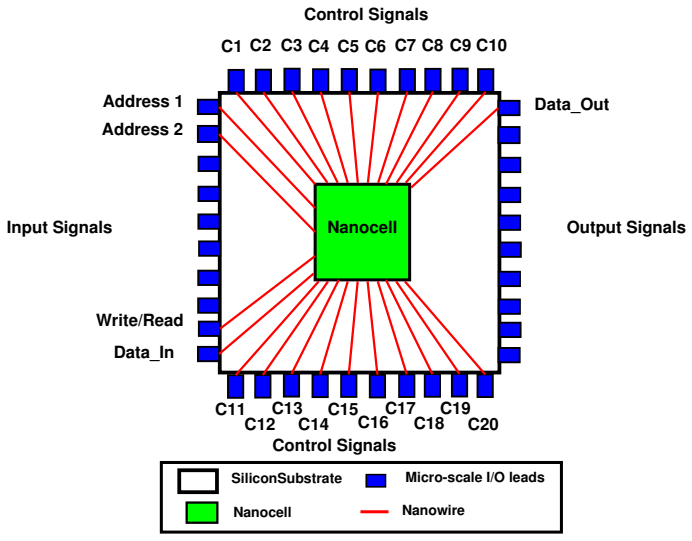

Fig. 6. Schematic of proposed mortally trained 2-bit memory device consisting of 20 control signals

trol voltage signals (CVS). The number of individuals (i.e. number of CVS) is kept constant throughout all generations. The GA trains the nanocell for a maximum of $max\_generations$ generations. The fitness function $fitness\_mem()$ computes whether the Data_Out voltage for read operation is obtained in acceptable noise margin or not. In the end, the GA converges to optimal set of CVS values for which the nanocell behaves as a n-bit memory. The simulation results of 2-bit memory device for read and write operations are similar to those obtained by omnipotent training, as shown in Fig. 4.

3.1. Design space exploration for mortal training of a 2-bit memory

ALGORITHM 2: Mortal training of n-bit molecular memory

Input: $untrained\_mem\_spice$: an untrained sample file of a nanocell;
num_cvs: number of CVS;
num_cvs_loc: number of nanoparticles to which CVS can be applied;
Output: MEM_nBIT: set of all n-bit memory devices;
1: **Generate_CVS**(num_cvs, 2);
{Generate a bipartition of all high and low voltage combinations that can be assigned to num_cvs control voltage signals and write all possible combinations to a folder (CVS_VALUES).} ;
2: **Generate_CVS_LOC**(num_cvs_loc, num_cvs);
{Generate all possible combinations of num_cvs_loc nanoparticles to which num_cvs control voltage signals can be connected and write all files to a folder ($LOCATIONS$).} ;
3: **Generate_memory_samples**($untrained\_mem\_spice$, $LOCATIONS$);
{Generate samples of $untrained\_mem\_spice$ by assigning CVS node number as generated in step 2 and save all files thus generated in folder ($MEM\_MODIFIED$). };
4: **Generate_spfiles**($MEM\_MODIFIED$, CVS_VALUES);
{For each memory file in the $MEM\_MODIFIED$ folder, assign CVS voltage values from CVS_VALUES folder. Save all memory samples to folder (SP_FILES). };
5: **Simulate**(SP_FILES);
{Simulate all memory samples using HSPICE and save all output files to folder (OUT_FILES).};
6: **Fetch_mem2bit**(OUT_FILES);
{Select all those memory samples for which n-bit memory read and write is done in acceptable noise margin using the results saved in OUT_FILES folder. Copy all spice files corresponding to these output files, from SP_FILES folder to a new folder (MEM_nBIT).};
return MEM_nBIT;

As a proof of concept, an experiment is preformed to analyze the mortal training approach in complete design space of a nanocell, consisting of N nanoparticles and M molecular switches. Suppose such a nanocell is to be trained for storing 2-bits of data. Then, as depicted from Fig. 6, there will be 5 input-output nodes and 5 or more CVS can be chosen from remaining $N-5$ nanoparticles. Let num_cvs and num_cvs_loc denote the number of CVS and the number of nanoparticles to which CVS can be assigned. Again, $num\_cvs\_loc \leq (N-5)$. So, a total of $C_{num\_cvs}^{num\_cvs\_loc}$ combinations in which num_cvs control voltage signals can be applied. Further, on these ports either DC voltage $V_{low} = 0.5V$ or $V_{high} = 2.0V$ is to be supplied in bipartition of num_cvs signals. For example, if there is a nanocell consisting of 20 nanoparticles and 6 CVS which can be connected to any of the 12 nanoparticles. Then, total combinations in which 6 CVS signals can be connected is $C_6^{12} = 924$ ($Generate\_CVS\_LOC(num\_cvs\_loc,\ num\_cvs)$) and total 31 bipartition's ($Generate\_CVS(num\_cvs,\ 2)$) in which high or low voltage is applied to CVSs. Using this, a total of 924 x 31 of modified spice files are simulated for

