Probabilistic Error Modeling for Nano-domain Logic Circuits

Thara Rejimon, Karthikeyan Lingasubramanian and Sanjukta Bhanja
Department of Electrical Engineering
University of South Florida, Tampa, Florida
Email: (rejimon, klingasu,bhanja)@eng.usf.edu

Abstract

In nano-domain logic circuits, errors generated are transient in nature and will arise due to the inherent uncertainty or the unreliability of the computing element itself. Due to these highly likely dynamic errors, it is more appropriate to model nano-domain computing as probabilistic rather than deterministic. We propose a probabilistic error model based on minimal graphical probabilistic model namely Bayesian networks to estimate this expected output error probability, in the context of probabilistic computing. We estimate the overall output error probability by comparing the outputs of a dynamic error-encoded model with an ideal logic model. We use both exact and approximate Bayesian inference schemes for propagation of probabilities. The exact inference shows better time performance than the state-of-the art by exploiting conditional independencies exhibited in the underlying probabilistic framework. However, since exact inference is worst case NP-hard, we propose two approximate inference schemes for medium size benchmarks. We demonstrate the efficiency and accuracy of these approximate inference schemes by comparing estimated results with logic simulation results. We have performed our experiments on *LGSynth'*93 and *ISCAS'*85 benchmark circuits. We explore our probabilistic model to calculate (a) error sensitivity of individual gates in a circuit (b) compute overall exact error probabilities for small circuits (c) compute approximate error probabilities for medium sized benchmarks using two stochastic sampling schemes (d) compare and vet design with respect to dynamic errors (e) characterize the input space for desired output characteristics by utilizing the unique backtracking capability of Bayesian networks (inverse problem) and (f) to apply selective redundancy to highly sensitive nodes for error tolerant designs.

I. INTRODUCTION

The ITRS road-map predicts CMOS device dimensions to reach close to the design limit of 50 nm by 2020. Circuits built with such nano-dimensional devices will face design challenges that have not been much of an issue so far. One such challenge involve dynamic errors in the interconnects and gates.

What is a dynamic error? These errors will be transient in nature, occurring anywhere in the circuit, but hard to detect by regular testing methodologies (since they are not permanent damages). They can be characterized only probabilistically. Each device (logic gate/interconnect) will have certain, non-zero, propensity for an output line error. These error probabilities could be as high as 10% [5]. Traditional error masking by extra logic might not be possible since these extra logic will itself be error-prone. *What complicates the picture is that this propensity for errors will, intrinsically, exist at each gate.* Hence, in future, reliable computation has to be achieved with "systemic" unreliable devices [13]. Thus making the entire computation process probabilistic rather than deterministic in nature. For instance, given inputs 1 and 0, an AND gate will output the state 0, only with probability $1 - p$, where p is the gate error probability. Thus, traditional, deterministic, truth-table based logic representation will not suffice. Instead, the output needs to be specified in terms of probabilities, conditioned on the states of the inputs. Table I shows such a specification for (a) a 2-input error-free AND gate used in current CMOS designs and (b) 2-input AND gate with dynamic errors for next-generation technologies.

There will be need for formalisms to compare, evaluate, and vet circuit designs with these dynamic error prone gates. Note that *these nano-domain dynamic errors are unlike the traditional permanent hard faults in CMOS circuits due to run-time device failure or manufacturing defects, nor is it similar to the conventional soft errors that can arise due to radiation and device noise, which are localized, purely random, and can be reduced by external means*.

TABLE I

PROBABILISTIC REPRESENTATION OF THE "TRUTH-TABLE" OF A TWO INPUT, ERROR-FREE, AND GATE AND FOR AN AND GATE WITH DYNAMIC ERRORS.

Ideal AND gate			
X_{i1}	X_{i2}	$P(X_o \mid X_{i1}, X_{i2})$	
		$X_o = 0$	$X_o = 1$
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

AND gate with dynamic error			
X_{i1}	X_{i2}	$P(X_o \mid X_{i1}, X_{i2})$	
		$X_o = 0$	$X_o = 1$
0	0	1-p	p
0	1	1-p	p
1	0	1-p	p
1	1	p	1-p

The **sources of dynamic errors** can be different for various emerging technologies. The error probability p (see Table I) will be dependent on the technology. For instance,

1. In nano-CMOS, dynamic error will arise due to the use of ultra low voltage design, resulting in supply to ground bounce, leakage and coupling noise and due to small device geometry operating with only a handful of electrons.
2. In carbon nano-tube(CNT) [12], [9] and resonant tunnel diodes(RTD) [18], dynamic errors will arise due to their operating conditions near thermal limit. In fact, predictions also suggest that increased clock speed and increased computational requirements would make switching transition energy limits to a few order of magnitude of kT [5], where T is the temperature in Kelvin and k is the Boltzmann's constant.
3. In quantum-dot cellular automata (QCA) based circuits, errors can be classified as static (decay) errors, switching (thermal) errors, dynamic errors, and errors due to background charge fluctuations [16]. Decay errors occur when information stored in a latch is lost before the end of a clock cycle. This can happen when an electron locked at the top dot, tunnels out of it into the bottom dot or vice versa, when the latch is in its locked state. Switching errors occur when a gate switches into the wrong logic state when clock is applied, because of external thermal excitation. Dynamic errors occur when clock frequency approaches the electron tunneling rate in the device. In addition to dynamic errors, perma-

nent hard faults can occur in QCA designs due to cell displacement, cell misalignment and cell omission [17].

