

A Probabilistic Approach to Nano-computing

J. Chen, J. Mundy, Y. Bai, S.-M. C. Chan, P. Petrica, and R. I. Bahar
Division of Engineering, Brown University, RI 02912, USA

ABSTRACT

As current silicon-based techniques fast approach their practical limits, the investigation of nanoscale electronics, devices and system architectures becomes a central research priority. It is expected that nano-architectures will confront devices and interconnections with high inherent defect rates, which motivates the search for new architectural principles.

In this paper, we propose a probabilistic-based design methodology for designing nano-scale computer architectures based on Markov Random Fields (MRF). The MRF can express arbitrary logic circuits and logic operation is achieved by maximizing the probability of state configurations in the logic network. Maximizing state probability is equivalent to minimizing a form of energy that depends on neighboring nodes in the network. Once we develop a library of elementary logic components, we can link them together to build desired architectures based on the belief propagation algorithm. Belief propagation is a way of organizing the global computation of marginal belief in terms of smaller local computations. Finally, we will illustrate the proposed design methodology with some elementary logic examples.

1. INTRODUCTION

The amazing success of computing over the past forty years is based in part on advances in the fabrication of CMOS-based integrated circuits, and both engineers and consumers have come to expect and plan for exponential increases in system performance over time. However, a number of physics and economic factors threaten the continued scaling of CMOS devices, which motivating research into computing systems based on other technologies such as molecular electronics, biological processes.

Nanoscale devices, whether assembled lithographically or chemically, will have a high probability of failure. Therefore, any architecture built from large numbers of nanoscale devices will necessarily contain a large number of defects, which fluctuate on time scales comparable to the computation cycle. The challenge for computer engineers is to develop an architecture that is dynamically *defect tolerant*. An example of a defect tolerant system is one based on reconfigurable structures. In this case the architecture contains many redundant components and must detect faults (as well as the location of the faults) and reconfigure the system around these faulty cells. This essentially is the approach used in the Nano Fabric scheme [4].

Reconfiguration is attractive in the sense that devices will be cheap and plentiful so we can always count on having “extra devices lying around” available to re-route or re-wire the needed computation correctly, thus providing reliability via redundancy. On the other hand, reconfiguring around faults requires some means of detecting faults, both statically or dynamically, and re-wiring as needed. If the fault detection logic is on chip, then there needs to be a means to reliably guarantee that this logic itself is not faulty. Finally, this approach would not easily cope with rapidly changing

fault configurations.

The other approach is synthetic neural nets. A single-electron latching switches have been proposed as nanoscale synapses [3], which seems a perfect application for hardware implementation of synthetic neural nets. Those neural network systems achieve high performance via the adaptive interconnection of simple switching elements that process information in parallel. Arrays of simple neural processing elements show features such as association, fault tolerance and self-organization. However, neural network style architectures require training, and it is difficult to analyze or optimize their performance according to engineering principles. It is not clear how their behavior generalizes to new computational examples.

Inspired by Von Neumann’s pioneering work [10], we propose to take an alternative approach to fault tolerance. Von Neumann asserted that device failure should not cause computing systems to malfunction if they have been designed from the beginning to tolerate faults. He proposed a majority voting scheme to overcome logic errors and a *randomizer* to scramble a cluster of errors so that they do not dominate any one logic circuit. Using similar principles, in this paper we propose a probabilistic approach for nano-scale computing. Our approach adapts to errors as a natural consequence of probability maximization, thereby removing the need to actually detect faults.

There are two aspects of fault tolerance that must be considered: 1) tolerance due to structural failure and 2) tolerance due to signal noise. The first error type will result in variation in the clique energy coefficients of the auto-model. The second type of error is directly accounted for in the probability maximization process inherent in MRF processing. In what follows, we illustrate the behavior of the model for both types of error. The proposed approach is presented in section 2. In section 3, we show numerical examples to demonstrate our design. The paper is finally concluded in section 4.

2. OUR PROPOSED APPROACH

2.1 Nano-scale Devices

Although the primary emphasis in this paper is on an abstract model of computation, the Markov random network, assumptions about key properties of the model are based on realistic nanodevice characteristics.

Specifically, we have selected the Y-Junction carbon nanotube (Y-CNT) as the basis for our architectural study. The Y-shaped CNTs are produced by chemical vapor deposition (CVD) growth in branched nanochannel alumina templates [6] as shown in Fig. 1. An attractive characteristic of the Y-CNT is the Y junction that acts as a pair of diodes as shown in Fig. 2. CNTs are p-type semiconductors; however, Martel *et al.* demonstrate that the semicon-

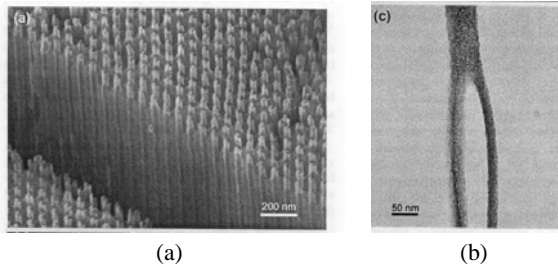


Figure 1: Scanning electron micrography of (a) a Y-shaped carbon nanotube array and (b) a Y-shaped carbon nanotube (After Papadopoulos).

ducting properties of a CNT can be altered to n-type by suitable annealing [8]. They also demonstrate that an operational CMOS process is feasible in an integrated device as shown in Fig. 3. The feasibility of active devices as a natural characteristic of carbon nanotubes provides a rich source of design possibilities. In addition, the regular structure of the Y-CNTs allows us to consider various approaches for connecting devices together, or creating logic functions through connections.

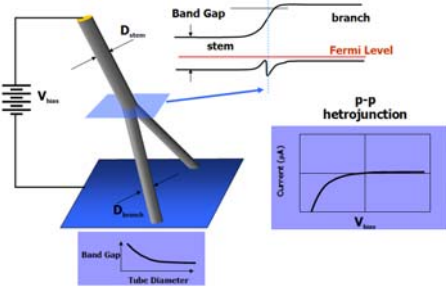


Figure 2: The band-gap of a CNT varies in a continuous manner with tube diameter, as shown on the lower left. The Y nanotube on the left forms a heterojunction at the branch point, as illustrated by the band structure in the upper right. The I-V characteristics of the diode is shown on the lower right (After Li).