Table I. Number of successfully trained nanocell configuration for 2-bit memory read and write operation. Here, noise margin for Data read operation on output voltage node is considered as $0.4V$

No. of nano-particles per Nanocell (N)	No. of molecules (M)	No. of successfully trained nanocell instances with i CVS			
		6 CVS	8 CVS	10 CVS	12 CVS
20	97	2	0	0	0
20	104	1	0	0	0
30	223	27	13	5	0
30	214	33	19	4	0
40	399	103	55	13	2
40	392	98	43	10	1
Number of memory samples (cxp)		31x924	127x495	511x66	2047x1

$\{(w_1^i, r_1^i), (w_0^i, r_0^i)\}$ for both the bits ($i = 1\ and\ 2$). In the end, all spice files which successfully perform read and write operation on both bits are selected. These are the set of trained memory files for a given noise margin. The process is repeated by (i) varying the number of CVS ($6,\ 8,\ 10,\ 12$), and (ii) increasing the number of nanoparticles ($20,\ 30,\ 40$). This methodology is discussed in detail in Algorithm 2. In this algorithm, all directory names are written in capital and other variables in small, to distinguish between them. The functions (or sub-routines) are written in bold with a brief description beneath them. These functions utilize MATLAB, HSPICE and PERL for simulation as well as result fetching and result storage.

Now consider 3 nanocells, each consisting 20, 30 and 40 nanoparticles, respectively. Generate two instances of these nanocells using NRPA (Algorithm 3) as explained in Section 5. So, we have a total of six memory samples with varying number of nanoparticles. Now, for each memory sample, create four instances of each memory sample by varying number of CVS, namely, 6, 8, 10 and 12 CVS respectively. So, a total of 24 memory samples are generated. Then, for each case apply Algorithm 2 and obtain number of successfully trained memory samples. The total number of successfully trained 2-bit memory samples are listed in the Table I. The last row of this table depicts the total number of spice files simulated for 6, 8, 10 and 12 CVS, respectively. It is inferred from this table that to successfully train a 2-bit memory, the number of CVS must be approx. one-forth of the total number of nanoparticles.

4. WEAKLY DEFECT TOLERANT NANOCELL MEMORY: AN ANALYSIS

A nanocell may have multiple paths connecting input to output ports. At least one of the minimal path must be present between these ports. This is a necessary condition for correct functioning of the nanocell based device. It is sufficient to have multiple paths from inputs to output and some of which may or may not intersect. The presence of multiple paths introduce redundancy and increase probability of getting correct output.

Let us model the nanocell as a planar graph $G(V, E)$, where nanoparticles are the nodes and molecular switches in $'ON'$ state are the edges. The graph $G(V, E)$ is assumed to be a directional graph such that all the molecules are oriented in same direction, i.e. from input to the output. The primary input port is the root node and the primary output port is the leaf vertex of the graph $G(V, E)$. Consider a nanocell with $'N'$ nanoparticles and $'M'$ molecular switches. Assume that these nanoparticles are always present and molecular switches are distributed by Gaussian distribution within the nanocell. A molecular switch i is present in $'ON'$ state with probability p_i.

$$P\{X_i = 1\} = p_i, \quad \forall i = 1\ to\ M$$

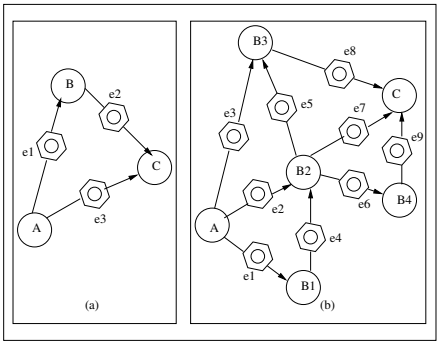

Fig. 7. A small network of nanoparticles and molecular switches (a) 3-particles, 3-switches, 2-paths (b) multiple particles and switches, multiple paths