Error probability, p of a gate depends on many factors such as switching at that node, leakage, temperature, V_{th}, etc. For example gates with high switching may produce wrong signals when they operate near their thermal limits due to increased power dissipation. A device under dynamic error at any time instant may operate correctly after a short period. Thus probability of dynamic error in individual devices needs extensive characterization at quantum level and it has to be obtained from some industrial data. Given the p values of the gates, our model accurately estimates the overall output error probabilities. In this work, we assume that all gates have the same dynamic error probability, p, to compare our work with the state-of-the-art. However, it could be easily extended to a more device friendly diverse probabilistic model where each gate would have unique error probability.

In this paper, we model the effect of dynamic errors in nano devices. We estimate the overall error probability at the output of a logic block where individual logic elements are subject to error with a finite probability, p. We construct the overall BN representation based on logic level specifications by coupling gate level representations. Each gate is modeled using a conditional probability table (CPT) which models the probability of gate output signal being at a logic state, given its input signal states. An ideal logic gate with no dynamic error has its CPT derived from the gate truth table, whereas the CPT of a gate with error is derived from its truth table and the error probabilities, which can be input dependent. The overall joint probability model is constructed by networking these individual gate CPTs. This model captures all signal dependencies in the circuit and is a minimal representation, which is important from a scalability perspective.

We measure the error with respect to the ideal logical representation. Suppose, we have logic block with inputs $Z_1, \cdots Z_N$, internal signals $X_1, \cdots X_M$, and outputs $Y_1, \cdots Y_K$. Let the corresponding

versions of internal signals and outputs with dynamic errors be denoted by $X_1^e, \cdots X_M^e$ and $Y_1^e, \cdots Y_K^e$, respectively. Thus, the error at the ith output,E_i can be mathematically represented as the XOR of the error-free output and the error-encoded output(We discuss this in detail in Section IV).

$$E_i = Y_i^e \oplus Y_i \tag{1}$$

We propose the output error probability $P(E_i = 1) = P(Y_i^e \bigoplus Y_i = 1)$ as a design metric, in addition to other traditional ones such as area or delay, to vet different designs in the nano-domain. Note that the probability of output error is dependent on the individual gate error probability p and also on the internal dependencies among the signals that might enhance or reduce the overall output error based on the circuit structure. In this work,*we prove that for causal logical circuits, these dependencies can be modeled by a Bayesian Network, which is known to be the exact, minimal probabilistic model for the underlying joint probability density function (pdf)*. Probabilistic belief propagation on these Bayesian Networks can then be used to estimate this probability.

Bayesian Networks are causal graphical probabilistic models representing the joint probability function over a set of random variables. A Bayesian Network is a directed acyclic graphical structure (DAG), in which nodes describe random variables and node to node arcs denote direct causal dependencies. A directed link captures the direct cause and effect relationship between two random variables. Each node is quantified by the conditional probability of the states of that node *given* the states of its parents, or its direct causes. The attractive feature of this graphical representation of the joint probability distribution is that not only does it make conditional dependency relationships among the nodes explicit but it also serves as a computational mechanism for efficient probabilistic updating. Bayesian networks have traditionally been used in medical diagnosis, artificial intelligence, image analysis, and specifically in switching model [19] and single stuck-at-fault/error model [21] in VLSI but their use in dynamic error modeling is new. We first explore an exact inference scheme also known as clustering technique [26], where the original DAG is

transformed into special tree of cliques such that the total message passing between cliques will update the overall probability of the system. We then explore two stochastic inference schemes, named Probabilistic Logic Sampling (PLS) [24], and Evidence Pre-Propagated Importance Sampling (EPIS) [22], [23]. In PLS, a full instantiation of the probabilistic network is collected based on a simplified importance function. The sampling is stopped when the probabilities of the nodes converge. *It is worth pointing out that unlike simulative approaches that sample the inputs, importance sampling based procedures generate instantiations for the whole network, not just for the inputs.* These samples can be looked upon as Markov Chain sampling of the circuit state space. However, PLS is less efficient for diagnostic reasoning since some of the generated samples which do not match with the given evidence set have to be discarded. Hence for problems involving diagnostic reasoning (inverse problem) we use a more efficient and accurate sampling algorithm, namely EPIS which takes into account any available evidence. We discuss these algorithms in detail in Section V.