2.2 The Markov Random Network

In this paper, we propose a probabilistic-based design methodology for designing CNT-based computer architectures based on Markov Random Fields (MRF). Before proceeding further, we will briefly introduce the MRF and its properties.

The basis for our architectural approach is the Markov random network, a network embodiment of the mathematical concept, the Markov random field (MRF) [7]. The Markov random field defines a set of random variables, $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_k\}$. Each variable σ_i can take on various values, e.g. state labels. Associated with each variable, σ_i , is a neighborhood, \mathcal{N}_i , which is a set of variables from $\{\Sigma - \sigma_i\}$. Simply put, the probability of a given variable depends only on a (typically small) neighborhood of other variables. In our model, the variables represent states of nodes in a network. The arcs or edges in the network convey the conditional probabilities with respect to the neighboring nodes.

The definition of the MRF is as follows:

$$P(\sigma) > 0, \forall \sigma \in \Sigma \quad (\text{Positivity}) \quad (1)$$

$$P(\sigma_i | \{\Sigma - \sigma_i\}) = P(\sigma_i | \mathcal{N}_i) \quad (\text{Markovianity}) \quad (2)$$

The conditional probability of a node state in terms of its neigh-

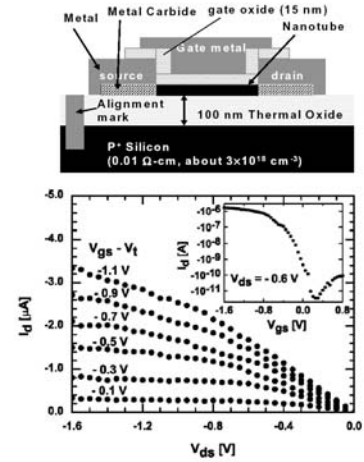


Figure 3: The structure of the CNT field-effect transistor(FET) is illustrated at the top. A layer of oxide insulates the tube from the gate. Source and drain electrode connect to each end of the tube. The resulting FET characteristics are shown at the bottom (After Martel et al.)

borhood can be formulated in terms of cliques. A neighborhood and two clique examples are shown in Figure 4. It can be shown, due to the the Hammersley-Clifford theorem [1], that

$$P(\sigma_i | \{\Sigma - \sigma_i\}) = \frac{1}{Z} e^{-\frac{1}{T} \sum_{c \in \mathcal{C}} U_c(\sigma)}. \quad (3)$$

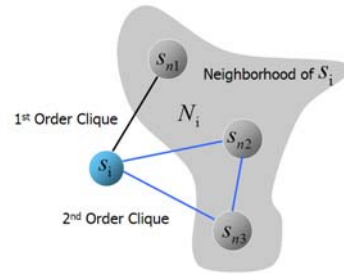


Figure 4: The neighborhood of a network node. The transition probabilities can be expressed in terms of node cliques.

This form for the probability is called the *Gibbs distribution*. The normalizing constant Z is called the partition function and insures that P is in the range $[0, 1]$. The set \mathcal{C} is the set of cliques for a given node, i . The function U_c is called the clique energy function and depends only on the nodes in the clique. The constant T is analogous to temperature in a physical model of energy states. Note that the probability of states are uniform at high values of T and becomes sharply peaked at low values of T . This form mimics the annealing behavior of physical systems.

This Gibbs formulation of the Markov random field is an attractive representation for computation, since the physical interpretation of the probabilities in terms of energy and temperature is likely to find ready interpretation in the physical device characteristics. In the next section, we briefly outline a possible mapping of the Markov network onto CNT nano-devices.

2.3 Mapping MRF to CNTs

Mapping the Markov random network onto CNTs, requires three essential functional elements: *weighted connections*, *clique energy*

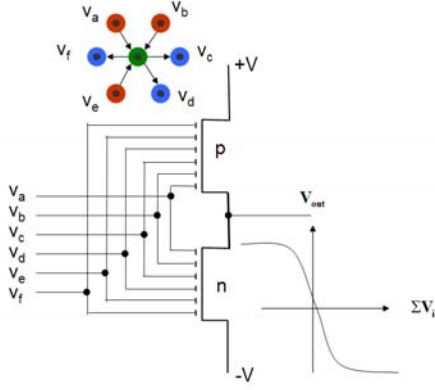


Figure 5: A realization of weighted summation using CNT devices.

summation, and *probability maximization*. The weighted connections can be achieved by using multiple nano-tube paths for the same functional weighted inputs. The sign of the weight corresponds to a positive or negative voltage applied to the connections. A significant advantage to using this redundant path weighting scheme is that there can be many bad connections, and the correct states will still have the highest probability.

Our idea for summation is to use the CMOS FET device capability of CNTs and embed it onto the regular CNT array as shown in Fig 5. The CNTs that are adjacent to the center tube each act as a gate and partially control its current. Let us assume that alternate layers of a 3-D stacked configuration have opposite type semiconductor properties, i.e., first n type and then p type. Thus we can form a CMOS inverter across the adjacent layers. The net current through each FET will be a result of the summation of the multiple gate voltages. The output will reflect the balance of these currents and be the inverse of the net input summation.

Achieving the correct state configuration in the network corresponds to propagating state values through the network and updating each node assignment with a node state having the maximum probability. The great advantage of the Markov network model is that this probability is maximum when the total clique energy is a minimum. This energy minimization can be achieved by a device or device configuration that produces a bi-stable energy function. A binary flip-flop circuit possesses this desired energy behavior where the required asymmetry of state energy is created by the summing mechanism just described.

Our exploration of the CNT device characteristics has just begun, and it is likely that a more suitable persistent state mechanism will be found in terms of inherent physical device characteristics. We note that it is likely that devices will be operated just slightly above the thermal energy level, which provides a natural probability minimization mechanism, through simulated annealing [5].