This probability that i^{th} molecule is present in $'ON'$ state is called reliability of that molecule at that instant of time, denoted as $r(p_i)$. So, the probability that there is no edge (or connection) in between two nodes (or nanoparticles) is $(1 - p_i)$. Again, the probability that at least one edge is present between two nodes is given by:

$$P\{X_i \geq 1\} = 1 - (1 - p_i), \ \ \forall i = 1 \ to \ M$$

To exemplify proposed approach, a small nanocell is considered with three molecular switches ($e1$, $e2$ and $e3$) and three nanoparticles (A, B and C), as shown in Fig. 7(a). Here, input voltage is applied on A and received on C. This can be done via two minimal paths, namely, $A - e3 - C$ or $A - e1 - B - e2 - C$. Correct output will be received on C via path ABC if both molecules $e1$ and $e2$ are present. Similarly, data will be correctly received on C via path AC if the molecule $e3$ is in $'ON'$ state. These are the two redundant paths and at least one of them should be working correctly to obtain correct output. Thus, the probability of receiving correct data on node C is given by

$$\begin{aligned} P_{path} \ &= \ (\text{Probability that at least one molecule between} \\ &\quad \text{both AB and BC is present in 'ON' state)} \\ &\quad \text{or (Probability that at least one molecule} \\ &\quad \text{between AC is present in 'ON' state)} \\ &= \ 1 - (1 - p_{e1}p_{e2})(1 - p_{e3}) \\ &= \ p_{e3} + p_{e1}p_{e2} - p_{e1}p_{e2}p_{e3} \end{aligned}$$

Table II. Simulation results for an example nanocell configuration when none or some of the molecules are missing

Missing Molecules	V(low)	V(High)	P_{path}
None	0.4935	1.9935	0.625
e1	0.4951	1.9951	0.500
e2	0.4951	1.9951	0.500
e3	0.4902	1.9902	0.250
e1, e2	0.4951	1.9951	0.500
e2, e3	0	0	0.000
e1, e3	0	0	0.000
e1, e2, e3	0	0	0.000

In this example, the nanocell is working as a buffer. The low and high voltage applied on input node A are $0.5\ V$ and $2.0\ V$, respectively. Table II shows the low and high voltage values received on node C when either none or some of the molecules are missing. Out of eight test cases, correct output is received for five cases only. In other words, $5/8 = 0.625$ or there are 62.5% chances of getting correct output voltage. Theoretically, on substituting $p_{ei} = 0.5,\ \forall i = 1, 2, 3$ in the above equation, we get $P_{path} = 0.625$. Thus, our theoretical and experimental results are matching. The last three cases in the Table II denotes the minimal cut sets for this nanocell, denoted by, $C_1 = \{e_2, e_3\}$, $C_2 = \{e_1, e_3\}$ and $C_3 = \{e_1, e_2, e_3\}$. As depicted from Table II, incorrect output voltage is received for these cases.

Further, to evaluate the reliability of a trained nanocell, we assume that there are k redundant paths from input to output. Each of these paths may vary in length. The length of any path i can be represented by variable $l_i,\ \forall i = 1\ to\ k$. That is, each path consists of l_i molecules connected in series and such k paths are working in parallel. For correct functioning of the system, at least one of the i paths must function correctly. Consider an indicator variable x_{ji} which denotes the state of molecular switch j on path i.

$$x_{ji} = \begin{cases} '1' & \text{if molecule is 'ON',} \\ '0' & \text{otherwise.} \end{cases}$$

We define a structure function $\phi(x)_i$ for path i as

$$\phi(x)_i = \prod_{j=1}^{l_i} x_{ji} = \phi_i, \qquad \forall i = 1 \text{ to } k$$

Then, structure function of the whole system can be given as:

$$\begin{aligned} \phi(X) &= max(\phi_1, \phi_2, \ldots, \phi_k) \\ &= 1 - \prod_{i=1}^{k} (1 - \phi_i) \end{aligned}$$