Contributions of this paper are summarized as follows:

1. We propose an exact probabilistic representation of the error model that captures the circuit topology for estimating the overall exact output error probabilities for small circuits
2. We use scalable pattern insensitive stochastic inference schemes for estimation of approximate output error probabilities of medium sized benchmarks (ISCAS'85). These inference schemes give excellent accuracy-time trade-off.
3. We compute sensitivities of individual gate errors to the overall output error of a circuit, which is useful for selective redundancy application.
4. We use the estimated overall output error probabilities as a design metrics for comparing equivalent designs.
5. We characterize the input space for achieving a desired output behavior in terms of error

tolerance by utilizing the unique backtracking feature of Bayesian Networks.

6. We apply selective redundancy techniques for reliability enhancement and evaluate reliability/redundancy trade-off.

II. Prior Work

The issue of reliable computing using unreliable devices is not new. Indeed the basic methods of Triple Modular Redundancy (TMR) and NAND Multiplexing were proposed in the 1950s [13], [14]. These ideas have been adapted for nano-computing architectures such as N-tuple Modular Redundancy [1], or NAND Multiplexing and reconfiguration [2]. A recent comparative study of these methods [3], indicates that a 1000-fold redundancy would be required for a device error (or failure) rate of 0.01[1]. Probabilistic model checking for reliability-redundancy trade-off was introduced in [6]. These techniques of providing error and fault tolerance methods, rarely utilized the topology of a circuit and dependency of errors on the circuit topology.

The analysis technique based on elementary bifurcation theory proposed in [8] for computing the exact error threshold values for noisy gates is applicable for building future fault-tolerant nano-domain networks. H. J. Gao *et. al* in [7] gives a comparison of different redundancy schemes such as Von Neumann's multiplexing logic, N-tuple modular redundancy and interwoven redundancy in terms of degree of redundancy and reliability by using markov chain models and bifurcation analysis. Reliability of nand-multiplexed nanoelectronic systems is characterized using markov chains and probabilistic computation schemes in [10].

In a recent work, S. Krishnaswamy *et al.* [4] provided a probabilistic transfer matrix-based (PTM) formalism for modeling dynamic errors at logic level. Computational complexity of this method is very high. Han *et al.* in [11], discuss an approximate method based on Probabilistic Gate Model (PGM) for reliability evaluation with less time and space complexity, compared to

[1] Note that this does *not* mean 1 out of 100 devices will fail, it indicates the devices will generate erroneous output 1 out of 100 times.

PTM method. Also, in [5] Chen *et al.* proposed Markov Random Field (MRF) based model which relates thermal energy and entropy. However, results for MRF based models were shown only for circuits that are extremely small and did not have any re-convergence.

The model proposed in this paper is the first step towards understanding the errors specific to a circuit. We know that error probability inference is NP-hard, however, using conditional independence and smart approximation, reasonable estimates can be found. In this work, we present an exact algorithm for error probability computation of small circuits and an approximate algorithm for the mid-sized ISCAS benchmarks. Note that a simulative approach is not selected as it is not only driven by input pattern dependence but also separate simulation and input trace generation schemes have to be adopted for biased inputs.

Rationale for using Bayesian Model:

We propose to use Bayesian modeling and inferencing for the following reasons

- Conventional logic computing is inherently causal. For example outputs of a gate is influenced by the inputs where as inputs are not changed by the outputs. This paradigm of conventional computing translates into a causal probabilistic model where the causality forces the probabilistic model to a **directed graph structure**.
- Nano-CMOS, CNT, RTD and Clocked QCA do exhibit causal flow in processing information.
- Certain dependencies observed in causal networks, induced and non-transitive dependencies, are captured most effectively by Bayesian Networks [29] and makes it a superior model for causal networks over MRF and other non graph-based algorithms.

  MRF are undirected graphs and Bayesian Networks are directed graphs. This directional feature adds significant impact in modeling conditional independencies, also called I-map [19] which are the underlying rationale to arrive at a minimal (sparse) structure. The d-separation

or directional separation (in a DAG) is directly coupled with I-maps in a causal system which let us model in an edge-minimal fashion [30], [29]. If the information flow through an edge is not known, then independencies would not be captured properly [19].

Fig. 11 [29] illustrates a few important aspects of dependency models and underlying graph structures. First, not all dependency models can be embedded in a graph. Also, it can be observed that dependencies that a BN can model might not be captured in MRF and vice versa [29], [30]. In logical scenario where inputs drive output, i.e. causal setup, BN is the natural choice and is the best model encapsulating all the dependencies retaining all the independencies (related to sparsity or minimality) [29], [30].

For an example, suppose we have 2-input AND gate. If the output is "0", the inputs becomes dependent (if one of the input is "1", the other is forced to be a "0" to maintain the evidence at output node "0"). Even though two inputs can be independent, the observation of the child imposes dependence in the parents. This is true in logic and causal networks. Since d-separation in a DAG models this dependence, we do not need an additional link between parents where as in an undirected environment, additional links between parents becomes necessary and hence the graph is not as sparse as it should be.

MRF is not an undirected Bayesian network. Bayesian networks are inherently directed, due to the use of conditional probabilities. If one removes the directions in a BN one does not get an MRF. One has to go through a series of reasoned transformations to preserve all the dependences and most of the independencies.