2.4 Markov Random Field Computation

The previous discussion has suggested an approach to embedding a Markov random network in CNT circuits. In this section we briefly describe the nature of computation in the MRF framework. The general algorithm for finding individual site labels that maximize the probability of the overall network is called *Belief Propagation* (BP) [11] and provides an efficient means of solving inference problems by propagating marginal probabilities through the network. There are three essential probability functions:

Joint probability:

$$p(x_0, x_1, \dots, x_{n-1})$$

Marginal probability:

$$p(x_i) = \sum_{x_0} \sum_{x_1} \dots \sum_{x_j} \dots \sum_{x_{n-1}} p(x_0, x_1, \dots, x_{n-1}), j \neq i$$

Conditional probability:

$$p(x_0, x_1, \dots, x_{i-1}, x_{i+1}, \dots | x_i) = \frac{p(x_0, x_1, \dots, x_{n-1})}{p(x_i)}$$

The basic idea of belief propagation is that the probability of state labels at a given node in the network can be determined by marginalizing (summing) over the joint probabilities for the node state given just the probabilities for site labels in the Markov neighborhood, \mathcal{N}_i as shown in Fig. 4. (In our design, we can think about the node as inputs/outputs of nano-scale circuit). We can classify the nodes in the network into those that have defined label probabilities, and those whose values must be determined by the propagation algorithm. The first node type would correspond to a computational input whose value is constrained by the problem setup. Such nodes are called *observable* nodes. The other nodes are called *hidden* nodes. We refer to marginal probabilities that are computed approximately as *beliefs*, and denote the belief at node i by $b(x_i)$ [11].

In MRF, we can consider the *observable* nodes y_i to be fixed, write $\phi_i(x_i)$ as a short-hand for $\phi_i(x_i, y_i)$ (here x_i is the *hidden* nodes). We further assume that there is some statistical dependency between x_i and y_i at each position i , which we write as a joint probability $\phi_i(x_i, y_i)$. The function $\phi_i(x_i, y_i)$ is often called the *evidence* for x_i [11]. For us to possibly be able to infer anything about the our nano-scale computer architecture, there has to be some structure to the x_i . We encode the assumed structure by saying that variable x_i should, insofar as possible, be “compatible” with nearby variable x_j , as represented by a compatibility function $\psi_{ij}(x_i, x_j)$, where ψ_{ij} only connects nearby positions. We then take the joint probability distribution for the unknown variables x_i as:

$$P(\{x\}) = \frac{1}{Z} \prod_{(i,j)} \psi_{ij}(x_i, x_j) \prod_i \phi_i(x_i)$$

where Z is a normalization constant [11].

This marginalization establishes the label probabilities for the next propagation step. It can be shown that this propagation algorithm will converge to the maximum probability site label assignment for the entire network, provided there are no loops [11]. This incremental algorithm has computational complexity on the order of the number of nodes in the network, with a weighting term proportional to the size of the neighborhood. In the case of loops, the marginalization must be done combinatorially over a region of the network that bounds the loops in order to guarantee maximum probability solutions. That is, one would partition the network into a loop-free network of blocks which internally contain loops. However it has been demonstrated that the belief propagation algorithm usually converges to the maximum probability state even in the presence of loops [11].

We will give examples of the belief propagation process in a later section.

3. NUMERICAL EXAMPLES

The MRF is a completely general computational framework and in principle any type of computation could be mapped onto the model. In order to illustrate the operation of the model, we will

use combinatorial logic as an example. The *programming* of the MRF is straightforward in this case, and will permit some analysis of the fault tolerance of the architecture.

Combinatorial logic can be implemented using a simple, yet powerful, form for the clique energy, called the *auto-model*. For cliques up to order three, the energy function is given by:

$$U_c(\sigma) = \kappa + \sum_{i \in C_0} \alpha_i \sigma_i + \sum_{i,j \in C_1} \beta_{ij} \sigma_i \sigma_j + \sum_{i,j,k \in C_2} \gamma_{ijk} \sigma_i \sigma_j \sigma_k.$$

The constants, α_i, β_{ij} and γ_{ijk} are called *interaction coefficients*. The constant κ acts an energy offset. This form for $U_c(\sigma)$ has been used in many MRF applications including image segmentation, texture classification and object recognition [2] [9]. To illustrate our design methodology, we will now present a half-adder example. Its range of operation over a distribution of device and connection faults and signal faults as well will be analyzed.

3.1 Mapping Logic onto MRF

The effect of errors on the coefficients in the clique energy is illustrated using a half-adder example. There are four nodes in the network: the inputs x_0, x_1 , the sum, x_2 , and the carrier x_3 of the gate (Here we don't consider the carrier from the previous stage.). The successful operation of the gate is designated by the compatibility function, $f(x_0, x_1, x_2, x_3)$ as shown in Figure 6.

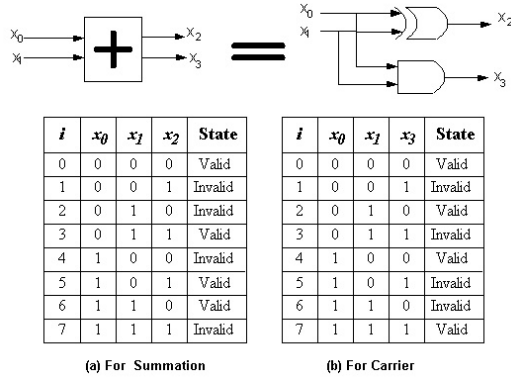


Figure 6: The logic compatibility function for a half-adder gate.

