FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 24, no. 3, December 2011, 451-482

The EXOR Gate Under Uncertainty: A Case Study

**Svetlana N. Yanushkevich, An Hong Tran, Golam Tangim,
Vladimir P. Shmerko, Elena N. Zaitseva and Vitaly Levashenko**

Abstract: Probabilistic AND/EXOR networks have been defined, in the past, as a class of Reed-Muller circuits, which operate on random signals. In contemporary logic network design, it is classified as *behavioral* notation of probabilistic logic gates and networks. In this paper, we introduce additional notations of probabilistic AND/EXOR networks: *belief propagation*, *stochastic*, *decision diagram*, *neuromorphic* models, and *Markov random field* model. Probabilistic logic networks, and, in particular, probabilistic AND/EXOR networks, known as turbo-decoders (used in cell phones and iPhone) are in demand in the coding theory. Another example is intelligent decision support in banking and security applications. We argue that there are two types of probabilistic networks: traditional logic networks assuming random signals, and *belief propagation* networks. We propose the taxonomy for this design, and provide the results of experimental study. In addition, we show that in forthcoming technologies, in particular, molecular electronics, probabilistic computing is the platform for developing the devices and systems for low-power low-precise data processing.

Keywords: AND-EXOR networks; probabilistic computation; belief propagation network; decision diagram; neuromorphic network; Markov random field.

1 Introduction

The elementary Boolean function EXOR (Fig. 1) is notable in logic design because it is a basic operation of Reed-Muller techniques (an arbitrary Boolean function can

Manuscript received July 3, 2011. An earlier version of this paper was presented at the Reed Muller 2011 Workshop, May 25-26, 2011, Gustavelund Conference Centre, Tuusula, Finland.

S. N. Yanushkevich, A. H. Tran, G. Tangim, and V. P. Shmerko are with the Department of Electrical and Computer Engineering, University of Calgary, Canada, (e-mail: syanshk@ucalgary.ca). E. N. Zaitseva and V. Levashenko are with the Department of Informatics, University of Zilina, Slovakia, (e-mail: Elena.Zaitseva@fri.uniza.sk).

Digital Object Identifier: 10.2298/FUEE1103451Y

be implemented using EXOR and AND operations, and constant "1") [29]. It is also the most often operation used in testing, coding techniques, and encryption [25].

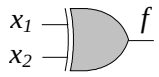
Various fault-tolerant techniques for Reed-Muller have been proposed recently, in particular, [21]. However they cannot solve the problems, related to random intrinsic noise, which is observed in ultra deep submicron and predictable nanoscale devices. Moreover, there are various physical and chemical phenomena in nanospace, which can be directly encoded as EXOR function [22]. A three-dimensional implementation of EXOR functions has been proposed in [23].

Two-input EXOR gate (Nonequivalence)

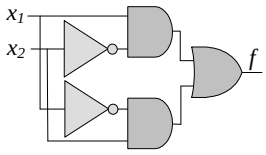
Truth table

x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	0

Denotation



Implementation

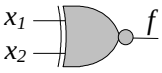


Two-input XNOR gate (Equivalence)

Truth table

x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	1

Denotation



Implementation

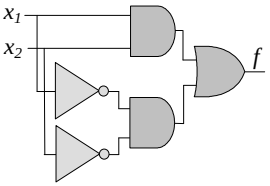


Fig. 1. Specification of two input EXOR gate $f = x_1 \oplus x_2$, and XNOR gate $f = \overline{x_1 \oplus x_2}$, given deterministic inputs: the truth table, graphical denotation, and AND/OR implementation.

The deterministic model of EXOR gate (Fig. 1) operates on noise-free signals. Focus of this paper is the behavior of EXOR function in probabilistic environment, which is defined by replacing deterministic signals whith random signals. Specifically, when noise is allowed, input signals are applied to EXOR gate with some level of probability. Probabilistic models assume, that correct output signals are calculated with some level of probability. Behavior of EXOR gate in probabilistic environment is completely described by probability distribution functions. Input probabilities are derived from these functions in the form of probabilistic truth table (Fig. 2). For simplification, it is also often assumed that the input signals are uncorrelated and independent. This problem in more general meaning is known as *probabilistic computation.*

There are various fields, where probabilistic computing is the key technique. For example, in communication, the best error correcting technique, known as turbo coding, is based on probabilistic data processing [1]. The devices and systems,

Probabilistic truth table

$p(x_1)$	$p(x_2)$	$p(f)$
$p(0)$	$p(0)$	$p(0)$
$p(0)$	$p(1)$	$p(1)$
$p(1)$	$p(0)$	$p(1)$
$p(1)$	$p(1)$	$p(0)$

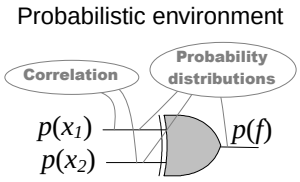


Fig. 2. Specification of two input EXOR gate in probabilistic environment: probabilistic truth table, probability distributions (inputs and output signals), and correlation between input signals.

known as *belief networks*, are widely used in decision-making, medical diagnosis, image and voice processing, robotics, and control systems [15]. In forthcoming technologies, in particular, molecular electronics, probabilistic computing represent a platform for developing the devices and systems for stochastic data processing [11]. It should be noted that:

1. Multiple meaning of the term "probabilistic computing" have been applied in various area-specific approaches [6, 14, 16, 17, 28].
2. Different methodologies are used in designing the devices and systems for probabilistic computing [1, 4, 10, 12].

This paper contributes in probabilistic computing paradigm as follows: (a) we systematize various libraries of probabilistic gates, which can be used in design and modeling of both logic and belief networks, and (b) introduce experimental results for elementary probabilistic gates. However, there are some novelties in our paper, such as rules for combining neuromorphic models of logic gates and extension of Markov model for multivalued logic gates.

Since our study relates to various fields and design methodologies, we introduce basic terminology and definitions.

Terminology

Probabilistic logic is a classical approach to reasoning under uncertainty.
Probabilistic gate is defined as logic gate, which operates on (a) random binary signals or (b) probabilities of these signals [10, 12, 20].
Belief propagation network is defined as a directed acyclic graph derived from causal graphs [15].
Stochastic computing is a method to design low-precision digital devices; this method is based on encoding signals by Bernoulli sequences and their statistical properties [4, 8, 14].

In this paper, we introduce the following models of probabilistic logic gates, while focusing on AND and EXOR gates:

Model 1: **Behavioral** model of a logic gate, assuming random input signals.
Model 2: **Belief propagation** model, in which input and outputs signals are probabilities (real numbers) of the corresponding binary signals; it is represented by belief propagation networks.
Model 3: **Stochastic** model, in which probabilities are represented by binary stochastic pulse streams; the information in a pulse stream is contained in the forms of primary statistics.
Model 4: **Decision diagram** model using noise injection techniques.
Model 5: **Neuromorphic** model, which is a typical Hopfield associative memory, utilizing the Boltzmann updating algorithm, and
Model 6: **Markov random field** model, which is similar to a Bayesian network, while it can describe cyclic dependencies, which Bayesian network cannot.

The common feature of these models is that they utilize probabilistic measures of uncertainty. However, they are different with respect to input and output data, algorithms, implementation, and applications, in particular:
(a) Stochastic model is useful for operations in presence of noise, such as addition, subtraction, multiplication, and division; that is, this model can be used in other probabilistic models as operational unit.
(b) Bayesian models require causal representations of data; in contrast, Markov random field model is based on noncausal data description;
(c) Hopfield model (implemented as neural network of a particular configuration) can be converted to Markov random field (implemented as operational unit), and vice versa.

2 Behavioral model

Behavioral model is based on probabilistic logic, which is a classical approach to reasoning under uncertainty, as devised by George Boole. Behavioral AND/EXOR model under assumption of random signals is defined as follows:

$$\underbrace{\texttt{AND/EXOR expression}}_{Phenomenon} \overset{Design}{\longrightarrow} \underbrace{\texttt{Logic network}}_{Behavioral\ form}$$

In this design paradigm, traditional taxonomy of AND/EXOR network design is extended by probabilistic descriptions of logic gates in behavioral form.

Theorem 1. *Given a* 2*-input logic gate* $f(x_1, x_2)$*, its behavioral model is derived by (a) transforming logic function* $f(x_1, x_2)$ *into an arithmetic form, and (b) replacing signals* x_1 *and* x_2 *with their probabilities* p_1 *and* p_2*, respectively, assuming that* x_1 *and* x_2 *are independent.*

Note that in behavioral model, p_1 and p_2 are the probabilities of signals x_1 and x_2 being logic "1", respectively. Proof of this theorem follows from the properties of probabilities for independent random events and the properties of arithmetic forms of Boolean functions.

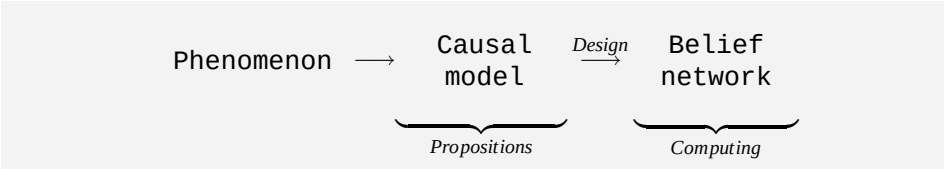
Example 1. *Let the inputs x_1 and x_2 of the 2-input EXOR gate be mutually independent with probabilities of being "1" p_1 and p_2, respectively. The behavioral model represents the probability of the output being logic "1" and is derived from the truth table of EXOR function:*

$$p = \underbrace{(1-p_1)p_2}_{\text{For } x_1=0, x_2=1} + \underbrace{p_1(1-p_2)}_{\text{For } x_1=1, x_2=0} = p_1 + p_2 - 2p_1p_2.$$

or by transforming the EXOR function into the arithmetic form, $x_1 \oplus x_2 = x_1 + x_2 + 2x_1x_2$ and replacing x_i with $p_i,\ i = 1,2$, that is, $p_1 + p_2 - 2p_1p_2$. Supposing $p_1 = 0.8,\ p_2 = 0.9$, the logic "1" at the output is produced with probability $p = 0.8 + 0.9 - 2 \times 0.8 \cdot 0.9 = 0.26$.