One can see that decomposable graphs (chordal graphs) can embed dependencies expressed by both directed and undirected graph and is the intersection of BN and MRF model. Every chordal graph can possess at least one junction tree. One might see that if we add all nodes together, it is a junction tree of one clique containing all nodes. As complexity of

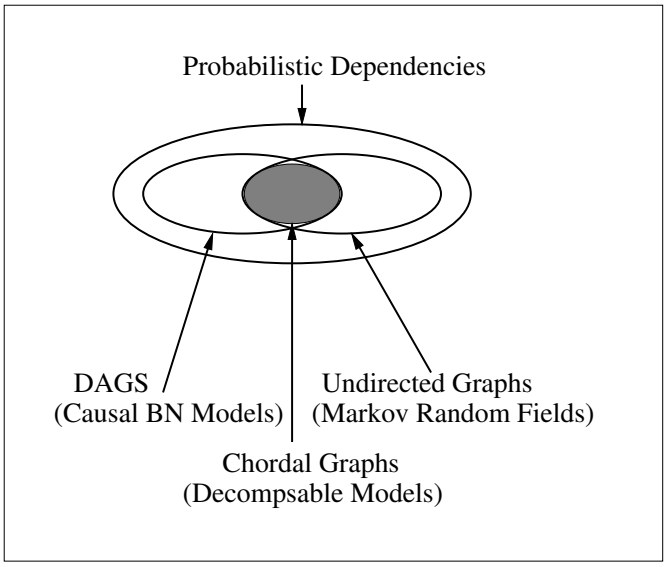


Fig. 1. Probabilistic models and graph structure. [29]

inference is exponential to the nodes in the largest clique of the junction tree, the key is to find a junction tree that has the smallest number of nodes in its largest clique (again drives us to the I-map and sparsity) and then we will have the computational leverage over marginalizing the numerical joint pdf. For modeling a causal network, junction tree cliques can be largely minimized when arrived through parent-child relationship as opposed to an MRF. Superiority of BN over MRF model is not determined by arriving at a junction tree but by arriving at a smallest junction tree.

- As we know all these probabilistic models are NP-hard, average case complexity of Bayesian Network is the least because it not only models the conditional independencies but exploits it to its advantage in probabilistic inference. More interestingly, the approximate BN models have found faster and easier convergence and has potential to infer for realistic network sizes.
- Last but not the least Bayesian Network are uniquely capable to solve the inverse problem and are capable of answering questions like “What are probabilistic input profile that guarantees zero error at the output?”. This makes it the most dominant probabilistic tool for

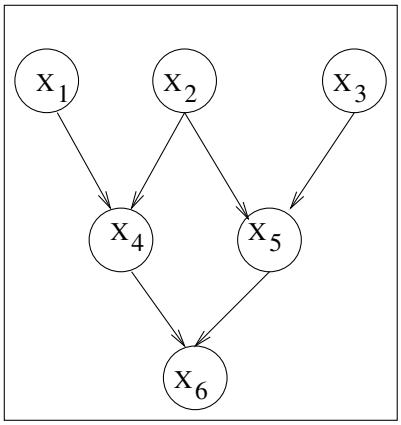


Fig. 2. A small Bayesian Network

diagnostic analysis.

III. Bayesian Networks

Bayesian Networks are graphical probabilistic models representing the joint probability function over a set of random variables using a directed acyclic graphical structure (DAG), whose nodes describe random variables and node to node arcs denote direct causal dependencies. Fig. 2 shows a small Bayesian network. The exact joint probability distribution over the variables $X_1 \cdots, X_6$ in this network is given by Eq. 2.

$$\begin{aligned} P(x_6,x_5,x_4,x_3,x_2,x_1) &= P(x_6|x_5,\cdots,x_1)P(x_5|x_4,\cdots,x_1) \\ & \quad P(x_4|x_3,x_2,x_1)P(x_3)P(x_2)P(x_1) \end{aligned} \tag{2}$$

In this BN, the random variable, X_6 is independent of X_1, X_2 and X_3 given the states of its parent nodes, X_4 and X_5. This *conditional independence* can be expressed by Eq. 3.

$$P(x_6|x_5,x_4,x_3,x_2,x_1) = P(x_6|x_5,x_4)) \tag{3}$$

Mathematically, this is denoted as $I(X_6,\{X_4,X_5\},\{X_1,X_2,X_3\})$.

If causality is used to choose the link directions, it can be shown [29], by invoking the properties of Markov blanket and d-separation that the factorized form, thus obtained, is minimal in the

number of links, or representation size, necessary to encode the joint probability function. The optimal factorized form will involve conditional probabilities based on the parents (or direct causes) to a node: $P(X) = \prod_{k=1}^{m} P(x_k|pa(x_k))$.

The attractive feature of this graphical representation of the joint probability distribution is that not only does it make conditional dependency relationships among the nodes explicit but it also serve as a computational mechanism for efficient probabilistic updating. Bayesian networks have traditionally been used in artificial intelligence, image analysis, and in specifically in switching model and timing analysis [19], [20] in VLSI but their use in nano-domain dynamic-error modeling is new. In the absence of real nano-domain benchmark, we show estimates on ISCAS'85 benchmark circuits in low time overhead.