In order to relate the logic compatibility function to a Gibbs energy form it is necessary to use the axioms of the *Boolean ring*. The Boolean ring expresses the rules of symbolic Boolean logic in terms of algebraic manipulations as follows:

$$\begin{aligned} \sim X &\rightarrow (1 - X) \\ X_1 \wedge X_2 &\rightarrow X_1 X_2 \text{ (multiplication)} \\ X_1 \vee X_2 &\rightarrow X_1 + X_2 + X_1 X_2. \end{aligned}$$

The logic variables are treated as real value algebraic quantities and logic operations are transformed into arithmetic operations. Additionally, it is desired that valid input/output states should have lower clique energies than invalid states. Thus, the clique energy expression is obtained by the negative sum over minterms from valid states,

$$U(x_0, x_1, x_2) = - \sum_i f_i(x_0, x_1, x_2),$$

where $f_i = 1$, and the minterms are transformed using the Boolean ring rules. Note that this form exploits the simplification that cross-products of minterms vanish. The Boolean ring conversion for the

minterm $(x_0, x_1, x_2) = 000$ is,

$$\begin{aligned} \sim x_0 \wedge \sim x_1 \wedge \sim x_2 &= (1 - x_0)(1 - x_1)(1 - x_2) \\ &= (1 - x_0 - x_1 + x_0 x_1)(1 - x_2) \\ &= 1 - x_0 - x_1 - x_2 + x_0 x_1 + x_0 x_2 + x_1 x_2 - x_0 x_1 x_2 \end{aligned}$$

For instance, let us take the exclusive-or portion of a half-adder for summation calculation to illustrate logic mapping onto MRF. By summing over the valid states,

$$\begin{aligned} &000((1 - x_0)(1 - x_1)(1 - x_2)), \quad 011((1 - x_0)x_1x_2), \\ &101(x_0(1 - x_1)x_2), \quad 110(x_0x_1(1 - x_2)) \end{aligned}$$

as shown in Fig. 6, we can compute the clique energy as following:

$$\begin{aligned} U &= -(1 - x_0)(1 - x_1)(1 - x_2) - (1 - x_0)x_1x_2 - \\ &\quad x_0(1 - x_1)x_2 - x_0x_1(1 - x_2) \\ &= -1 + x_0 + x_1 + x_2 - 2x_0x_1 - 2x_0x_2 - 2x_1x_2 + \\ &\quad 4x_0x_1x_2. \end{aligned} \quad (4)$$

3.1.1 Structural Errors

Nanoscale devices, whether assembled lithographically or chemically, will have a high probability of failure. Therefore, any architecture built from large numbers of nanoscale devices will necessarily contain a large number of defects or *structural errors*, which fluctuate on time scales comparable to the computation cycle. If we take the structural errors into our design consideration, the clique energy in Eqn.4 can be rewritten as:

$$U = \kappa + Ax_0 + Bx_1 + Cx_2 - 2Dx_0x_1 - 2Ex_0x_2 - 2Fx_1x_2 + 4Gx_0x_1x_2. \quad (5)$$

where κ is a constant, and the nominal *weight* values for the coefficients are: $A = B = C = D = E = F = G = 1$, as derived above for the error free case.

In the modified equation above, the energy coefficients have been replaced by variables to indicate that their values can deviate from the ideal setting due to failures. The variables A, B, C stand for the first-order clique energy coefficients, and D, E, F are second coefficients. The third order coefficient, G , constrains the values of all the lower order coefficients as will be shown shortly. In the nano-architecture being described here, the coefficient values are determined by a set of gate connections, so coefficient error is caused by connection failure.

valid state	energy relation to invalid states	
000	κ	$< \kappa + C$
	κ	$< \kappa + B$
	κ	$< \kappa + A$
	κ	$< \kappa + A + B + C - 2D - 2E - 2F + 4G$
011	$\kappa + B + C - 2F$	$< \kappa + C$
	$\kappa + B + C - 2F$	$< \kappa + B$
	$\kappa + B + C - 2F$	$< \kappa + A$
	$\kappa + B + C - 2F$	$< \kappa + A + B + C - 2D - 2E - 2F + 4G$
101	$\kappa + A + C - 2E$	$< \kappa + C$
	$\kappa + A + C - 2E$	$< \kappa + B$
	$\kappa + A + C - 2E$	$< \kappa + A$
	$\kappa + A + C - 2E$	$< \kappa + A + B + C - 2D - 2E - 2F + 4G$
110	$\kappa + A + B - 2D$	$< \kappa + C$
	$\kappa + A + B - 2D$	$< \kappa + B$
	$\kappa + A + B - 2D$	$< \kappa + A$
	$\kappa + A + B - 2D$	$< \kappa + A + B + C - 2D - 2E - 2F + 4G$

Table 1: The inequalities that must hold among the energy coefficients for successful gate operation.

For successful operation of the logic, it is necessary that the energy of correct logic state configurations always be *less* than invalid state configurations.

LEMMA 1. *For combination logic, the energy of correct logic state is always less than that of invalid state by a constant.*

Proof: For example, in simple exclusive-or design shown in Fig. 6, the clique energy is

$$U = -1 + x_0 + x_1 + x_2 - 2x_0x_1 - 2x_0x_2 - 2x_1x_2 + 4x_0x_1x_2.$$

By substituting the invalid and valid states into this energy equation, we get that the energy for valid states is always ‘-1’ while that of invalid states is always ‘0’. The energy difference is a constant (in this case, it is one).

The reason is embedded in our clique energy definition:

$$U(x_0, x_1, x_2) = - \sum_i f_i(x_0, x_1, x_2).$$

For a valid state of any logic, the summation of valid states (f_i) is always one. Or, clique energy U is always $U = -1$. On the other hand, for any invalid state, the summation of valid states is always zero or $U = 0$. Therefore, the energy of correct logic state is always less than invalid state by a *constant*.

□

For our example, the set of inequalities that must hold is given in Table 1. Here we relate a valid state to all possible invalid states. For example, for valid state $(x_0, x_1, x_2) = 000$ the clique energy in Eqn.5 must evaluate to a lower energy state than all possible invalid state. These relations are given in the 16 inequalities listed in Table 1. These inequalities can be solved using the proposed algorithm similar to Gaussian elimination where a variable that appears with opposite signs in two equations can be eliminated. Applying this procedure to the inequalities in Table 1 the following constraints on the clique coefficients are obtained:

$$\begin{aligned} 2G > D & \quad 2F > C & \quad 2E > A & \quad 2D > B \\ 2G > F & \quad 2F > B & \quad 2E > C & \quad 2D > A \\ 2G > E & & & \end{aligned}$$

The constraints should be viewed as being driven by coefficient G which can take on any positive value. A selected value for $G > 0$ then determines bounds on coefficients, D, E, F in terms of ($2G > D, 2G > F, 2G > E$). They in turn bound A, B, C . The bounds are linear, and so the constraints form a *polytope* in the space of energy coefficients. This concept is illustrated in Figure 7 where a projection onto the D, A, B subspace is depicted. In general, the polytope will be a cone whose cross-section increases linearly with the highest order clique coefficient. The nominal values for the

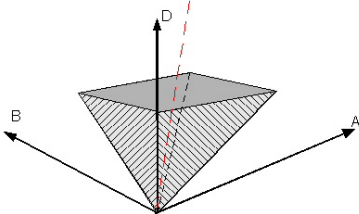


Figure 7: The constraints on the clique coefficients form a cone in the space of coefficient values for $2D > B$ and $2D > A$. The red line indicates the nominal coefficient values.

coefficients of Figure 7 are $A_0 = D_0$ and $B_0 = D_0$.