3 Belief propagation model

In the belief propagation model, any phenomenon must first be described in causal form, and then, using probabilistic relationships, transformed into a belief propagation network:



Causal modeling attempts to resolve question about possible causes so as to provide explanation of phenomena (effects) as the result of previous phenomena (causes). Causal knowledge is modeled using the causal networks, in which the nodes represent propositions (or variables), the arcs signify direct dependencies between the linked propositions, and the strengths of these dependencies are quantified by conditional probabilities. A Bayesian network is a type of belief network that captures the way the propositions relate to each other probabilistically.

The simplest form of the belief propagation model is as follows. If k events $B_1, B_2, \ldots, B_k$ constitute a partition of the sample space S, such that $P(B_i) \neq 0$ for $i = 1, 2, \ldots, k$, then, for any events B_r and A of S such that $P(A) \neq 0$,

$$\underbrace{P(B_r|A)}_{\texttt{Posterior}} = \underbrace{P(A|B_r)}_{\texttt{Likelihood}} \times \frac{\overbrace{P(B_r)}^{\texttt{Prior}}}{\underbrace{P(A)}_{\texttt{Evidence}}}$$

where $r = 1,2,\ldots k$; $P(B_r|A)$ is a *revised* or a *posterior* probability; $P(A|B_r)$ is the *likelihood* of B_r with respect to A; $P(A)$ is the *evidence factor* and can be viewed as merely a scale factor, that guarantees that the posterior probabilities sum to one, as all good probabilities must.

This belief propagation form, or Bayesian principle, advises on how to update probabilities, once such a conditional probability structure has been adapted, given appropriate prior probabilities.

Let the nodes of a graph represent random variables $X = \{x_1,\ldots,x_m\}$, and the links between the nodes represent direct causal dependencies. A *Bayesian belief networks* is based on a *factored* representation of joint probability distribution.

3.1 Probabilistic logic gates for belief propagation model

Belief propagation model is implemented using probabilistic logic gates, which operate on probabilities (real numbers). The general design taxonomy of these gates is as follows:

Probabilistic logic gate for belief propagation model

A two-input probabilistic logic gate with random inputs, $x_1 \in \{0,1\}$, $x_2 \in \{0,1\}$, and random output, $y \in \{0,1\}$, is defined as a computational unit, that performs computations as follows:

$$p(y) = \alpha \sum_{x_1} \sum_{x_2} p(x_1)p(x_2)f(x_1,x_2,y) \tag{1}$$

where $p(x_1)$, $p(x_2)$, and $p(y)$ are the probability distributions of binary inputs x_1, x_2, and output $y \in \{0,1\}$, respectively;
$f(x_1,x_2,y) \in \{0,1\}$ is the binary function called *compatibility truth table*, that indicates the truth of logic function y, i.e. $f(x_1,x_2,y) = 1$ if y is true, and $f(x_1,x_2,y) = 0$ otherwise; and
$\alpha \in \{0,1\}$ is an appropriate scale factor.

Equation 1 shows how the probability distribution of the output random variable Y is derived from probability distributions of input random variables X_1 and X_2. Using the parametrization property of the function $f(x_1,x_2,y) \in \{0,1\}$, a library of probabilistic logic gates is defined.

Example 2. *Using Equation (1), probabilistic* EXOR *gate is defined as follows:*

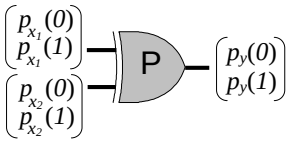
$$\begin{cases} p(0) &= \sum_{x_1}\sum_{x_2} p(x_1)p(x_2)f(x_1,x_2,y=0) \\ p(1) &= \sum_{x_1}\sum_{x_2} p(x_1)p(x_2)f(x_1,x_2,y=1) \end{cases}$$

To calculate $p(0)$ and $p(1)$, we sum over all possible (binary) values of x and y. The constraint within $f(x_1,x_2,y)$ serves to include some probability terms and exclude others. Function $f(x_1,x_2,y)$ is calculated using the truth table of EXOR *gate as follows:*

Compatibility truth table

x_1	x_2	y		f
0	0	0	TRUE	1
0	0	1		0
0	1	0		0
0	1	1	TRUE	1
1	0	0		0
1	0	1	TRUE	1
1	1	0	TRUE	1
1	1	1		0

Graphical denotation of probabilistic EXOR *gate*



The probabilistic EXOR *gate is given*

$$\begin{bmatrix} p_y(0) \\ p_y(1) \end{bmatrix} = \begin{bmatrix} p_{x_1}(0)p_{x_2}(0) + p_{x_1}(1)p_{x_2}(1) \\ p_{x_1}(0)p_{x_2}(1) + p_{x_1}(1)p_{x_2}(0) \end{bmatrix} \tag{2}$$

Example 3. *The probabilistic model of* AND *gate is a vector form*

$$\begin{bmatrix} p_y(0) \\ p_y(1) \end{bmatrix} = \begin{bmatrix} p_{x_1}(0)p_{x_2}(0) + p_{x_1}(0)p_{x_2}(1) + p_{x_1}(1)p_{x_2}(0) \\ p_{x_1}(1)p_{x_2}(1) \end{bmatrix} \tag{3}$$

Implementation of Reed-Muller polynomial $y = x_1x_2 \oplus x_3 \oplus x_4$ using both behavioral and reasoning models is given in Fig. 3.

3.2 Logarithm operational domain for probabilistic logic gates

Due to the high hardware complexity of sum-of-product form of probabilistic gates, operations can be implemented in logarithm domain where probabilities are considered as *log-likelihood ratio* [1, 12].

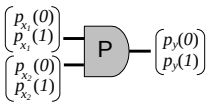
Let X be a binary random variable, and 0- and 1-value of this variable can be observed with probability $p(x=0)$ and $p(x=1)$, respectively. The log-likelihood ratio of X is as follows:

$$L[x] = \ln \frac{p(x=1)}{p(x=0)} \tag{4}$$

Table 1. The sum-of-product and log-likelihood forms of AND and EXOR probabilistic logic gate models.

Sum-of-product model

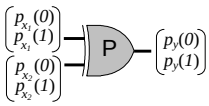
Probabilistic AND



x_1	x_2	y	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Probabilistic model: see equation (2)

Probabilistic EXOR

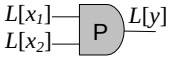


x_1	x_2	y	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Probabilistic model: see equation (3

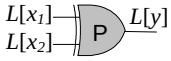
Log-likelihood model

Probabilistic AND



$$L[y] = L[x_1 x_2] = \mathbf{MAX}(|L[x_1]|, |L[x_2]|)$$

Probabilistic EXOR



$$L[y] = L[x_1 \oplus x_2] = \mathbf{SGN}(L[x_1])\mathbf{SGN}(L[x_2]) \times \mathbf{MIN}(|L[x_1]|, |L[x_2]|)$$

The probability $p(x=0)$ can be recovered from Equation 4:

$$p(x=0) = \frac{e^{L[x]}}{1+e^{L[x]}}$$

Operations in the logarithm domain are specified by properties of log-likelihood ratio model of probabilistic logic gate (Equation 4), in particular, multiplication of real numbers is translating into additions.

The AND and EXOR probabilistic logic gates based on log-likelihood ratio model (Equation 4) are used to design belief propagation networks.

Example 4. *The design goal of decoder of turbo error correcting codes is to propagate the belief efficiently. Specifically, the sign and magnitude of $|L[x]|$ (belief) are estimated, and then improving these estimates using the local redundancy of the*

Design example: Reed-Muller network design using behavioral model

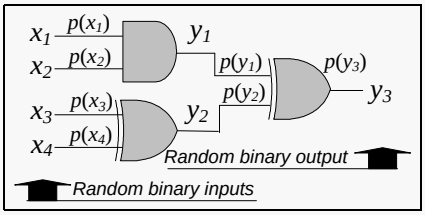
Given:

(a) a switching function $y = x_1x_2 \oplus x_3 \oplus x_4$,
(b) the binary input signals x_1, x_2, x_3, and x_4 are independent,
(c) the probabilities of 1's in the input streams,
$p(x_1) = p(x_2) = p(x_3) = p(x_4) = 0.8$

Design a probabilistic logic network using behavioral model, and calculate $p(y)$.