IV. Probabilistic Error Model

We compute the error probability $P(e_i) = P(Y_i^e \bigoplus Y_i = 1)$ by marginalizing the joint probability function over the inputs, internal lines, and the outputs[2].

$$P(e_i) = \sum_{z_1,\cdots z_N} P(e_i|z_1,\cdots z_N)P(z_1)\cdots P(z_N) \tag{4}$$

$$= P(z_1)\cdots P(z_N)\sum_{z}\sum_{x,x^e} P(e_i|z_1,\cdots z_N) \tag{5}$$

$$= P(z_1)\cdots P(z_N)$$

$$\sum_{\forall z}\sum_{\forall x,\forall x^e} P(e_i,y_i,y_i^e,z_1,\cdots z_N,x_1,\cdots x_M,x_1^e,\cdots x_M^e) \tag{6}$$

where Eq. 6 shows that the joint density function that is necessary to compute the dynamic error exactly. Summing over all possible values of all the involved variables is computationally expensive (NP-hard), hence we require a graphical model that would use the causal structure and conditional independences to arrive at the minimal optimally factorized representation of this joint probability function as a Bayesian network.

[2] In this paper, $P(x)$ denotes the probability of the event $X = x$, i.e. P(X=x).

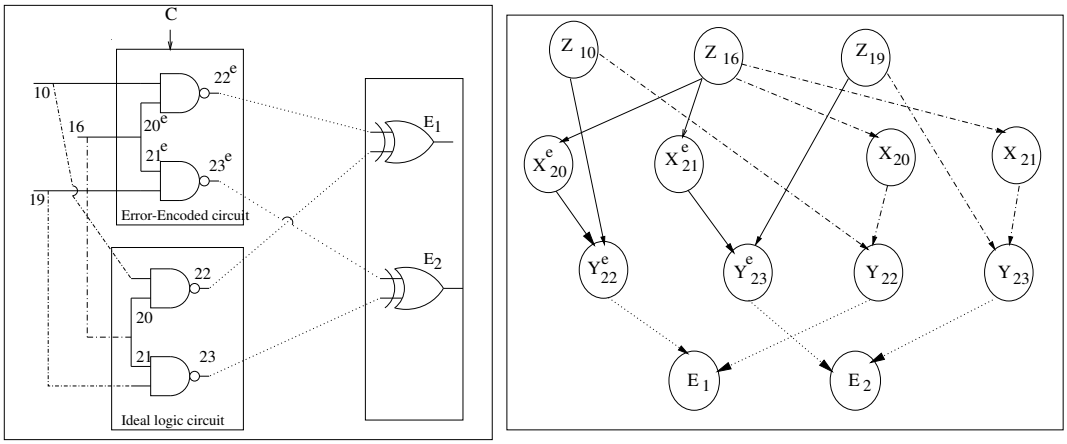


Fig. 3. (a) Conceptual circuit representation of the logic used to detect errors involving the error-free logic and the unreliable logic components. (b) The corresponding Bayesian Network representation.

A. The Bayesian Network Structure

We model, both the error free logic and the logic with dynamic errors, as a Directed Acyclic Graph (DAG). These two models, which we will refer to as the ideal logic model and the error-encoded model, are then connected at the outputs by comparators. The comparator output is the variable $E_i = Y_i^e \oplus Y_i$ in Eq. 1. The comparator output of logic 1 indicates that the ideal logic model and error-encoded model outputs are different. The probability of the comparator outputs being in state *"1"* provides the overall circuit error probability, $P(E_i = 1)$.

Figure 3(a) shows the (conceptual) representation of a error detection circuit for a simple logic involving two NAND gates. Block C represents the same logic, but built with unreliable components. These gates are assumed to have gate error probability of p. The inputs to both the blocks are the same. The two outputs are connected to two comparators. The output of the comparator represents error in computation. Note that this is just a conceptual representation, we do not actually propose synthesizing the circuit. Also the ideal logic circuit and the comparators are fictitious study elements and are not hardware components. From the conceptual circuit design, we can construct the Bayesian network representation, which we call the LIPEM model. Each

node in the LIPEM is a line in circuit and the links denote a connection between the lines via gates. Figure 3(b) shows the LIPEM corresponding to the circuit in Figure 3(a). In the rest of this section, we present a more formal definition and prove that the representation, thus obtained, is minimal.

Definition: The Logic Induced Probabilistic Error Model (LIPEM) corresponding to a combinational circuit $< C, \{p\} >$, where C is the circuit and $\{p\}$ are individual gate error probabilities, can be constructed as follows. Nodes are random variables with two values: 0 or 1. There are 6 types of nodes.

- $\{Z\}$: Primary inputs
- $\{X^e\}$: Internal signals with error probability p.
- $\{X\}$: Internal signals under ideal logical condition.
- $\{Y^e\}$: Erroneous output signals.
- $\{Y\}$: Primary output under ideal logical condition.
- $\{E\}$: Error nodes representing the error at each output.