Lets consider another example having multiple paths from input to output, Fig. 7(b). The node A is input and node C is output. The set of minimal paths from A to C are x_3x_8, $x_2x_5x_8$, $x_2x_6x_9$, $x_2x_6x_9$, x_2x_7, $x_1x_4x_7$, $x_1x_4x_6x_9$, $x_1x_4x_5x_8$. We can say that, at least one of the minimal path from A to C must be present to receive correct output at C. This can be defined as necessary condition for a workable nanocell. It is sufficient to have more than one path from A to C. The structure function for this system is given as:

$$\begin{aligned} \phi(x) &= max(x_3x_8, x_2x_5x_8, x_2x_6x_9, x_2x_7, x_1x_4x_7, \\ &\quad x_1x_4x_6x_9, x_1x_4x_5x_8) \\ &= 1 - (1 - p_3p_8)(1 - p_2p_5p_8)(1 - p_2p_6p_9)(1 - p_2p_7) \\ &\quad (1 - p_1p_4p_7)(1 - p_1p_4p_6p_9)(1 - p_1p_4p_5p_8) \end{aligned}$$

Hence, the reliability of the whole system at any instant of time is given as:

$$r = p\{\phi(X) = 1\}$$

If all molecules are assumed to be independent, the reliability of individual molecule $r(p_i),\ \forall i = 1\ to\ M$ can be expressed as r, i.e., $r(p) = r$.

4.1. Bounds on reliability of a nanocell

Let $A_1, A_2, \ldots, A_s$ denote minimal path sets connecting input node to the output node and we define F_i, $i = 1 \ldots s$ as F_i = {at least one molecular connection on path A_i has failed}.
By failing of the molecular connection, we mean to say that, molecule is in $'OFF'$ state. If at least one of the molecules in the minimal path set has failed, the system will fail eventually. Mathematically, it is denoted as:

$$\begin{aligned} 1 - r(p) &= P(F_1 F_2 \ldots F_s) \\ &= P(F_1)P(F_2|F_1)\ldots P(F_s|F_1 F_2 \ldots F_{s-1}) \end{aligned}$$

Henceforth, it can be easily derived that failure of at least one molecule in minimal path A_i can increase the probability that at least one molecule of A_j is failed. This would be the case if both paths A_i and A_j overlap. So,

$$P(F_j|F_i) \geq P(F_j)$$

Similarly,

$$P(F_i|F_1 F_2 \ldots F_{i-1}) \geq P(F_i)$$

Substituting in equation stated above we get,

$$1 - r(p) \geq \prod_i P(F_i)$$

or,

$$r(p) \leq 1 - \prod_i \left[1 - \prod_{j \in A_i} p_j \right]$$

Again, let $C_1 \ldots C_r$ denote the minimal cut sets. We define the events $E_1, \ldots, E_r$ by E_i = {at least one molecular device in C_i is functioning}. Since, the nanocell will function iff all of the events E_i occur, we say,

$$\begin{aligned} r(p) &= P(E_1, E_2 \ldots E_r) \\ &= P(E_1)P(E_2|E_1)\ldots P(E_r|E_1 \ldots E_{r-1}) \\ &\geq \prod_i P(E_i) \end{aligned}$$

Hence,

$$r(p) \geq \prod_i \left[1 - \prod_{j \in C_i} (1 - p_j) \right]$$

So, bounds on reliability function are given as:

$$\prod_i \left[1 - \prod_{j \in C_i} (1 - p_j) \right] \leq r(p) \leq 1 - \prod_i \left[1 - \prod_{j \in A_i} p_j \right]$$

Considering the same example as shown in Fig. 7(a), the reliability bounds are expressed as:

$$(1-(1-p_{e2})(1-p_{e3}))(1-(1-p_{e1})(1-p_{e3}))$$
$$(1-(1-p_{e1})(1-p_{e2})(1-p_{e3}))$$
$$\leq r(p) \leq 1-(1-p_{e1}p_{e2})(1-p_{e3}))$$

Substituting $p_{ei} = 0.5 \quad i = 1, 2, 3$ we get,

$$0.4922 \leq r(p) \leq 0.6250$$

The reliability bounds match the values computed in column 4 of Table II. This example can be generalized for nanocells consisting of more than three nanoparticles and molecules and similar results can be obtained.