Solution:

Step 1: Design Type I probabilistic logic network:



Step 2: Using behavioral models

$$p(y_1) = p(x_1)p(x_2) = 0.8 \times 0.8 = \boxed{0.64}$$

$$p(y_2) = p(x_3) + p(x_4) - 2p(x_3)p(x_4)$$
$$= 0.8 + 0.8 - 2 \times 0.8 \times 0.8 = \boxed{0.32}$$

$$p(y_3) = p(y_1) + p(y_2) - 2p(y_1)p(y_2)$$
$$= 0.64 + 0.32 - 2 \times 0.64 \times 0.32$$
$$= \boxed{0.5504}$$

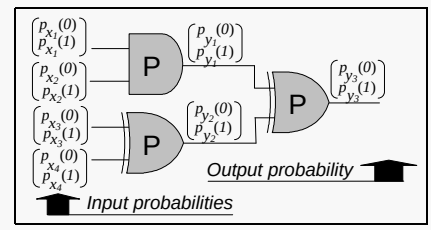
Design example: Reed-Muller network design using reasoning model

Given:

(a) a switching function $y = x_1x_2 \oplus x_3 \oplus x_4$,
(b) the input binary signals x_1, x_2, x_3, and x_4 are independent,
(c) the probabilities of the input signals, $p_{x_1}(0) = \cdots = p_{x_4}(0) = 0.2$ and $p_{x_1}(1) = \cdots = p_{x_4}(1) = 0.8$.

Design a probabilistic logic network to implement function y using reasoning model, and calculate $p(y)$.

Solution: Step 1: Design Type II probabilistic logic network:



Step 2: Using Equations 2 and 3

$$\begin{bmatrix} p_{y_1}(0) \\ p_{y_1}(1) \end{bmatrix} = \begin{bmatrix} 0.2 \cdot 0.2 + 0.2 \cdot 0.8 + 0.8 \cdot 0.2 = 0.36 \\ 0.8 \times 0.8 = \boxed{0.64} \end{bmatrix}$$

$$\begin{bmatrix} p_{y_2}(0) \\ p_{y_2}(1) \end{bmatrix} = \begin{bmatrix} 0.2 \times 0.2 + 0.8 \times 0.8 = 0.68 \\ 0.2 \times 0.8 + 0.8 \times 0.2 = \boxed{0.32} \end{bmatrix}$$

$$\begin{bmatrix} p_{y_3}(0) \\ p_{y_3}(1) \end{bmatrix} = \begin{bmatrix} 0.36 \cdot 0.68 + 0.64 \cdot 0.32 = 0.4496 \\ 0.36 \cdot 0.32 + 0.64 \cdot 0.68 = \boxed{0.5504} \end{bmatrix}$$

Fig. 3. Reed-Muller probabilistic network design using behavioral (left) and reasoning (right) models.

code. The result of this belief propagation can be written as iterative process:

$$L_{t+1}[x] = L_t[x] + p(x), \quad t = 1, 2, \ldots, m,$$

where $p(x)$ is the probabilistic quantity, or extrinsic *information about x.*

In Table 1, the log-likelihood ratio model is given for two-input AND and EXOR gates assuming that binary variables x_1 and x_2 are statistically independent.

Figure 4 provides a graphical representation of the log-likelihood ratio model of probabilistic EXOR logic gate and approximation of this model. We can observe that the EXOR model has a lot of deviation from the original model and its approximation.

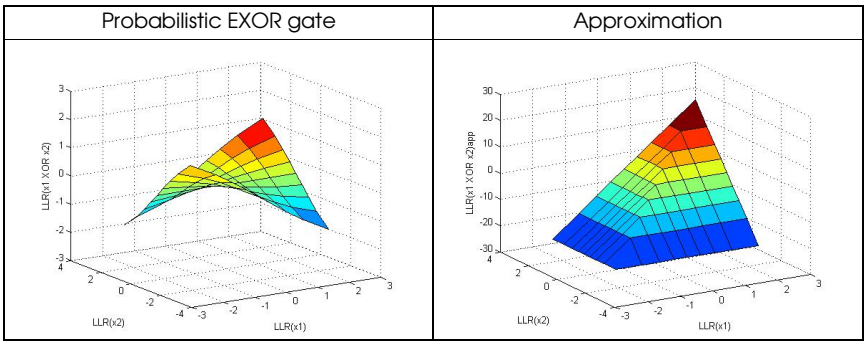


Fig. 4. Graphical representation of log-likelihood ratio model of probabilistic logic gates.

3.3 Belief trees and networks

Belief trees are predecessors of the belief networks, and can be used to construct belief networks, when complete data is not available. In the below example (Fig. 5), we consider a belief network that investigates how to include possible errors of distance measurements of temperature in determining the probability of flu in the presence of temperature [30].

Let high temperature T in a pre-screened individual be detected. The prior statistics include the following parameters:
(a) The prior probability of a flu F is $P(F) = 0.05$;
(b)] The conditional probability of a flu not causing high temperature is $P_{T|\overline{F}} = 0.2$;
(c) The probability of a flu causing high temperature is $P_{T|F} = 0.9$.
The temperature T is evaluated by means of an infrared image with the following errors (a) 5% FRR, and 15% FAR.

Computing using the Bayesian belief network is based on the following computational aspects:
Local computing is the key principle of belief network.
Updating beliefs is the main principle in scheduling of computational tasks.
Decision profile is a specification of a computing task.
Data transmission consists of transmission of the probability values (from local memories or computed) and additional messages for activation of nodes accordingly to a decision profile.

Reasoning with Bayesian networks is done by updating beliefs, that is, computing the posterior probability distributions, given new information, called *evidence*. The basic idea is that new evidence has to be propagated to the other parts of the network.

Fundamental expansion for belief networks

A *Bayesian belief networks* is a graphical representation of a chain rule; that is, a *factored* representation of joint probability distribution in the form

$$P(x_1, x_2, \ldots, x_n) =$$
$$= \overbrace{P(x_1) \times P(x_2|x_1) \times \ldots, P(x_n|x_1, \ldots, x_{n-1})}^{\texttt{Factored form}}$$
$$= \underbrace{\prod_i P(x_i|x_1, x_2, \ldots, x_{i-1})}_{\texttt{Chain rule}} \Leftrightarrow \underbrace{\prod_{i=1}^{m} P(x_i|\texttt{Par}(X_i)}_{\texttt{Graphical representation}}$$

where $\texttt{Par}(X_i)$ denotes a set of parent nodes of the random variable x_i. The nodes outside $\texttt{Par}(X_i)$ are conditionally independent of x_i.

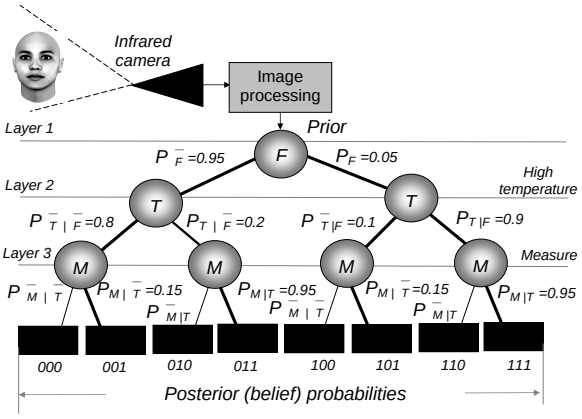


Fig. 5. Probabilistic network as belief tree.

4 Stochastic model

Stochastic model is a typical computational paradigm, which utilizes statistical averaging in data representation and manipulation, such as addition, subtraction, multiplication, and division. Because of data averaging, these operations are highly immune to noise.

A binary stochastic pulse stream is defined as a sequence of binary digits, or bits. The information in a pulse stream is contained in the primary statistics of the bit stream, or the probability of any given bit in the stream being a logic 1. Hence, the output of a gate is generally in the form of a nonstationary Bernoulli sequence (random process of repeated trials with two possible outcomes; this process is characterized by the binomial distribution) [8, 4, 10]. Such a sequence can be considered in probabilistic terms as a *deterministic signal with superimposed noise*. Suppose that the statistical characteristics of these streams are known, and can be measured. These streams carry a signal by statistical characteristics (a single event carries very little information, which is not enough for decision making).

Example 5. *Let the binary variables x_1 and x_2 correspond to the stochastic pulse signals with the means $E(x_1)$ and $E(x_2)$. Suppose that these pulse streams are independent. It is possible to find some logic operations that correspond to the sum $E(x_1)+E(x_2)$ and the product $E(x_1)\times E(x_2)$.*

If the input stochastic streams are independent (technically, this means that independent generators of random pulses are used with some additional tools for decorrelation of signals), and are represented by $E(x_1)$ and $E(x_2)$, the output of gate is described by the equation $E(f)=E(x_1)\times E(x_2)$. The values are normalized into the range $[0,1]$.

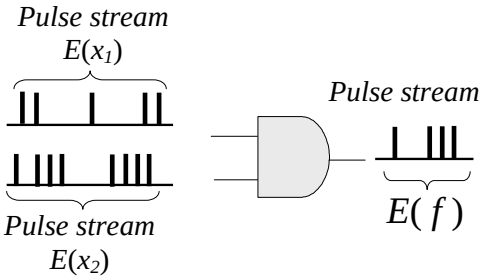


Fig. 6. Stochastic pulse stream model of computing.

The stochastic computer introduces its own errors in the form of *random variance*. If we observe a sequence of N logic levels and k of them are 1, then the estimated probability is $\hat{p}=k/N$. The sampling distribution of the value of k is binomial, and, hence, the standard deviation of the estimated probability $\hat{p}$ from the true probability p is $\sigma(\hat{p})=[p(1-p)/N]^{1/2}$. Therefore, the accuracy in estimation of the generated probability increases as the square root of the length, or time, of computation.

Let $p_1=E(x_1)$ and $p_2=E(x_2)$, then:

(a) The AND gate $f=x_1x_2$ is modeled by $E(f)=p_1p_2$, if input pulse streams are

independent, and by $E(f) = p_1p_2 + K_{x_1,x_2}$ otherwise;
(d) The EXOR gate $f = x_1 \oplus x_2$ is modeled by $E(f) = p_1 + p_2 - 2p_1p_2$, if the input pulse streams are independent, and by $E(f) = p_1 + p_2 - 2p_1p_2 - 2K_{x_1,x_2}$ otherwise.
(e) The XNOR gate $f = \overline{x_1 \oplus x_2}$ is modeled by $E(f) = 1 - p_1 - p_2 + 2p_1p_2$ if the input pulse streams are independent. Here, K_{x_1,x_2} is correlation function.