Edges of LIPEM are directed and denoted by the ordered pair $(u \rightarrow v)$ where u causes a v. The edge set can be classified as follows:

- $(Z \rightarrow X^e)$: Edges between nodes representing primary inputs and outputs of erroneous gates that are directly fed by the primary inputs. $(Z \rightarrow X)$: Edges between nodes representing the same primary inputs and outputs of corresponding ideal logic gates.
- $(X_i^e \rightarrow X_j^e)$, $(X_i \rightarrow X_j)$: Edges between random variables representing internal signals (Edges from the input of a gate to the corresponding output). If there is an edge $(X_i \rightarrow X_j)$, then there must be a mirror edge $(X_i^e \rightarrow X_j^e)$. These two edges differ in the conditional probability discussed later during the quantification of LIPEM.
- Edges $(X \rightarrow Y)$ and $(X^e \rightarrow Y^e)$
- Edges $(Y \rightarrow E)$ and corresponding $(Y^e \rightarrow E)$

Then LIPEM structure thus formed is proved to be a Bayesian Network.

Theorem: The LIPEM structure, corresponding to the combinational circuit C is a minimal I-map of the underlying dependency model and hence is a Bayesian network.

Proof: Markov boundary of a random variable v in a probabilistic framework, is the minimal set of variables that make the variable v conditionally independent of all the remaining variables in the probabilistic network.

Let us order the random variables in the node set, such that for every edge $(u,\ v)$ in LIPEM, u appears before v. With respect to this ordering, the Markov boundary of any node, $v \in \{\{Z\}, X, Y, \{\{X^e\}, \{Y^e\}, \{E\}\}$ is given as follows. If v represents a primary input signal line, then its Markov boundary is the null set. And, since the logic value of an output line is just dependent on the inputs of the corresponding gate (whether v is in $\{\{X\}$, or $\{X^e\}$, or $\{\{Y\}$, $\{Y^e\}$, or $\{\{E\})$ the Markov boundary of a variable representing an output line consists of just those variables that represent the inputs to that gate. Thus in LIPEM structure the parents of each node are its Markov boundary elements. Hence the LIPEM is a boundary DAG. Note that LIPEM is a boundary DAG because of the causal relationship between the inputs and the outputs of a gate that is induced by logic. It has been proven in [29] that if graph structure is a boundary DAG D of a dependency model M, then D is a minimal I-map of M ([29]). This theorem along with definitions of conditional independencies, in [29] (we omit the details) specifies the structure of the Bayesian network. Thus LIPEM is a minimal I-map and thus a Bayesian network (BN).

B. Bayesian Network Quantification

LIPEM thus constructed, consists of nodes that are random variable of the underlying probabilistic model and edges denote direct dependencies. All the edges are quantified with the corresponding conditional probabilities of the form $p(x_{v_i}|parent(x_{v_i}))$, where $parent(x_{v_i})$ is the set of nodes that has directed edges to x_{v_i}. These conditional probability specifications are determined by the gate type. A complete specification of the conditional probability of a two input AND gate

output will have 2^3 entries since each variable has 2 states. The edges in the error-encoded part would be quantified by logic function allowing for some randomness. The conditional probability of an error-free AND gate and an unreliable AND gate with gate error probability p are shown in Table I(a) and (b), respectively.

V. Computing the Error Probability

We explore three inference schemes for the LIPEM, an exact inference scheme based on clustering and two stochastic inference schemes based on importance sampling. The exact inference scheme is suitable for small circuits. The stochastic sampling algorithms are scalable, pattern insensitive, anytime method with excellent accuracy-time trade-off.

A. Exact Inference

We demonstrate this inference scheme with an example shown in Fig 4.The combinational circuit is shown in Fig. 4(a) and Fig 4(b) is its equivalent Bayesian Network representation. The first step of the exact inference process is to create an undirected graph structure called the *moral graph* (denoted by D^m) given the Bayesian network DAG structure (denoted here by D). The moral graph represents the Markov structure of the underlying joint function [28]. The dependencies that are preserved in the original DAG are also preserved in the moral graph [28]. From a DAG, which is the structure of a Bayesian network, a moral graph is obtained by adding undirected edges between the parents of a common child node and dropping the directions of the links. Fig. 4c shows the undirected moral graph and the dashed edges are added at this stage. This step ensures that every parent child set is a complete sub graph. Moral graph is undirected and due to the added links, some of the independencies displayed in DAG will not be graphically visible in moral graph. Some of the independencies that are lost in the transformation contributes to the increased computational efficiency but does not affect the accuracy [28]. The independencies that are graphically seen in

the moral graph are used in inference process to ensure local message passing.