5. EXPERIMENTAL SETUP FOR RELIABILITY PREDICTION

A Nanocell Reliability Predication Algorithm (NRPA) is proposed in this paper and implemented in MATLAB. This Algorithm 3 computes set of all minimal paths connecting the input node to the output node of a nanocell and return as an output, the probability P_{path} of at least one path being present between input and output node. The NRPA algorithm is based on probabilistic analysis of nanocell, as discussed in Section 4. The parameters such as number of nanoparticles (N), input (IP) and output (OP) terminals of the nanocell and the probability of presence of a molecule and subsequently of being found in $'ON'$ state between two nanoparticles (P_{mol}) are given as an input to the algorithm. Firstly, a directed acyclic graph of the nanocell is generated. The nodes of the graph are nanoparticles and directed edges are the molecules, which points from the input to the output node. It is assumed that the molecules are distributed with Gaussian Distribution within the nanocell. Further, the Dijkstra's shortest path algorithm is modified and utilized to find set of all paths from the input node to the output node. Thereafter, the probability $P_{mol} = 0.5$ is assigned to each molecule. This NRPA algorithm finally computes the probability that at least a path is present between in-

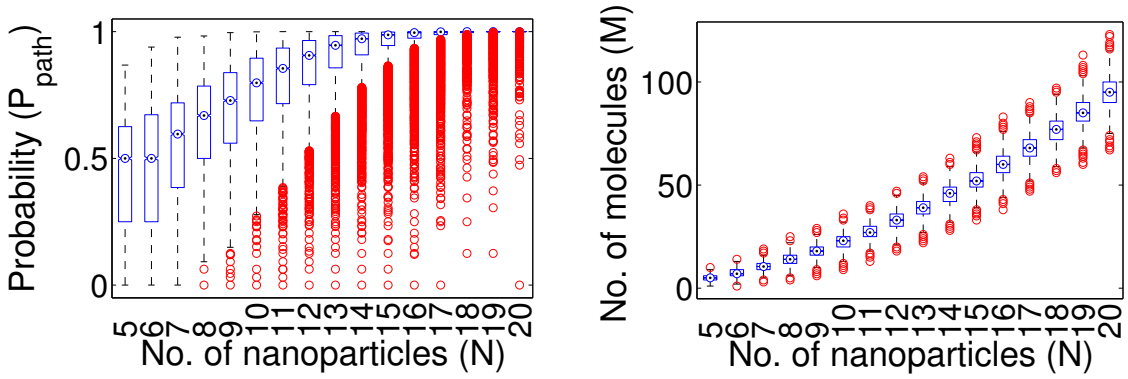

Fig. 8. Simulation results for samples generated using NRPA algorithm (a) Probability of existence of at least one minimal path in a nanocell (P_{path}) (b) Number of molecules (M) in a nanocell. Here, (i) the central mark is the median (ii) the edges of the box are 25^{th} and 75^{th} percentiles (iii) the whisker extend to the most data points which are not considered as outliers (iv) the unfilled dots represent the outliers

ALGORITHM 3: Nanocell Reliability Prediction Algorithm (NRPA)

Input: N := Number of Nanoparticles;
IP := Primary Input Node ;
OP := Primary Output Node ;
P_{mol} := Probability of a molecule being present between two nanoparticles and found in 'ON' state ;
Output: P_{path} := Probability that at least a path exists between IP and OP;
1: Initialization ;
2: Adj_Matrix = **Generate_matrix(**N,N,μ,σ**)** ;
{Generate a NxN random array (Adjacency Matrix) with Gaussian Distribution } ;
3: $Edge\_Matrix$ = **Convert_to_EdgeMatrix(**Adj_Matrix**)** ;
{Convert this Adjacency Matrix to Edge Matrix} ;
4:$Fwd\_pointing\_array$ = **Generate_DAG(**$Edge\_Matrix$**)** ;
{Remove self loops and backward pointing edges from this matrix.} ;
5: M = **size(**$Fwd\_pointing\_array$,1**)** ;
{Calculate Number of Molecules, denoted as 'M'} ;
6: $Nanocell\_Mat[N][N]$ = **Weighted_Matrix(**$Fwd\_pointing\_array$,P_{mol}**)** ;
{Add weight to each connected edge = P_{mol} and unconnected edge = ∞ of $Fwd\_pointing\_array$} ;
7: $shortest\_path$ = **Dijkstra_k(**$Nanocell\_Mat$,IP,OP,k**)** ;
{Calculate k shortest paths from IP to OP using modified Dijkstra algorithm} ;
8: **for** i = 1 *to* M **do**
  (i): $P_M(i)$ = P_{mol} ;
end
{Assign probabilities to each molecule} ;
9: num_paths = **size(**$shortest\_path$, 2**)** ;
10: **for** num = 1 *to* num_paths **do**
  (i): a = $shortest\_path\{1, num\}$;
  (ii): pa=1 ;
  (iii): **for** i=1 *to* ***(size(***a,2*)***-1*) **do**
    (a): pa=$pa * P_M(a(1, i))$;
  end
  (iv): $P(num)$=pa ;
  (v): $P_c(num)$=$1 - pa$;
end
{Calculate probability for all shortest paths} ;
11: ϕ_x=1 ;
12: **for** z = 1 *to* num_paths **do**
  (i): ϕ_x=ϕ_x x $P_c(z)$;
end
13: P_{path} = 1 - ϕ_x ;
{Calculate probability P_{path} of at least one path from input node to output node of a nanocell} ;
return *[*P_{path}*]* ;