The precision of computing depends on the size of the stochastic sequence. This effect can be evaluated by standard statistical techniques. Let X be a binomial random variable; then the limiting form of the distribution is the normal distribution. Given a precision of computation ε, the result of computing must satisfy the equation $|k/n - p(x)| \leq \varepsilon$. The confidence interval is

$$-\varepsilon \sqrt{\frac{n}{p(x)q(x)}} \leq \frac{k - np(x)}{\sqrt{np(x)q(x)}} \leq \varepsilon \sqrt{\frac{n}{p(x)q(x)}}$$

The size of Bernoulli stream is $n = (z_{\frac{\alpha}{2}} pq/\varepsilon)^2$. In practice, the size n varies from hundreds to thousands, depending on the required precision of computations.

The most reasonable stochastic pulse encoding models are *one-bit adders* and *one-bit multipliers*. A simple rule for describing the model is used, such that it replaces the Boolean variable x_i in an arithmetic equation by the mean $E[x_i]$ of the stochastic sequence [8, 14, 17, 28].

5 Decision diagram model

A decision diagram is a graphical data structure in the form of a rooted directed acyclic graph, consisting of the root node, a set of non-terminal nodes and a set of terminal (constant) nodes connected via directed edges (links). The topology of a decision diagram is characterized by the parameters such as size, number of non-terminal nodes, number of links, and shape. The computing paradigm underlying decision diagrams is based on the hierarchical decomposition of a logic function (each level corresponds to a single variable). Each path from the root node to a terminal node represents a term in the algebraic description of the function.

An arbitrary logic function can be implemented using decision diagrams, in which nodes are multiplexers, or switches. The library of multiplexer-based implementations is given in Fig. 7. This model is a good candidate for implementation using molecular switches [11].

5.1 Experiments

The goal of this experiment is to model some unreliable gates with the introduction of noise of various scales and observe the deviation of the output value from the

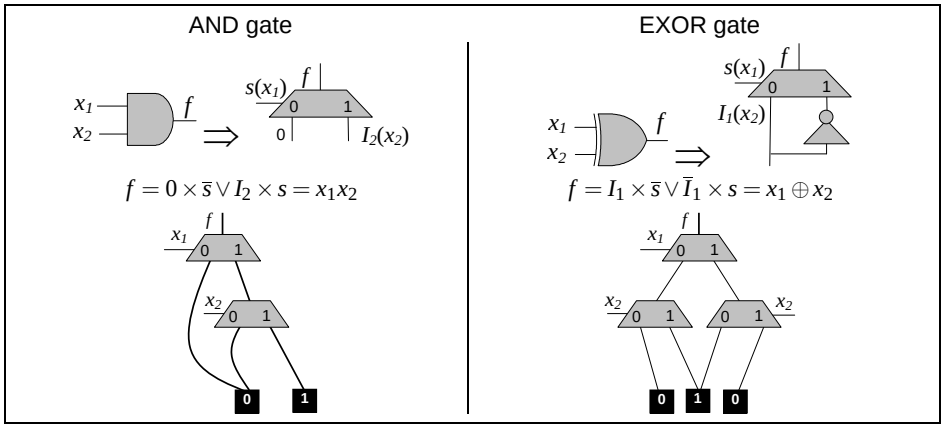
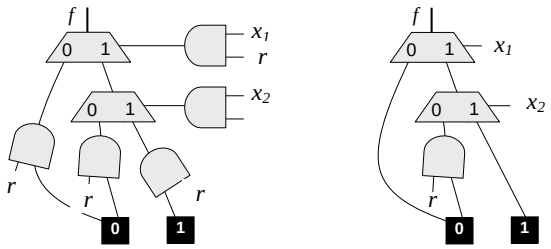


Fig. 7. Switch based AND/EXOR library.

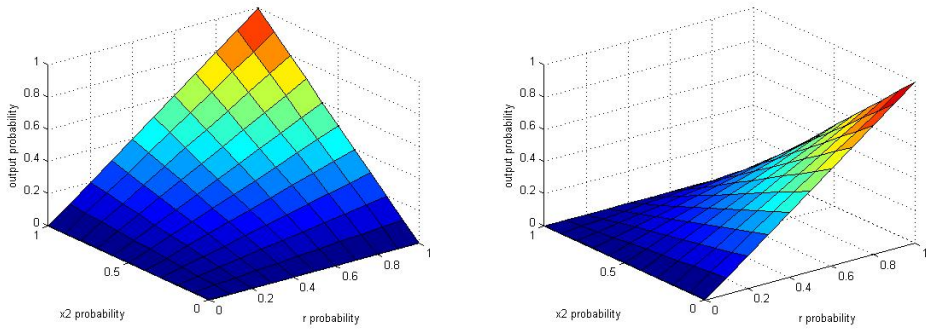
actual one. We consider line noise (any line connecting nodes of the diagram) and node noise (any node within diagram) models. The noise (r) is either additive (modeled via OR gate) or multiplicative (modeled via AND gate). The latter noise model is shown in the form of AND gates, added to the lines and to the selected input of multiplexers in Fig. 8a.



(a) (b)

Fig. 8. Complete (a) and one-line (b) noise-injected AND gate model.

Fig. 9 shows the results of simulation of the output probability (axis Z) for AND (a) and EXOR (b) while varying multiplicative noise on one line only. The line between terminal node 1 and the lower MUX was chosen for for AND gate model, and the line between terminal node 1 and the lower right MUX for EXOR gate model, while $p(x_1 = 1) = 1$ (input x_1 is constant 1), and the probabilities $p(x_2)$ and $p(r)$ vary between 0 and 1 on axes X and Y.



(a) (b)

Fig. 9. Graphical representation of the probabilistic AND (a) and EXOR (b) model based on noise-injected decision diagram.

6 Neuromorphic model

Neuromorphic networks are hardware implementation of artificial neural networks, they resemble cooperative phenomena, and can process probabilistic, noisy, or inconsistent information [13].

The Hopfield computing paradigm is based on the concept of *energy minimization* in a stochastic system [9]. Control over the type of logic function is exercised by the threshold θ and the weights $w_i \in \{1,-1\}$ in the arithmetic sum representing the output value, $f = w_1 \times x_1 + w_2 \times x_2 - \theta$.

Hopfield networks are capable of reliable computing, despite imperfect neuron cells and degradation of their performance. This is because the degraded neuron cells store information (in weights, thresholds, and topology) in a *distributed* (or redundant) manner. A single "portion" of information is not encoded in a single neuron cell but is rather spread over many. Boolean function is computed via a process called "simulated annealing". A value of a Boolean function, given an assignment of its Boolean variables, is computed through relaxation of the neuron cells in the network, while the initial "temperature" of the network is given.

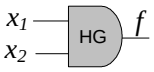
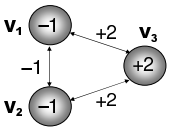
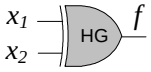
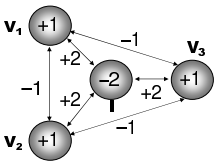
A set of fundamental two-input logic functions can be implemented as a three-node Hopfield network. The logic functions AND and EXOR are shown in Table 2. The logic function AND can be implemented using a three-node Hopfield network, and EXOR requires a four-node Hopfield network. The Boolean function of a logic gate is encoded in the energy function, using connection weights and neuron threshold. The energy function of the network is defined as follows:

$$E = -\frac{1}{2}\sum_{i}^{n}\sum_{j}^{n} x_i x_j C_{ij} + \sum_{i}^{n} x_i N_i,$$

where x_k is the state of cell k, C_{ij} is the connection weight between the cells i and j, N_i is the neuron's threshold value, and n is the total number of cells, excluding *Bias* cell; and the factor 1/2 ensures that we do not count each connection twice ($i \neq j$).

All the global minima of the energy function correspond to the correct Boolean output, while the 'False' is encoded as having an energy value greater than the global minimum. In the process of relaxation, the Hopfield networks try to achieve the global minimum and, therefore, the correct output.

Table 2. Hopfield models of AND and EXOR gates.

Gate	Energy function	Model	Compatibility truth table
AND $f = x_1 x_2$ 	$E = -v_1 - v_2 + 2v_3$ $+v_1v_2 - 2v_1v_3$ $-2v_2v_3$		v_1 v_2 v_3 E f −1 −1 −1 −3 −1 −1 −1 1 9 False −1 1 −1 −3 −1 −1 1 1 1 False 1 −1 −1 −3 −1 1 −1 1 1 False 1 1 −1 1 False 1 1 1 −3 1
EXOR $f = x_1 \oplus x_2$ 	$E = v_1 + v_2 + v_3$ $-2i + v_1v_2 + v_1v_3$ $+v_2v_3 - 2iv_1 - 2iv_2$ $-2iv_3$		v_1 v_2 v_3 I E f −1 −1 −1 −1 −4 −1 −1 −1 −1 1 4 False −1 −1 1 −1 −2 False −1 −1 1 1 −2 False −1 1 −1 −1 −2 False −1 1 −1 1 −2 False −1 1 1 −1 4 False −1 1 1 1 −4 1 1 −1 −1 −1 −2 False 1 −1 −1 1 −2 False 1 −1 1 −1 4 False 1 −1 1 1 −4 1 1 1 −1 −1 4 False 1 1 −1 1 −4 −1 1 1 1 −1 14 False 1 1 1 1 −2 False

6.1 Network design using Hopfield gates

To obtain a Hopfield network, capable of implementing large truth table, a technique to map the logic networks directly into Hopfield networks has been proposed in [7].