3.1.2 Carrier Computation in Half-adder

The clique energy for summation in a half-adder design is the same as the exclusive-or case in Eqn.4, while the clique energy for carrier is as following:

$$\begin{aligned} U &= -(1-x_0)(1-x_1)(1-x_3) - (1-x_0)x_1(1-x_3) - \\ &\quad x_0(1-x_1)(1-x_3) - x_0x_1x_3 \\ &= -1 + x_3 + x_0x_1 - 2x_0x_1x_3. \end{aligned}$$

The reason why we want to separately compute the clique energy for summation and carrier is that the summation (x_2) and carrier (x_3) are *independent* outputs. Their results only depend on inputs x_0 and x_1 . Based on such a design, we can drastically reduce the computational complexity of mixing both x_2 and x_3 into clique energy computation as following:

$$\begin{aligned} U &= -(1-x_0)(1-x_1)(1-x_2)(1-x_3) - \\ &\quad (1-x_0)x_1x_2(1-x_3) - x_0(1-x_1)x_2(1-x_3) - \\ &\quad x_0x_1(1-x_2)x_3 \\ &= -1 + x_0 + x_1 + x_2 + x_3 - x_0x_1 - 2x_0x_2 - x_0x_3 - \\ &\quad 2x_1x_2 - x_1x_3 - x_2x_3 + 3x_0x_1x_2 + 2x_0x_2x_3 + \\ &\quad 2x_1x_2x_3 - 2x_0x_1x_2x_3. \end{aligned} \quad (6)$$

If we take the device error into consideration in our design, the clique energy for the summation portion is the same as in Eqn.5 while the clique energy for the carrier portion is as following:

$$U = \kappa + Hx_3 + Dx_0x_1 - 2Ix_0x_1x_3. \quad (7)$$

Here we assume the same error coefficient D for connection x_0x_1 . By combining constraints on the clique coefficients for both summation and carrier cases, we can obtain the following results:

$$\begin{aligned} 2I > D & \quad 2G > D & \quad 2F > C & \quad 2E > A & \quad 2D > B \\ 2I > H & \quad 2G > F & \quad 2F > B & \quad 2E > C & \quad 2D > A \\ & & & & & \quad 2G > E \end{aligned}$$

For the physical realization in terms of CNTs, we expect that the errors in coefficients (i.e. $A, B, C \dots$) will be proportional to their values (i.e. x_0, x_1, \dots). This follows if we assume a fixed probability of connection failure since the coefficient value is made up by a set of redundant gate taps as shown in Fig. 5. For example, a coefficient value of 1.0, might be determined by 5 gate paths, a value of 4.0 by 20 paths, etc.

Given a fixed error rate, α , in the connections leads to a coefficient error proportional to its value, e.g. $D' = D_0 \pm \epsilon$, where $\epsilon = \alpha D_0$. The inequality relating $2D > A$ requires that,

$$\begin{aligned} 2D_0(1 \pm \alpha) &> D_0(1 \pm \alpha) \text{ or} \\ 2 * (1 - \alpha) &> (1 + \alpha), \end{aligned}$$

when the worst case condition is used. Thus, α can be as large as 1/3 without causing a failure of the inequality. The constraint on the D coefficient also permits $\alpha < 1/3$. Similar conditions arise from considering the remaining constraints. Thus for the half-adder circuit, up to one third of the connections can be bad and the correct logic state will still be achieved.

From this example, we observe that complex logic can be decomposed into simple designs by exploiting properties embedded in a circuit. In general, the highest order clique coefficient can be increased until the lowest order coefficient has sufficient connection redundancy to be guaranteed to attain the average error rate. This policy guards against catastrophic failure, where a few bad connections affect a large percentage of the coefficient values. The conical structure of the constraint surface insures that this strategy is always possible.

3.2 Failure Analysis

It should be noted that the failure tolerance depends on the particular clique energy function and follows from the form of the inequalities that arise from correct logic operation. From the simplest inverter logic, its clique energy for general design is:

$$U = \kappa - Ax_0 - Bx_1 + 2Cx_0x_1.$$

By solving the 4 inequality conditions, we can get the constraint conditions as $2C > A, 2C > B$. The bounding *polytope* is similar to that in Fig. 7 and its nominal value set $(C, A, B) = (1, 1, 1)$ provides the optimal solution that can tolerate maximum faults. Let us prove our claim. We can express the 4 inequalities all together in the following matrix format:

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & -1 & 2 \end{bmatrix}}_{\tilde{\mathbf{L}}} \cdot \underbrace{\begin{bmatrix} A \\ B \\ C \end{bmatrix}}_{\tilde{\mathbf{x}}} > \mathbf{0}, \quad (8)$$

or, simply expressed as $\tilde{\mathbf{L}}\tilde{\mathbf{x}} > \mathbf{0}$. To find the optimal solution for fault tolerance is equivalent to find $maximum(\tau_1, \tau_2, \tau_3)$ and sign assignment so that any one or more equations in

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & -1 & 2 \end{bmatrix}}_{\tilde{\mathbf{L}}} \cdot \underbrace{\begin{bmatrix} A(1 \pm \tau_1) \\ B(1 \pm \tau_2) \\ C(1 \pm \tau_3) \end{bmatrix}}_{\tilde{\mathbf{x}'}} \quad (9)$$

will become zero.