put and output node of the nanocell device, denoted as P_{path}.
A single input single output memory element is considered as a test case for exemplifying the algorithm. The number of nanoparticles are increased from 1 to 20. For each case, 10000 samples are generated using NRPA algorithm. It has been observed that P_{path} approaches to unity with increase in number of connected paths between input and output nodes. The number of connected paths depend on N and M. With 20 nanoparticles, P_{path} is close to one for majority of the samples. The simulation results for nanoparticles from 5 to 20 are summarized as boxplots in Fig. 8. Further, Table III depicts the simulation results for 1 to 20 nanoparticles. Theoretically, it can be proved from (9.10) of [Ross 2007] that the probability of a random graph with n nodes to be

Table III. Simulation results for path probability (P_{path}) and number of molecules (M) w.r.t. number of nanoparticles (N) in a nanocell

No. of nano-particles per Nanocell (N)	Probability that atleast a path exists (P_{path})			No. of molecules per Nanocell (M)		
	min.	avg.	max.	min.	avg.	max.
1	0.0000	0.0000	0.0000	0	0	0
2	0.0000	0.5000	0.5000	0	1	1
3	0.0000	0.3434	0.6250	1	2	3
4	0.0000	0.3654	0.7542	1	3	6
5	0.0000	0.4111	0.8679	1	5	10
6	0.0000	0.4714	0.9390	1	7	14
7	0.0000	0.5412	0.9781	3	10	19
8	0.0000	0.6100	0.9832	4	14	25
9	0.0000	0.6712	0.9961	6	18	29
10	0.0000	0.7434	0.9999	9	22	36
11	0.0000	0.7969	1.0000	13	27	40
12	0.0000	0.8473	1.0000	18	32	47
13	0.0000	0.8909	1.0000	22	40	54
14	0.0000	0.9234	1.0000	28	45	63
15	0.0000	0.9472	1.0000	33	52	73
16	0.0000	0.9669	1.0000	38	60	83
17	0.0000	0.9783	1.0000	47	70	90
18	0.1250	0.9871	1.0000	56	76	97
19	0.1250	0.9924	1.0000	60	85	113
20	0.0000	0.9952	1.0000	67	95	123

connected is given by:

$$1 - P_n \approx nq^{n-1} \quad as \quad n \to \infty$$

For $n = 20$ and $q = 0.5$, the probability P_n is computed as 0.99998.

6. CONCLUSION AND FURTHER WORK

In this manuscript, we have proposed the design methodology for a nanocell based memory, utilizing circuit behavior of an OPE molecule. This nanocell is explored being trained in two different ways to exhibit the behavior of a multi-bit memory device - (i) omnipotently as well as (ii) mortally trained using genetic algorithm. Such a memory possesses read, write and erase capability. The simulation results of a 1-bit memory are compared against analytical probabilistic modeling approach for the nanocell based memory devices. The presence of at least one minimal path from input to output is a necessary condition for correct functioning of the device. Again, it is sufficient to have multiple minimal paths to obtain the desired output. These multiple paths add redundancy to the nanocell device and thus make it defect tolerant. The mortal training approach proposed here can be used to train logic gates and combinational circuits. The proposed methodology is flexible enough to design multi-bit memory device. However, Genetic Algorithm requires high convergence time, even for training a small nanocell of 50 nanoparticles. Thus, some other adaptive learning algorithms are to be explored to reduce the training time. The work on mortal training of a large memory is currently under progress.