Rule I: Hopfield gate merging

Connecting an output of one Hopfield gate to the input of another (cascading the gates) is performed by merging the corresponding "output" cell of the first gate with the "input" cell of the second one. The threshold value of the new cell is the cumulative threshold of the merged cells.

Rule II: Hopfield gate merging

Connecting an output of one Hopfield gate to the input of another (cascading the gates), while the second input of the other gate is also the input of the first one, is performed by merging the corresponding "output" cell of the first gate with the "input" cell of the second one, as well as the both "input" cells. The threshold value of the two new cells is the cumulative threshold of the merged cells.

Example 6. *Rule I: While connecting AND and EXOR gate, both cells G1 are merged in a new cell, which threshold value is 2+1=3, obtained by adding the threshold values of the merged cells (Fig. 10, upper plane).*

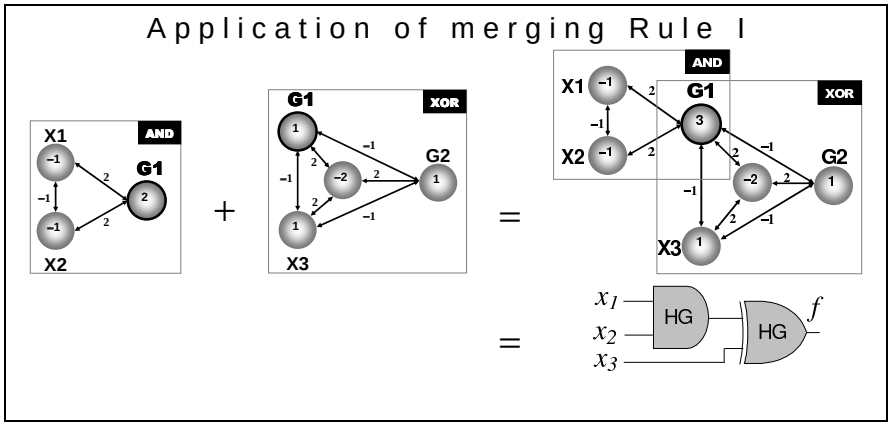


Fig. 10. Taxonomy of neuromorphic AND/EXOR network design: application of merging rules in implementation of function $f = x_1x_2 \oplus x_3$.

Example 7. *Rule II: Input x_2 is common for both AND and EXOR gates, so their cascading involves merging gates $X2$ and $G1$, and their threshold are cumulative values, -1+1=0 and 2+1=3, respectively (Fig. 10, lower plane).*

6.2 Noise Model for the Hopfield networks

In this paper, to investigate fault tolerance of the Hopfield networks, a discrete noise is added in the form of noise probability. Noise probability is defined here as

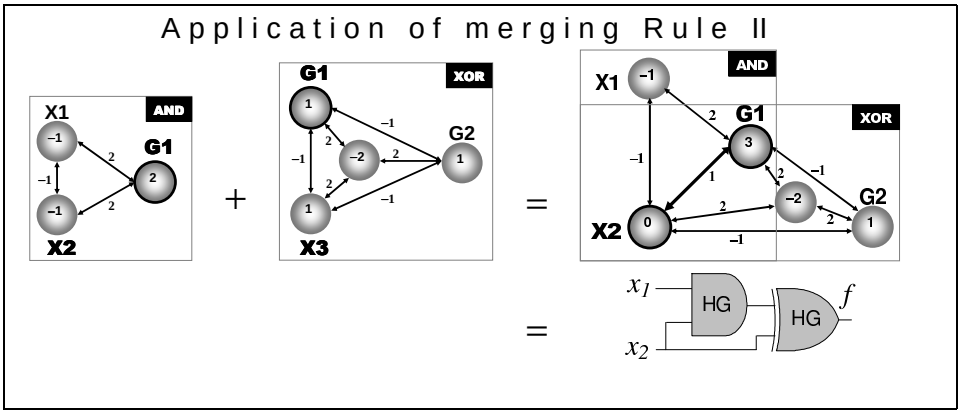


Fig. 10. (Continue) Taxonomy of neuromorphic AND/EXOR network design: application of merging rules in implementation of function $f = x_1x_2 \oplus x_3$.

the probability that a neuron cell is affected by noise resulting in a bit flip, called "a state flip" in terms of Hopfield networks. In other words, change of the state is modelled by the uniformly distributed discrete random signal. For example, noise level 0.1 (10%) means that the probability of changing the current state of the neuron cell is 0.1.

There are two updating rules that applies to the Hopfield network: (a) the Hopfield deterministic rule, and (b) stochastic Boltzmann rule, or simulated annealing. For example, the Boltzmann updating rule is based on assumption of an uncertainty of state of a particular cell during the updating process. Instead of setting the state of cell k deterministically, the process uses the probability $p(k)$ that cell k takes state 1. It implies the following steps: (a) Select temperature T; (b) Randomly select a cell k, (c) Calculate $\Delta E(k) = \sum_i^n x_i c_{ik} - N_k$. If $\Delta E(k) > 0$, then calculate the probability that cell k takes state 1: $p(k) = \frac{1}{1+e^{-\Delta E(k)/T}}$. Else set the state of cell k to -1; (d) Repeat steps (b) through (d) until the state of cell does not change for a certain interval (this is also called stable state condition), for example, 20 iterations.

6.3 Experimental Results

In this study, we have compared the robustness of the Hopfield models for EXOR gate using the 4-node Hopfield EXOR gate ($f = x \oplus y$), and the alternative implementations of EXOR: one in the form of $f = \overline{x}y \vee x\overline{y}$ using the network of one 2-node Hopfield NOT gate, two 3-node AND gates and one 3-node OR gate, and another in the form $f = \overline{x \cdot xy} \vee x\overline{y \cdot y}$.

Example 8. *Given the node noise probability ranging between 0 and* 0.5*, the number of iteration to reach the stable state vary for the EXOR Hopfield gate itself*

(Figure 11, A), and for the implementation of EXOR on a network of NOT, AND and OR gates (B), as well as a network of NAND, AND and OR gates. The latter two seem to need less iteration, while the noise level increases beyond 0.2, and is significantly different for noise level 0.5. The EXOR gate itself (which requires four neurons) required much more iterations to achieve the correct output, while the networks of NOT, AND, OR or NAND gates, each consisting of two (NOT only) or three neurons), connected in the network with total 10 (B) and 15 neurons (C), respectively, are mush faster to reach the stable state.

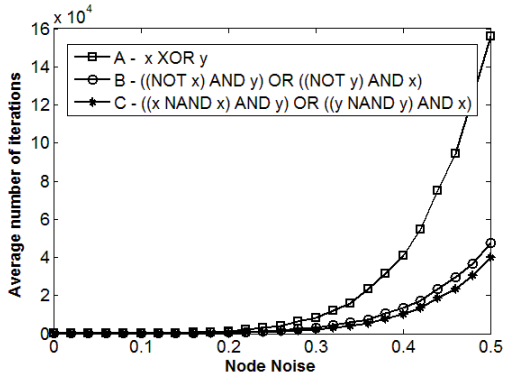


Fig. 11. Behavioral characteristics of neuromorphic model: X and Y axes correspond the level of noise and number of required iterations to reach stable state condition, respectively; the curves A, B and C correspond to EXOR gate, and the network of NOT, AND, OR, and NAND, AND and OR, implementing the same EXOR function, respectively.

On the other hand, even if the EXOR Hopfield gate is much slower in achieving the stable condition, it is more robust than the network of AND, OR or NAND gates, implementing the same function EXOR.

Example 9. *Given the node noise probability ranging between 0 and* 0.5*, the probability of achieving the correct output is better for the EXOR Hopfield gate itself (Figure 12, A), and for the implementation of EXOR on a network of NAND, AND and OR gates (C), while the probability of achieveing the correct output for the network of NOT, AND and OR gates (B) is only around 0.6.*

Another experimental study of performance on the Hopfield networks, implementing simple AND-EXOR expressions, have been performed using the circuits $x_1x_2 \oplus x_3x_4$. This experiment compares the probability of achieving the correct output of the benchmark, for both logic network and Hopfield network with noise.

Fig. 13 shows the probability of correct output with respect to the noise probability ranging from 0 to 0.5. The stable state condition for the Hopfield network

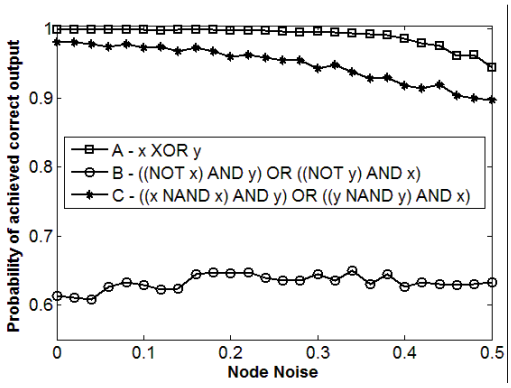


Fig. 12. Behavioral characteristics of neuromorphic model: X and Y axes correspond the level of node noise and the probability of achieveing the correct output, respectively; three curves correspond to EXOR Hopfield gate (A), the network of NOT, AND and OR gates (B), and the network of NAND, AND and OR gates (C) to implement EXOR, respectively.

is set to 15, 10, and 5 iterations. The results show that probability of achieving the correct output is higher for the Hopfield network compared against the logic network, as the noise probability increases.