The optimal solution of an inverter is its nominal value ($A = B = C = 1$) because

$$\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad (10)$$

which has the equal distances to the boundary condition reflected at the right handside of Eqn.10, which we can also envision in Fig 7. In addition, we can get the worst case $\tau = 1$. For instance, we can replace $A = 1$ by $A' = A(1 + \tau) = 2$, and one of the inequalities in Eqn.9 equals to zero. In our nano-scale computation, we should make our design near the nominal value that provides maximum fault tolerance.

In general, the constraint condition can be expressed as $\tilde{\mathbf{L}}\tilde{\mathbf{x}} > \mathbf{0}$, where $\tilde{\mathbf{L}}$ is an $m \times n$ matrix depending on the valid and invalid states in a logic (in an inverter design, $\tilde{\mathbf{L}}$ is a 4×3 matrix). $\tilde{\mathbf{x}}$ is an $n \times 1$ column vector determined by the number of terms in the clique energy expression (in the inverter design, $\tilde{\mathbf{x}}$ is a 3×1 column vector).

In our design, the clique energy of valid states is always less than that of invalid states. For its nominal vector (here it is a column vector with all elements that equal to k), the difference between the different energies is constant (refer to Lemma 1). Or, we can express the relation as following:

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix}}_{\tilde{\mathbf{L}}} \cdot \underbrace{\begin{pmatrix} k \\ k \\ \vdots \\ k \end{pmatrix}}_{\text{nominal vector}} = \begin{pmatrix} k \\ k \\ \vdots \\ k \end{pmatrix}, \quad (11)$$

Here k is a constant number. Let us keep all coefficients except for the x_i in column vector $\tilde{\mathbf{x}}$ unchanged (i can be any value). Now,

the t^{th} row in the inequalities $\tilde{\mathbf{L}}\tilde{\mathbf{x}} > \mathbf{0}$ can be rewritten as:

$$a_{t1}k + \cdots + a_{t(i-1)}k + a_{ti}x_i + a_{t(i+1)}k + \cdots + a_{tn}k > 0 \quad (12)$$

From Eqn.11, we know,

$$a_{t1}k + \cdots + a_{t(i-1)}k + a_{ti}k + a_{t(i+1)}k + \cdots + a_{tn}k = k \quad (13)$$

By combining Eqn.12 and Eqn.13, we can get

$$a_{ti}x_i > k(a_{ti} - 1), \quad (14)$$

here a_{ti} can be either a positive or negative number depending on the clique energy expression of a logic.

- If $a_{ti} \geq 0$, any positive x_i ($x_i > 0$) satisfies the inequality equation (14).
- If $a_{ti} < 0$, we get,

$$x_i < k\left(1 - \frac{1}{a_{ti}}\right) \leq 2k$$

Therefore, $a_{ti}x_i > k(a_{ti} - 1) \quad \forall x_i \in (0, 2k)$

If we keep $(n - 1)$ elements in $\tilde{\mathbf{x}}$ unchanged, the remain coefficient in $\tilde{\mathbf{x}}$ will be a value in the range $(0, 2k)$ to satisfy the inequalities. Or,

$$\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{k}}\| \leq k, \quad \forall \tilde{\mathbf{x}}_i \in \{\tilde{\mathbf{x}} | \tilde{\mathbf{L}}\tilde{\mathbf{x}} > \mathbf{0}\}.$$

3.3 Errors in Signal

There are two aspects of fault tolerance that must be considered: 1) tolerance due to structural failure and 2) tolerance due to signal noise. In this section, we will discuss the second type of error that is directly accounted for in the probability maximization process inherent in MRF processing. In addition, we will give examples of the *belief propagation* process.

3.3.1 Discrete Errors in Signal

The use of a probabilistic approach to logic has the advantage that the process is inherently fault tolerant to errors in the value of logic state variables. The behavior of a simple inverter circuit will be used to illustrate this aspect of the Markov random network approach.

In order to consider the error behavior of more complex circuits, it is necessary to describe the processing of logic signals through the Markov random network. This process is carried out by the chaining of conditional probabilities by *Belief Propagation* [11]. As explained in Section 2.4, the probability of logic variables can be determined by summing (marginalizing) over the set of possible values of clique neighborhood states except for the variable in question. What remains is the probability for the single variable. This probability can be propagated to the next node in the network and used for the next summation. An example of this basic algorithm will be shown in Eqn.17.

The Gibbs distribution for an inverter is given by:

$$p(x_0, x_1) = \frac{1}{Z} e^{-\frac{1}{T}(2x_0x_1 - x_0 - x_1)}. \quad (16)$$

The partition function Z normalizes the expression as required for a probability. Suppose the input, x_0 , takes on values from $\{0, 1\}$. The dependence on the input x_0 can be marginalized away by summing over its possible values, i.e.,

$$\begin{aligned} p(x_1) &= \frac{1}{Z} \sum_{x_0=\{0,1\}} e^{-\frac{1}{T}(2x_0x_1 - x_0 - x_1)} \\ &= \frac{e^{\frac{x_1}{T}} + e^{\frac{(1-x_1)}{T}}}{2(1 + e^{\frac{1}{T}})}. \end{aligned} \quad (17)$$

This function is plotted in Figure 8 (a) for various values of T . Both 0 and 1 are equally likely, but note that the most likely outputs are 0 or 1 with the likelihood of any intermediate values becoming vanishingly small as $T \rightarrow 0$. This behavior is characteristic of Markov random network processing. As long as the energy balance is favorable to the correct logic state, decreasing temperature will lock in the valid configurations.

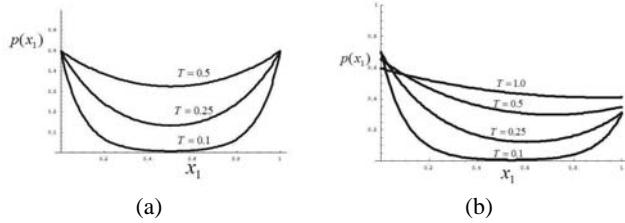


Figure 8: (a) The probability of an inverter output as a function of T . (b) the input is one, with probability 0.7.