REFERENCES

CHEN, J., REED, M. A., RAWLETT, A. M., AND TOUR, J. M. 1999. Large on-off ratios and negative differential resistance in a molecular electronic device. *Science 286,* 5444, 1550–1552.

CHEN, J., WANG, W., REED, M. A., RAWLETT, A. M., PRICE, D. W., AND TOUR, J. M. 2000. Room-temperature negative differential resistance in nanoscale molecular junctions. *Applied Physics Letters 77,* 8.

CHEN, Y., JUNG, G.AND OHLBERG, D. A. A., LI, X., STEWART, D. R., JEPPESEN, J. O., NIELSEN, K. A., FRASER STODDART, J., AND WILLIAMS, R. S. 2003. Nanoscale molecular-switch crossbar circuits. *Nanotechnology 14,* 4, 462–468.

CUI, Y. AND LIEBER, C. 2001. Functional nanoscale electronic devices assembled using silicon nanowire building blocks. *Science 291,* 5505, 851–853.

DEHON, A. 2005. Nanowire-based programmable architectures. *ACM Journal on Emerging Technologies in Computing Systems (JETC) 1,* 2, 109–162.

GREEN, J. E., WOOK CHOI, J., BOUKAI, A., BUNIMOVICH, Y., JOHNSTON-HALPERIN, E., DEIONNO, E., LUO, Y., SHERIFF, B. A., XU, K., SHIK SHIN, Y., TSENG, H.-R., STODDART, J. F., AND HEATH, J. R. 2007. A 160-kilobit molecular electronic memory patterned at 1011 bits per square centimetre. *Nature 445*, 414–417.

HUANG, Y. 2001. Logic gates and computation from assembled nanowire building blocks. *Science 294,* 5545, 1313–1317.

HUSBAND, C., HUSBAND, S., DANIELS, J., AND TOUR, J. 2003. Logic and memory with nanocell circuits. *Electronic Devices, IEEE Transactions on 50,* 9, 1865 – 1875.

REED, M. A., CHEN, J., RAWLETT, A. M., PRICE, D. W., AND TOUR, J. M. 2001. Molecular random access memory cell. *Applied Physics Letters 78,* 23.

ROSE, G., YUXING, Y., T., J. M., ADAM, C. C., NADINE, G., NABANITA, M., JOHN, C. B., LLOYD, R. H., AND MIRCEA, R. S. 2007. Designing cmos/molecular memories while considering device parameter variations. *ACM Journal on Emerging Technologies in Computing Systems (JETC) 3,* 1.

ROSE, G., ZIEGLER, M., AND STAN, M. 2004. Large-signal two-terminal device model for nanoelectronic circuit analysis. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 12,* 11, 1201 –1208.

ROSS, S. M. 2007. *Introduction to Probability Models* ninth Ed. Academic Press.

SKOLDBERG, J., ONNHEIM, C., AND WENDIN, G. 2007. Nanocell devices and architecture for configurable computing with molecular electronics. *Circuits and Systems, IEEE Transactions on 54,* 11, 2461 –2471.

STAN, M., FRANZON, P., GOLDSTEIN, S., LACH, J., AND ZIEGLER, M. 2003. Molecular electronics: from devices and interconnect to circuits and architecture. *Proceedings of the IEEE 91,* 11, 1940–1957.

TOUR, J., VAN ZANDT, W., HUSBAND, C., HUSBAND, S., WILSON, L., FRANZON, P., AND NACKASHI, D. 2002. Nanocell logic gates for molecular computing. *Nanotechnology, IEEE Transactions on 1,* 2, 100 – 109.

WU, W., JUNG, G.-Y., OLYNICK, D., STRAZNICKY, J., LI, Z., LI, X., OHLBERG, D., CHEN, Y., WANG, S.-Y., LIDDLE, J., TONG, W., AND WILLIAMS, R. S. 2005. One-kilobit cross-bar molecular memory circuits at 30-nm half-pitch fabricated by nanoimprint lithography. *Applied Physics A: Materials Science and Processing 80*, 1173–1178.

ZHONG, Z., WANG, D., CUI, Y., BOCKRATH, M., AND LIEBER, C. M. 2003. Nanowire crossbar arrays as address decoders for integrated nanosystems. *Science 302,* 5649, 1377–1379.

ZIEGLER, M. M. AND STAN, M. R. 2002. A case for cmos/nano co-design. In *ICCAD*. 348–352.