Example 10. *Given the noise probability* 0.2*, the Hopfield network converges to stable state condition in* 10 *iterations, thus achieving the correct output with probability* 0.96*, while the logic network is only able to achieve the correct output with probability* 0.81 *(the difference is* 15%*). It is also observed that by increasing the the number of iterations (for stable state condition) from* 10 *and* 15*, the Hopfield network is able to achieved probability of correct output* $>$ 99%*.*

This shows that Hopfield implementation is capable of operating with high accuracy in the noisy environment.

Example 11. *Given the extremely high noise probability* 0.5*, the Hopfield network is capable of achieving the correct output with probabilities* 0.9 *and* 0.97*, for stable state condition* 10 *and* 15 *iterations, respectively. On the other hand, the network is only able to achieve the correct output with probability* 0.55*. Therefore, for the logic gate implementation, the solution breaks down to a random signal at the high noise probability of* 0.50*. However, the Hopfield network implementation achieves the correct output value with probability* 0.97*.*

These results confirms that the behavior of Hopfield networks and its modification such as Boltzmann machine demonstrates high fault tolerance in the presence of critical noise in a part or in all neurons of the network.

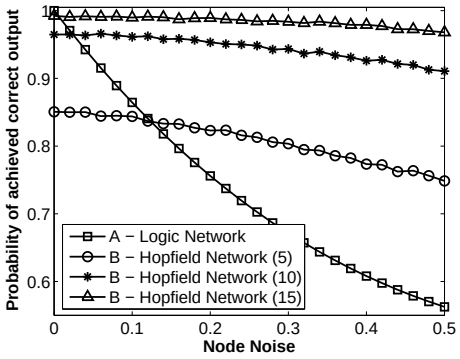


Fig. 13. Behavioral characteristics of neuromorphic model: X and Y axes correspond to the level of noise and relative frequency of reaching the correct solution, respectively; four curves correspond to logic network, and Hopfield network with stable conditions at 5, 10, and 15 iterations, respectively.

Fig. 14 shows the average number of iterations required to reach the stable state condition.

Example 12. *Given the noise probability* ≤ 0.2*, the Hopfield network is able to reach stable state condition in less than* 200 *iterations. For the high noise probability* 0.5*, the network requires less than* 400 *iterations to achieve the probability of obtaining the correct output greater than* 0.9*. In this particular case, the Hopfield implementation requires* 2600 *iterations to achieve the greater than* 0.97 *probability of obtaining the correct output.*

7 Markov random field model

The basic definitions for MRF models of logic gates are introduced below based on [3, 5].

7.1 Basic definitions

Let $X = \{x_1, \ldots, x_m\}$ be a set of random variables. This set is called a *random field* if a variable x_i takes value y_i. The joint event $X = y = \{y_1, \ldots, y_m\}$ is called a *configuration* of X, corresponding to a realization of the field. A set X is called a *Markov network*, or *Markov random field*, if (a) the probability that a random variable x_i takes the value y_i is defined, $p(x_i) > 0$, and (b) the local characteristics of X are specified. The joint probability of X can be formulated in terms of the associated *clique* of the graph structure. Clique is defined as a set of nodes where each node in

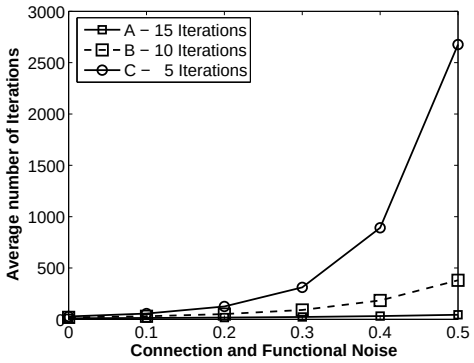


Fig. 14. Behavioral characteristics of neuromorphic model: X and Y axes correspond the level of noise and number of required iterations to reach stable state condition, respectively; three curves correspond to 5, 10, and 15 iterations, respectively.

the set connects to other nodes in the graph. The conditional probability of a node only depends on its neighborhood. This model considers the effects of noise and other uncertainty factors on a node by considering the conditional probabilities of energy values with respect to its clique.

7.2 Concept of energy of a logic function

In MRF model, computing the simplest logic operations in the presence of noise, such as NOT, AND, NAND, OR, is performed using notation of energy. The "energy" of a logic function is accumulated by "potentials" of cliques. This terminology comes from statistical physics.

Given a clique $c \in C$, a *clique potential*, $V_c(x)$, is a non-negative real-valued function of this particular clique. It follows from this definition that a logic function must be represented in a particular form, such that arithmetic operations are used instead of logic ones.

Example 13. *A complete graph with three nodes v_1, v_2 and v_3 is a clique, because all distinct pairs of nodes, v_1, v_2, v_1, v_3, and v_2, v_3 are neighbors.*

A sum of clique potentials, $V_c(x)$, over all possible cliques C is called the *energy function*, and is denoted as

$$E(x) \quad = \quad \sum_{c \in C} V_c(x) \tag{5}$$

The energy function is a quantitative measure of the global quality of the solution; the correct solution corresponds to the maximum of energy function. Note

that in this paper, we consider logic primitives $f(x_1, x_2)$, which are represented by complete graph (a single clique model); that is, energy $E(x)$ is equal to a clique potential: $E(x) = V_{c=1}(x) = V(x)$.

Gibbs joint probability distribution or *Gibbs random field*, is defined in the form

$$p(X = x) = \frac{1}{Z} \exp\left(\frac{E(x)}{kT}\right) \tag{6}$$

where Z is a *scaling* factor (to normalize the total probability to 1) and kT is the temperature factor (in physics view, T is "temperature", and k is Boltzmann constant).

Given a joint probability distribution $p(x_1, \ldots, x_r)$, the **marginal distribution** is defined as follows:

$$p(x_1, \ldots, x_s) = \sum_{x_{s+1}, \ldots, x_r} p(x_1, \ldots, x_r), \;\; s \leq r$$

Marginal distribution can be viewed as a projection of the joint distribution on a smaller set of variables.

Hence, Gibbs random fields is defined by a joint probability. On the contrary, the MRF is based on a conditional probability. The equivalence between the MRF and the Gibbs random field can be established by the following theorem.

Hammersley-Clifford theorem states that a set of random variables $X = \{x_1, \ldots, x_s\}$ is a MRF, if and only if X is Gibbs distributed. That is, the global probabilistic characteristics can be computed using local interactions via factorization. Specifically, the joint probability of an MRF can be constructed from the local conditional probabilities.

Logic function is incorporated into a Markov network using an *arithmetic* forms [24, 23].

7.3 Algorithm for synthesis of MRF models of logic gates

Given the compatibility truth vector $\mathbf{U}$ of a Boolean function f of n variables, the vector of coefficients $A = (a_1, a_2, \ldots, a_n)$ is calculated using the arithmetic transform [23]:

$$\mathbf{A} \;\; = \;\; \mathbf{A}_{2^n} \cdot \mathbf{U} \tag{7}$$

where matrix $\mathbf{A}_{2^n}$ is formed as follows

$$\mathbf{A}_{2^n} \;\; = \;\; \bigotimes_{j=1}^{n} \mathbf{A}_{2^j}, \;\; \mathbf{A}_{2^1} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

Arithmetic form of r-input, $x_1, x_2, \ldots, x_r$, logic gate f, given by spectral coefficients, $a_1, a_2, \ldots, a_n$ is defined by the polynomial:

$$f = \sum_{i=0}^{2^r-1} a_i \cdot (x_1^{i_1} \cdots x_r^{i_r}), \tag{8}$$

where i_j is the j-th bit $1, 2, \ldots, r$, in the binary representation of the index $i = i_1 i_2 \ldots i_r$; $x_j^{i_j} = 1$ if $i_j = 0$, and $x_j^{i_j} = x_j$ if $i_j = 1$.

Example 14. *Arithmetic representation of a 3-input logic gate ($n = 3$) is defined by Equation (8) as follows:* $f = a_0 + a_1 x_3 + a_2 x_2 + a_3(x_2 x_3) + a_4 x_1 + a_5(x_1 x_3) + a_6(x_1 x_2) + a_7(x_1 x_2 x_3)$.

Consider a 2-input logic gate of a function L. The algorithm for designing the MRF model of this gate is as follows.

The MRF model design for a 2-input logic gate

Input data: (a) Graph model $\{v_1, v_2, v_3\}$ (complete graph) and (b) function L.
Step 1: Form the compatibility truth table.
Step 2: Calculate the energy function, $E(v)$, using Fourier-like transform, in particular, arithmetic transform (7).
Step 3: Specify Gibbs distribution (6) by substituting the energy function, $E(v)$, into it.
Step 4: Calculate the marginal probability distribution of the output node.
Output: An MRF model of L gate.

Designing the MRF model for the AND gate is illustrated by the following example (Fig. 15).

Table 3 provides design results for two-input OR and EXOR gate. It is shown in Table 3, that the valid input/output states have higher clique energies than invalid states to maximize the probability of a valid energy state at the nodes: $E(v)$ of valid states is 1, and for any invalid state, this value is 0. The logic margin in this case is the difference between the probabilities of occurrence of a logic low and a logic high, which, if high, leads to a higher probability of correct computation.

Further applying the belief propagation algorithm, the energy distributions and entropy at different nodes of the network can be calculated [24].