The moral graph is said to be triangulated if it is chordal. The undirected graph G is called chordal or triangulated if every one of its cycles of length greater than or equal to 4 possesses a chord [28] that is we add additional links to the moral graph, so that cycles longer than 3 nodes are broken into cycles of three nodes. Note that in this particular example, moral graph is chordal and no additional links are needed. The junction tree is defined as a tree with nodes representing cliques (collection of completely connected nodes) of the choral graph and between two cliques in the tree T there is a unique path. Fig 4d shows the junction tree. Note that every clique in the moral graph is a node (example $C1 = [Z1, Z2, Y1e, Y1)$ in the junction tree. Junction tree possesses a property called running intersection that ensures that if two cliques share a common variable, the variable should be present in all the cliques that lie in the unique path between them. This property of junction tree is utilized for probabilistic inference so that local operation between neighboring cliques guarantees global probabilistic consistency.

Chordal graphs are essential as they guarantee the existence of at least one junction tree. Hence chordalization is a necessary step. There are many algorithms to obtain junction tree from chordal graph and we use a tool HUGIN [26] that uses minimum-fill-in heuristics to obtain a minimal chordal and junction tree structure.

Since every child parent team is present together in one of the cliques, we initialize the clique joint probabilities by the original joint probability of a child parent team. We then use a message passing scheme to have consistent probabilities. Suppose we have two leaf cliques in the junction tree , C3 and C4 as in our example in Fig 4d. Both the cliques are initialized based on the child parent team (C3 by nodes Y1,Y1e and E1 and C4 by node Y2,Y2e, E2). The initial clique probability of clique Ci is termed as ϕ_{Ci} and is also called potential of a clique.

Let us now consider two neighboring cliques to understand the key feature of the Bayesian

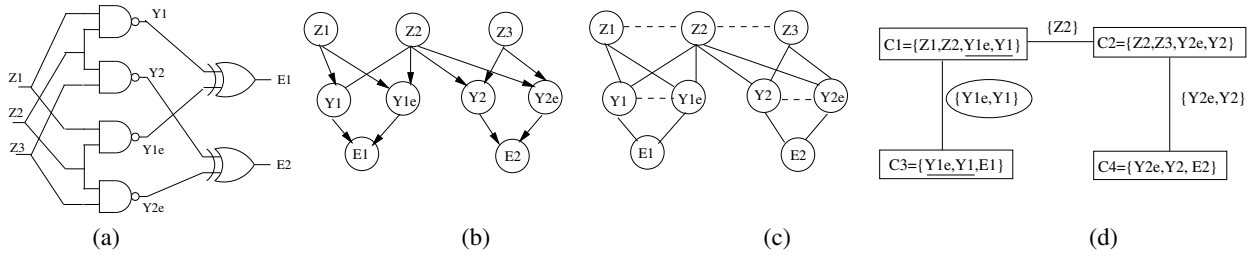


Fig. 4. (a) A small error model (b) Bayesian Network representation (c) Chordal graph (d) Junction tree

updating scheme. Let two cliques Cl and Cm have probability potentials ϕ_{Cl} and ϕ_{Cm}, respectively. Let S be the set of nodes that separates cliques A and B (Example: $S = \{Y1e, Y1\}$ between cliques C1 and C3 in Fig. 4d). The two neighboring cliques have to agree on probabilities on the node set S which is their separator. To achieve this we first compute the marginal probability of S from probability potential of clique Cl and then use that to scale the probability potential of Cm. The transmission of this scaling factor, which needed in updating, is referred to as message passing. New evidence is absorbed into the network by passing such local messages. The pattern of the message is such that the process is multi-threadable and partially parallelizable. Because the junction tree has no cycles, messages along each branch can be treated independently of the others.

Note that since junction tree has no cycle and it is also not directional, we can propagate evidence from any node at any clique in any direction. It is in sharp contrast with input pattern sensitive simulative approaches (like HSpice) where the flow of information always propagate from input to output. Thus, we would be able to use it for input space characterization for achieving zero output error due to dynamic errors. We would instantiate a desired observation in an output node (say zero error) and backtrack the inputs that can create such a situation. If the actual input trace has large distance from the characterized input space, we can conclude that zero error is reasonably unlikely. Note that this aspect of probabilistic modeling is already used in medical diagnosis but are new in the context of input space modeling for dynamic error.

Complexity Analysis

The computational complexity of the exact method is exponential in terms of number of variables in the largest cliques. Space complexity of the exact inference is $n.2^{|Cmax|}$ [19], where n is the number of nodes in the Bayesian Network, and $|Cmax|$ is the number of variables in the largest clique. The time complexity is given by $p.2^{|Cmax|}$ [19] where p is the number of cliques.

Since exact inference in expensive in terms of time and hence for larger circuits, we explore two stochastic sampling algorithms, namely probabilistic Logic Sampling (PLS) and Evidence Pre-propagated Importance Sampling (EPIS). These algorithms have been proven to converge to the correct probability estimates [24], [23], without the added baggage of high space complexity.

B. Approximate Inference

The stochastic sampling algorithms are based on a sampling scheme known as importance sampling. This algorithm generates randomly selected instantiations of the network variables in topological order, according to probabilities in the model, and then calculates frequencies of instantiations of interest to obtain estimates of the probabilities. An elaborate discussion on this is given in [22].