In actual operation of a logic circuit, the input states would not be equally likely but would have higher probability of being in a given state, as required by deterministic behavior. For example, suppose the input to the inverter has $p(1) = 0.7, p(0) = 0.3$ then the Gibbs distribution of Figure 8(a) is as shown in Figure 8(b). As the temperature increases, this probability margin asymptotically approaches zero. A nano logic device will by necessity operate with logic energies within a few times kT in order to achieve the expected reduction in power afforded by the small scale of nano-devices. For finite temperatures, the policy of choosing the output state with the highest probability always yields the correct logic operation. However, it can be expected that errors will result if $|p(x) - 0.5|$ is small, since any physical realization of the Markov network will have significant fluctuation of the logic levels.

3.3.2 Continuous Errors in Signal

We have discussed the impact of discrete errors previously. Now, let us extend our discussion to simulate the impact caused by continuous noise. We use a *Gaussian* distribution to model the noise introduced by the movement of electrons, such as Brownian motion, in a nano-scale electric circuit.

The joint probability of our inverter design is expressed in Eqn.16. In our design, we assume the *Gaussian* process is centered around μ and it can be expressed as:

$$p_{gaussian} = \frac{1}{\sqrt{2\pi\sigma}} e^{-(x_0 - \mu)^2 / 2\sigma^2}.$$

For the '0' and '1' inputs, the Gaussian noise distribution is shown in Fig. 9(a). However, the noise distribution below zero will be truncated or filtered out because negative values are invalid in the design. To make the *Gaussian* noise symmetrically distributed about the inputs of an inverter, we have to make the coordinate shift ($x = 0 \rightarrow x' = -1$, and $x = 1 \rightarrow x' = 1$) as shown in Fig. 9(b).

By applying the shift

$$x'_0 = 2(x_0 - \frac{1}{2}), x'_1 = 2(x_1 - \frac{1}{2}),$$

we can rewrite the joint probability in Eqn.16 as:

$$p(x'_0, x'_1) = \frac{1}{Z} e^{-\frac{1}{2T}(x'_0 x'_1 + 1)}. \quad (18)$$

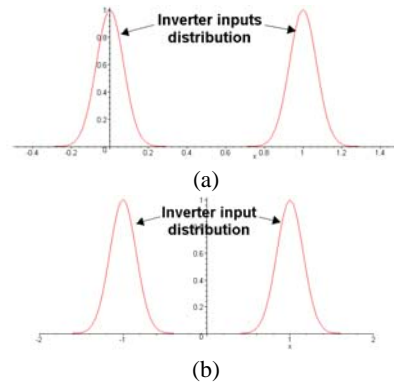


Figure 9: Two choices of noise distribution among inputs of inverter design (a) inputs at 0 and 1 (b) inputs at -1 and 1

We can compute the conditional probability as:

$$\begin{aligned} p(x'_1 | x'_0) &= \frac{p(x'_1, x'_0)}{\int_{-1}^1 p(x'_1, x'_0) dx'_1} = \frac{e^{-\frac{1}{2T}(x'_0 x'_1 + 1)}}{\int_{-1}^1 e^{-\frac{1}{2T}(x'_0 x'_1 + 1)} dx'_1} \\ &= \frac{x'_0}{2T} \frac{e^{-\frac{1}{2T}(x'_0 x'_1 + 1)}}{e^{-\frac{1-x'_0}{2T}} - e^{-\frac{1+x'_0}{2T}}} = \frac{x'_0}{2T} \frac{e^{-\frac{1}{2T}x'_0 x'_1}}{e^{\frac{x'_0}{2T}} - e^{-\frac{x'_0}{2T}}} \end{aligned}$$

By applying *Taylor* approximation $e^x = 1 + x + \dots$, we get

$$p(x'_1 | x'_0) = \frac{e^{-\frac{1}{2T}x'_0 x'_1}}{2}$$

The conditional probability is shown in Fig. 10. In this 3D plot, we can observe that right states ($(x_0, x_1) = (0, 1)$ and $(1, 0)$) have higher probability (close to one) while the wrong states ($(x_0, x_1) = (0, 0)$ and $(1, 1)$) have lower (close to zero) probability. Here, x_0 and x_1 stand for the input and output, respectively. Its marginalized probability is

$$p(x'_1) = K \int_{-1}^1 p(x'_1 | x'_0) dx'_0 = KT \frac{e^{x_1/2T} - e^{-x_1/2T}}{x_1},$$

and it is shown in Fig. 11 with different temperature T . From the figure, we observe that the INVERTER is *symmetrical* in its probability distribution and the logic error arises when T gets higher. Next, we consider how the noise around input one impacts IN-

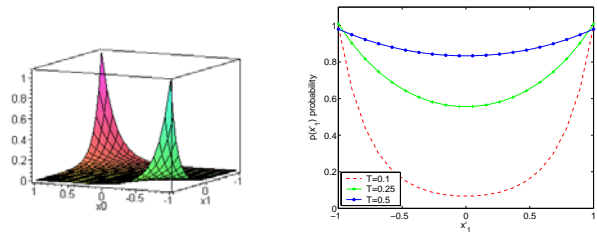


Figure 10: Conditional probability $p(x'_1 | x'_0)$ of proposed inverter design.

Figure 11: Marginalized probability $p(x'_1)$ varies with T .

VERTER logic and the same arguments can be applied to those around zero (or, -1 after the coordinate shift).