These MRF models show that maximization of logic state probability can be viewed as a process of energy maximization. Note that energy minimization process can be achieved by sign manipulation in compatibility truth table and Gibbs

Design example: MRF model for logic 2-input AND gate

Given:

(a) Complete noncausal graph, $\{v_1, v_2, v_3\}$
(b) Logic function of gate L

Design an MRF model of L gate.
Solution:
Step 1: Form the compatibility truth table:

v_1	v_2	v_3	**U**	Comment
0	0	0	1	
0	0	1	0	Undesirable
0	1	0	1	
0	1	1	0	Undesirable
1	0	0	1	
1	0	1	0	Undesirable
1	1	0	0	Undesirable
1	1	1	1	

Step 2: Calculate the energy function, $U(v)$:
(a) Calculate a clique potential using arithmetic transform (7):

$$\mathbf{A} = \mathbf{A}_{2^3} \cdot \mathbf{U} = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{array}\right] \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{array}\right] = \left[\begin{array}{c} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 2 \end{array}\right]$$

(b) Convert the vector of coefficients $\mathbf{A}$ into algebraic form using equation (8):
$E(v) = 1 - v_3 - v_1 v_2 + 2 v_1 v_2 v_3$
Step 3: Specify Gibbs distribution:

$$p(v_1, v_2, v_3) = \frac{1}{Z} \exp\left(\frac{1 - v_3 - v_1 v_2 + 2 v_1 v_2 v_3}{kT}\right)$$

Step 4: (a) Calculate marginal distribution by summing over all possible states of v_1:

$$\begin{aligned} p(v_2, v_3) &= \frac{1}{Z_1} \sum_{v_1 \in 0,1} \exp\left(\frac{1 - v_3 - v_1 v_2 + 2 v_1 v_2 v_3}{kT}\right) \\ &= \frac{1}{Z_1}\left[\exp\left(\frac{1 - v_3}{kT}\right) + \exp\left(\frac{1 - v_3 - v_2 + 2 v_2 v_3}{kT}\right)\right] \end{aligned}$$

Fig. 15. MRF-based model for AND gate.

(b) Calculate marginal distribution of v_3 by summing over all possible states of v_2:

$$p(v_3) = \frac{1}{Z_2} \sum_{v_2 \in 0,1} \left[\exp\left(\frac{1 - v_3}{kT} \right) + \exp\left(\frac{1 - v_3 - v_2 + 2v_2 v_3}{kT} \right) \right]$$
$$= \frac{1}{Z_2} \left[3\exp\left(\frac{1 - v_3}{kT} \right) + \exp\left(\frac{v_3}{kT} \right) \right]$$

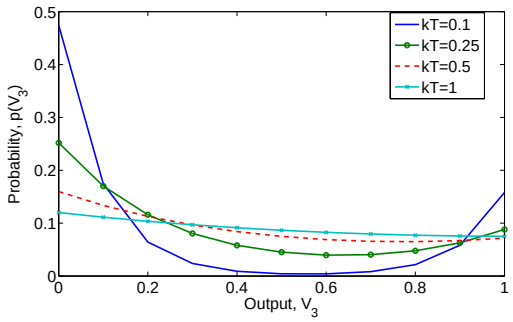


Fig. 15. (Continue) MRF-based model for AND gate.

distribution. Therefore, a bistable storage element with feedback is an appropriate hardware architecture for binary logic [19, 31].

8 Conclusion and discussion

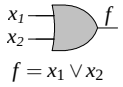
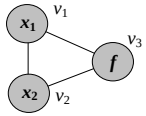
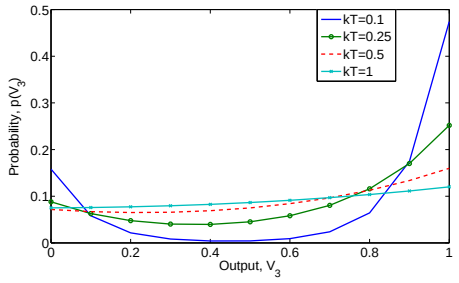
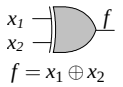
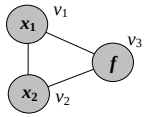
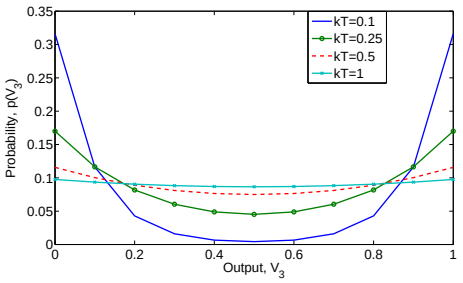
An extended vision of probabilistic network design, including probabilistic AND-EXOR networks, is introduced. This interdisciplinary view includes the field of coding for error correction based on statistical techniques, and belief networks for decision making. In these fields, data processing is based on probabilistic and statistic techniques, using both discrete and analog technology for implementation of computing networks over the libraries of probabilistic logic gates.

There exists a diversity in terminology related to probabilistic computing. For example, the term “probabilistic EXOR gate” addresses the following meanings:

(a) traditional EXOR gate assuming random input and output signals (we proposed to distinguish this meaning as “behavioral model”), and

(b) computing device which operates with probabilities (real numbers) with respect to EXOR switching function (in our systematization, it is a “ belief model”).

Table 3. Components of the MRF model of binary gates.

Gate	Graph model	Compatibility truth table	Clique potential (arithmetic form of gate)	Probabilistic behavior
		v_1 v_2 v_3 U **0 0 0 1** 0 0 1 0 0 1 0 0 **0 1 1 1** 1 0 0 0 **1 0 1 1** 1 1 0 0 **1 1 1 1**	$\mathbf{U} = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 1]^T$ $\mathbf{A} = [1\ -1\ -1\ 2\ -1\ 2\ 1\ -2]^T$ $E(v) = 1 - v_1 - v_2 - v_3 + v_1 v_2 + 2v_1 v_3 + 2v_2 v_3 - 2v_1 v_2 v_3$	
		v_1 v_2 v_3 U **0 0 0 1** 0 0 1 0 0 1 0 0 **0 1 1 1** 1 0 0 0 **1 0 1 1** **1 1 0 1** 1 1 1 0	$\mathbf{U} = [1\ 0\ 0\ 1\ 0\ 1\ 1\ 0]^T$ $\mathbf{A} = [1\ -1\ -1\ 2\ -1\ 2\ 2\ -4]^T$ $E(v) = 1 - v_1 - v_2 - v_3 + 2v_1 v_2 + 2v_1 v_3 + 2v_2 v_3 - 4v_1 v_2 v_3$	

Thus, the main goal of our study was to systematize the known approaches to probabilistic logic gate design. We compared six models of probabilistic logic gates: behavioral, belief network, decision diagram, stochastic, neuromorphic, and Markov random field. In traditional logic network design, the behavior model dominates. However, there are other forms of probabilistic description of logic gate behavior in the presence of noise, for example [6]. These models are relatively new, and have been introduced in the context of computational models for novel deep submicron and nano technologies.

Different design taxonomies are required to construct logic networks using libraries of probabilistic gates. Theoretical platform of these techniques is traditional logic design, extended by probabilistic and statistical methods.

We mentioned only two hardware-centered belief networks: Bayesian networks for security applications, such as real time decision-support assistants [30], and stochastic decoder for turbo code [1, 2, 12]. The requirements to the performance, power consumption, and size of these devices, especially for mobile systems (cell phones, iPODs, hand players, personal computers, etc.) are very strict. However, these are low-precision computations which can be considered as a key argument for analog implementation of these devices. Note that new technologies, such as molecular electronics, are based on inherently analog phenomena. In addition, random physical and chemical phenomena explain why probabilistic computation is a natural way in the era of nano technology [10, 11].

One of the feasible candidates for future technologies is neuromorphic networks [13]. The neuromorphic model, based on Hopfield network with Boltzmann updating rule [27], is robust to noise, as shown in this paper via experimental study. The latter confirm that the behavior of Hopfield networks and its modification, the Boltzmann machine, demonstrates high fault tolerance in the presence of critical noise in a part or in all nodes and interconnects of the network.

However, multivalued extension of Hopfield-based logic networks is very complicated. In contrast, the MRF model can be easy generalized for multivalued logic. Example is given in Table 4. We used 0-polarity arithmetic transforms for the 3-valued NOT gate [23]. Extension for an arbitrary library of multivalued gates is straightforward.

Other models of fault-tolerant logic gates exist, for example, *polymorphic* gates for sensor-based systems [26]. Polymorphic gate is a multi-functional logic device that performs logic function f_j, $j = 1,2,\ldots,m$, if its control input is activated by value I_j of the signal. In particular, 2-function polymorphic gate is defined as follows:

$$\texttt{2-function gate} = \begin{cases} f_1, & \text{if } I_1 \text{ control value;} \\ f_2, & \text{if } I_2 \text{ control value.} \end{cases}$$

Table 4. Markov random field model for ternary ($m = 3$) NOT gate.

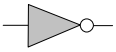
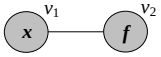
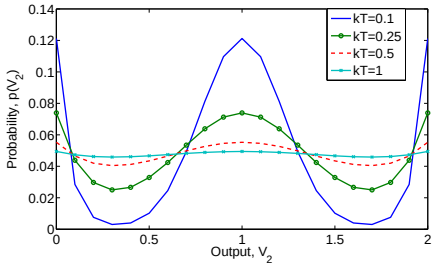
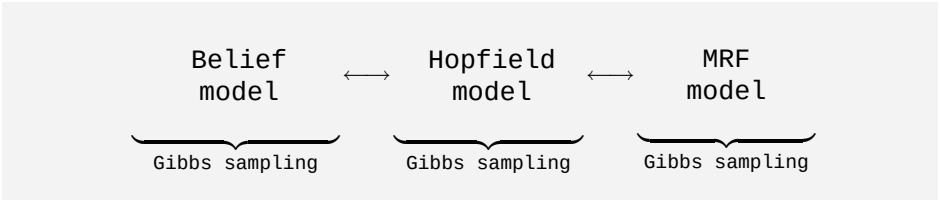
Ternary gate	Graph	Energy function
 x | $\bar{x}$ **0** | 2 **1** | 1 **2** | 0		$\mathbf{U} = [0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0]^T$ $\mathbf{A} = [0\ -2\ 2\ -2\ 22\ -12\ 2\ -12\ 6]^T$ $E(v) = {}^1/_4(-2v_2 + 2v_2^2 - 2v_1 + 22v_1v_2$ $-12v_2^2v_1 + 2v_1^2 - 12v_1^2v_2 + 6v_1^2v_2^2)$

Table 4. (Continue) Markov random field model for ternary ($m = 3$) NOT gate.

Probabilistic behavior


Polymorphic Reed-Muller networks are designed using 3-functional polymorphic AND/OR/EXOR gates (Fig. 16). Such gates perform the two-inputs AND ($I = 0$), OR ($I = 1$), and EXOR ($I = 2$) function with probabilities $p(I = 0)$, $p(I = 1)$, and $p(I = 1)$, respectively. The probabilities of inputs ("1"s) of all gates are the same, that is, $p(x_1)$ and $p(x_2)$. For the best of our knowledge, behavior of polymorphic gates under these conditions has not been studied yet.

Finally, there are equivalents between three basic probabilistic models, which can be established using Gibbs distribution or Gibbs sampling method [18]:



It follows from this similarity that we can expect similar numerical results while

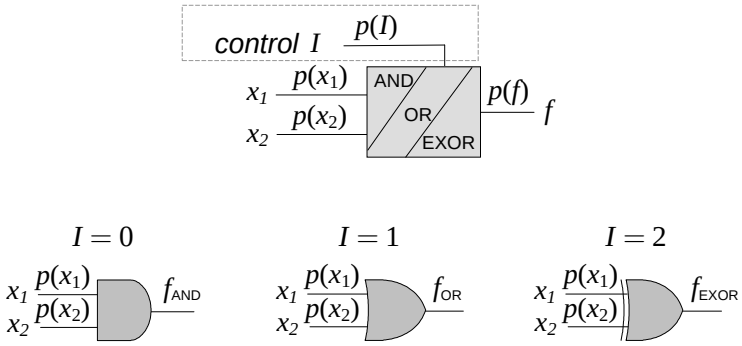


Fig. 16. Two-input 3-functional polymorphic AND/OR/EXOR gate and equivalent logic AND ($I = 0$), OR ($I = 1$), and EXOR ($I = 2$) gates.

using various models. However, the techniques for achieving these results, the algorithms, behavioral characteristics, and hardware implementation are different.

Acknowledgment

This work was partially supported by Natural Sciences and Engineering Research Council of Canada, and NATO linkage grant "Intelligent Assistance Systems: Multisensor Processing and Reliability Analysis". G. Tangim acknowledges the Information and Communication Technologies Recruitment Scholarship, University of Calgary. We acknowledge also unknown reviewers for useful remarks and proposals for improvements of the paper.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding turbo-codes", *Proc. IEEE Int. Conf. Commun.*, Geneva, pp. 1064–1070, May, 1993.

[2] C. Berrou and V. Gripon, "Coded Hopfield networks", *Proc. 6th Int. Symp. Turbo Codes and Iterative Information Processing*, 2010.

[3] J. E. Besag, "Spatial interaction and the statistical analysis of lattice systems", *J. Roy. Stat. Soc.*, series B, vol. 36, no. 3, pp. 192–236, 1974.

[4] B. D. Brown and H. C. Card, "Stochastic neural computing I: Computational elements", *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 891–905, 2001.

[5] H. Derin and P. A. Kelly, "Discrete-index Markov-type random process", *Proceedings of the IEEE*, vol. 77, no. 10, pp. 1485–1510, 1989.

[6] T. J. Dysart and P. M. Kogge, "Analysisng the Inherent Reliability of Moderately Sized Magnetic and Electrostatic QCA Circuits Via Probabilistic Transfer Matrices", *IEEE Trans. VLSI Systems*, vol. 17, no. 4, pp. 507–516, 2009.

[7] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, Neural Models and Algorithms for Digital Testing. Kluwer, Dordrecht, 1991.

[8] B. R. Gaines, "Stochastic computing systems", in "Advances in Information Systems Science", J. T. Tou, Ed., Plenum, New York, vol. 2, chap. 2, pp. 37–172, 1969.

[9] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of National Academy of Sciences*, USA, vol. 79, pp. 2554–2558, 1982.

[10] S. E. Lyshevski, S. N. Yanushkevich, V. P. Shmerko, et al., "Computing Paradigms for Logic Nanocells.", *J. Computational and Theoretical Nanoscience*, vol. 5, pp. 2377–2395, 2008.

[11] S. E. Lyshevski, V. P. Shmerko, M. A. Lyshevski, et al., "Neuronal processing, reconfigurable neural networks and stochastic computing", *Proc. IEEE Conf. Nanotechnology*, Arlington, TX, 2008.

[12] H.-A. Loeliger, F. Lustenberger, M. Helfenstein, et al., "Probability propagation and decoding in analog VLSI", *IEEE Trans. Inf. Theory*, vol.47, no.2, pp.837–843, 2001.

[13] C. Mead, "Neuromorphic electronic systems", *Proc. IEEE*, vol. 78, no 10, pp. 1629–1639, 1990.

[14] C. L. Janer, J. M. Quero, J. G. Ortega, et al., "Fully parallel stochastic computation architecture", *IEEE Trans. Signal Processing*, vol. 44, no. 8, pp. 2110–2117, 1996.

[15] F. V. Jensen, *Bayesian Networks and Decision Graphs*, Springer, 2001.

[16] A. A. Mullin, "Stochastic combinational relay switching circuits and reliability", *IRE Trans. Circuit Theory*, vol. 6, no. 1, pp. 131–133, 1959.

[17] A. F. Murray and A. V. W. Smith, "Asynchronous VLSI neural networks using pulse-stream arithmetic", *IEEE J. of Solid*, vol. 23, no. 3, pp. 688–697, 1988.

[18] P. Myllymäki, "Using Bayesian networks for incoporating probabilistic a priory knowledge into Boltzmann machines", *Proc. SOUTH Conf.*, pp. 97–102, Orlando, 1994.

[19] K. Nepal, R. I. Bahar, J. Mundy, W. R. Patterson, et al., "Designing Nanoscale Logic Circuits Based on Markov Random Fields", *J. Electronic Testing: Theory and Applications*, vol. 23, pp. 255–266, 2007.

[20] K. P. Parker and E. J. McCluskey, "Probabilistic treatment of general combinational networks", *IEEE Trans. Comput.*, vol. 24, no. 6, pp. 668–670, 1981.

[21] U. Kalay, D. V. Hall, and M. A. Perkowski, "A minimal universal test set for self-test of EXOR-sum-of-products circuits", *IEEE Trans. on Comput.*, Vol. 49, Issue 3, pp. 267 - 276, 2000.

[22] P. W. K. Rothemund, N. Paradakis, and E. Winfree, "Algorithmic self-assembly of DNA Sierpinski triangles", PloS Biology — www.plosbiology.org, vol.2, issue 12, e424, pp. 2041–2053, 2004.

[23] V. P. Shmerko, S. N. Yanushkevich, and S. E. Lyshevski, *Computer Arithmetics for Nanoelectronics*, Taylor & Francis/CRC Press, Boca Raton, FL, 2009.

[24] S. Shukla and R. I. Bahar (Eds.), "Nano, Quantum and Molecular Computing: Implications to High Level Design and Validation", Kluwer, 2004.

[25] S. Stanković and J. Astola, "Representation of Resilient Boolean Functions Using Binary Decision Diagrams", *Proc. Reed-Muller 2011 Workshop*, Tuusula, Finland, pp. 93-98, May 2011.

[26] A. Stoica, R. Zebulum, and D. Keymeulen, "Polymorphic electronics", *Proc. 4th Int. Conf. Evolvable systems: from biology to hardware*, Tokyo, Japan, 2001, pp. 291-001.

[27] A. H. Tran, S. N. Yanushkevich, S. E. Lyshevski, et al. "Fault Tolerant Computing Paradigm for Random Molecular Phenomena: Hopfield Gates and Logic Networks", *IEEE Int. Symp. Multi-Valued Logic*, pp. 93 - 98, 2011.

[28] D. E. Van Den Bout and T. K. Miller III, "A digital architecture employing stochastism for the simulation of Hopfield neural nets", *IEEE Trans. Circuits and Syst.*, vol. 36, no. 5, pp. 32–738, 1989.

[29] S. N. Yanushkevich and V. P. Shmerko, "Teaching Reed-Muller techniques in introductory classes on logic design", *FACTA Universitatis.*, ser. Elec. Energ., vol. 20, no. 3, pp. 331–065, 2007.

[30] S. N. Yanushkevich, A. Stoica, and V. P. Shmerko, "Experience of design and prototyping of a multi-biometric early warning physical access control security system (PASS) and a training system (T-PASS)", *Proc. 32nd Annual IEEE Industrial Electronics Society Conf.*, Paris, 2006.

[31] I-C. Wey, Y-G. Chen, C-H Yu, et al., "Design and Implementation of Cost-Effective Probabilistic-Based Noise-Tolerant VLSI Circuits", *IEEE Trans. Circuits and Syst.*, vol. 56, no. 11, pp. 2411–2424, 2009.