By modeling input $p(x'_0)$ as *Gaussian* process with mean of one

($\mu = 1$), we can compute the output probability $p(x'_1)$ as following:

$$\begin{aligned} p(x'_1) &= K \int_{-1-2\sigma}^{1+2\sigma} p(x'_1|x'_0) e^{-\frac{(x'_0-1)^2}{2\sigma^2}} dx'_0 \\ &= K \int_{-1-2\sigma}^{1+2\sigma} \frac{e^{-\frac{1}{2T}(x'_0 x'_1)}}{2} e^{-\frac{x'_0-1}{2\sigma^2}} dx'_0 \end{aligned}$$

where K is a constant. The marginalized probability with Gaussian noise, $p(x'_1)$, is shown in Figure 12 (a) and (b). From those fig-

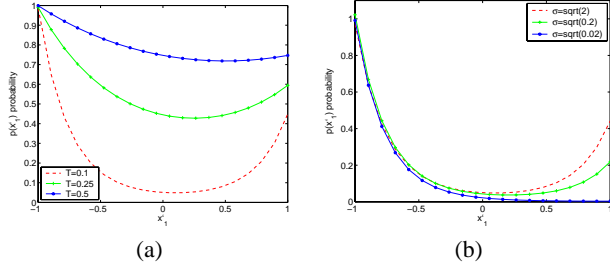


Figure 12: How a Gaussian process ($\mu = 1$) impacts an inverter output (a) $p(x'_1)$ varies with T , (b) $p(x'_1)$ varies with σ .

ures, we observe that the Gaussian process around 1 makes $p(x'_1)$ *asymmetrical*. Or, the probability around -1 is higher than that around 1, and the inverter error rate increases with σ as expected.

We also quantify our observations using the 'error rate' criteria to measure fault tolerance in terms of T and σ . The 'error rate' is defined as:

$$\text{error rate} = \frac{\text{incorrect probability}}{\text{correct} + \text{incorrect probability}}$$

The results are shown in Figure 13 (a) and (b). For instance, when $T = 0.1$, the error rate is around 0.1%. From both Figure 12

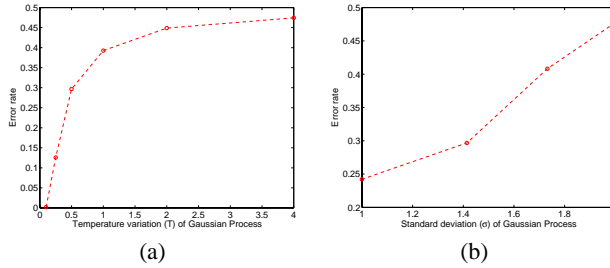


Figure 13: The error rate of an inverter output (a) with different T while $\sigma = \sqrt{2}$, (b) with different σ while $T = 0.5$.

and Figure 13, we observe that our proposed design favors low T and small σ . For instance, when $T > 4$, the inverter starts to malfunction because error rate approaches 50% or it reaches ambiguity level – we can not distinguish correct outputs from wrong outputs.

Most applications of the MRF model treat T as a variable that can be manipulated in solving for the maximum probability state. In our realization of the model, T actually corresponds to physical temperature in terms of device signal energy. We expect that this relationship will provide an important bridge between physics and computation.

3.3.3 Logic Processing

In the previous discussion, we take a simple inverter as an example. In this section, we consider a two-input NAND design. The

Gibbs joint and conditional probability distributions for a NAND gate are given by,

$$p(x_a, x_b, x_c) = \frac{1}{Z} e^{-\frac{1}{T}(2x_a x_b x_c - x_a x_b - x_c)}$$

where x_a, x_b are the inputs and x_c is the output. The NAND gate is *asymmetrical* in its probability distribution as shown in Figure 14. This result was obtained by marginalizing over all input combinations.

$$p(x_c) = \int_{-1}^1 \int_{-1}^1 p(x_c|x_a, x_b) dx_a dx_b$$

The result shows that for a uniform distribution of inputs, the prob-

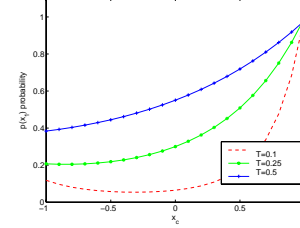


Figure 14: The marginalized probability $p(x_c)$ with different temperature T .

ability of a '1' output state is three times that of a '0' state. Efforts are underway to carry out investigation of more complex logic and analyze the fault tolerance of temperature T and Gaussian standard deviation σ .

4. CONCLUSION AND FUTURE WORK

In this paper, we propose a unique probabilistic-based design methodology for nano-scale computer architecture. The main reason for selecting the Markov random network and Gibbs energy distribution as the basis for our nano-architectural approach is that its operation does not depend on perfect devices or perfect connections. Most importantly, our proposed design is dynamically *defect tolerant*. We have provided examples of how the MRF mapping can be used to model device and connection failure. Extended work will also include examples of how marginal signal probabilities are propagated through a Markov random network to model tolerance to signal noise. This process is carried out by the chaining of conditional probabilities by *belief propagation* [11]. Ultimately the network converges to a stable set of state probabilities reflecting the required result. Successful operation only requires that the energy of correct states is lower than the energy of errors.

5. REFERENCES

- [1] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36(3):192–236, 1994.
- [2] R. Chellappa. *Markov random fields: theory and applications*. Academic Press, 1993.
- [3] S. Folling, O. Turel, and K. Likhav. Single-electron latching switches as nanoscale synapses. In *Proc. of Int. Joint Conf. on Neural Networks*, pages 216–221, July 2001.
- [4] S. C. Goldstein and M. Budiu. Nanofabrics: Spatial computing using molecular electronics. In *The 28th Annual International Symposium on Computer Architecture*, pages 178–189, 2001.
- [5] S. C. D. Kirkpatrick, J. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
- [6] J. Li, C. Papadopoulos, J. M. Xu, and M. Moskovits. Highly-ordered carbon nanotube arrays for electronics applications. *Applied Physics Letters*, 75(3):367–, July 1999.
- [7] S. Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer, 1995.
- [8] R. Martel, V. Derycke, J. Appenzeller, S. Wind, and P. Avouris. Carbon nanotube field-effect transistors and logic circuits. In *Proceedings of the 39th Conference on Design Automation*, pages 94–98, 2002.
- [9] R. R. Schulz and R. L. Stevenson. A Bayesian approach to image expansion for improved definition. *IEEE Transactions on Image Processing*, 3(3):233–242, 1994.
- [10] J. von Neumann. *Probabilistic Logic and the Synthesis of Reliable Organisms from Unreliable Components*. Automata Studies. Princeton University Press, 1956.
- [11] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *International Joint Conference on AI*, 2001. Distinguished Lecture.