Given the states of certain nodes in the network (evidence) and the conditional probability table of each node, this algorithm generates sample instantiations of the network so as to generate state of each node in the network. From these sample instantiations, it can find approximate probabilities of each state for each node in the network. This sampling is done according to the optimum importance function $g(x)$, that closely resembles the underlying joint probability distribution $P(x)$ for which the variance of $\hat{I}_N$ is *zero*, where $\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N}\frac{P(s_i)}{g(s_i)}$ for N samples $s_i \forall i = 1,\cdots,N$ obtained by sampling the importance function $g(x)$ [22]. It has been proven that in a Bayesian network, the product of the conditional probability functions at all nodes form the optimal importance function [22]. Let $X = \{X_1, X_2, \cdots, X_m\}$ be the set of variables in a Bayesian network, $Pa(X_k)$ be the parents of X_k, and E be the evidence set. Then, the optimal importance function is given by

$P(X|E) = \prod_{k=1}^{m} P(x_k|Pa(x_k,E)$.

Probabilistic Logic Sampling (PLS): Probabilistic logic sampling is the first and the simplest sampling algorithms proposed for Bayesian Networks [24]. The salient features of these algorithms are: (1) they scale extremely well for larger systems making them a target inference for nano-domain billion transistor scenario and (2) they are any-time algorithm, providing adequate accuracy-time trade-off and (3) The samples are not based on inputs and the approach is input pattern insensitive. The flow of the algorithm is as follows.

1. Complete set of samples are generated for the Bayesian network using the optimal importance function, which is the joint probability distribution function $P(X)$. The importance function is never updated once it is initialized. This assumption makes sense when the child node evidences are not present.
2. In case of child node evidences (important in diagnostic backtracking), samples that are incompatible with the evidence set are disregarded.
3. The probability of all the query nodes are estimated based on counting the frequency with which the relevant events occur in the sample.

In predictive inference, logic sampling generates precise values for all the query nodes based on their frequency of occurrence but with diagnostic reasoning, this scheme fails to provide accurate estimates because of large variance between the optimal importance function and the actual underlying joint probability distribution function. The above scheme is not efficient for diagnostic reasoning due to the need to generate, but disregard samples that do not satisfy the given evidence. It would be more efficient not to generate samples which are not matching with the evidence set. We discuss such a method next.

Evidence Pre-propagated Importance Sampling (EPIS):

The evidence pre-propagated importance sampling (EPIS) [22], [23] uses local message pass-

ing and stochastic sampling. This method scales well with circuit size and is proven to converge to correct estimates. This is also an anytime-algorithm since it can be stopped at any point of time to produce estimates. Of course, the accuracy of estimates increases with time.

Like PLS, EPIS is also based on importance sampling that generates sample instantiations of the *whole* DAG network, i.e. for all line states in our case. These samples are then used to form the final estimates. The difference is with respect to the importance function used for sampling. EPIS takes into account any available evidence. In a Bayesian network, the product of the conditional probability functions at all nodes form the optimal importance function. Let $X = \{X_1, X_2, \cdots, X_m\}$ be the set of variables in a Bayesian network, $Pa(X_k)$ be the parents of X_k, and E be the evidence set. Then, the optimal importance function is given by

$$P(X|E) = \prod_{k=1}^{m} P(x_k|Pa(x_k, E)) \tag{7}$$

This importance function can be approximated as

$$P(X|E) = \prod_{k=1}^{m} \alpha(Pa(X_k))P(x_k|Pa(X_k))\lambda(X_k) \tag{8}$$

where $\alpha(Pa(X_k)) = (P(E^-|Pa(X_k)))^{-1}$ is a normalizing constant dependent on $Pa(X_k)$ and $\lambda(X_k) = P(E^-|x_k)$, with E^+ and E^- being the evidence from parents and children, respectively, as defined by the directed link structure. Calculation of λ is computationally expensive and for this, Loopy Belief Propagation (LBP) [25] over the Markov blanket of the node is used. The importance function can be further approximated by replacing small probabilities with a specific cutoff value [22].

This EPIS stochastic sampling strategy works because in a Bayesian Network the product of the conditional probability functions for all nodes is the optimal importance function. Because of this optimality, the demand on samples is low. We have found that just thousand samples are sufficient to arrive at good estimates for the ISCAS'85 benchmark circuits. The ability of EPIS to handle diagnostic *and* predictive inference comes at the cost of a somewhat increased time per iterations needed for the calculation of λ messages. We quantify this increase by our experiments.

Time and Space Complexity:

The space requirement of the Bayesian network representation is determined by the space required to store the conditional probability tables at each node. For a node with n_p parents, the size of the table is 2^{n_p+1}. The number of parents of a node is equal to the fan-in at the node. In our model we break all fan-ins greater than 2 into a logically equivalent, hierarchical structure of 2 fan-in gates. For nodes with fan-in f, we would need $\lceil f/2 \rceil$ extra Bayesian network nodes, each requiring only 2^3 sized conditional probability table. For the worst case, let all nodes in the original circuit have the maximum fan-in, f_{max}. Then the total number of nodes in the LIPEM structure for each fault model is $n + f_{max}n/2$. Thus, the worst case space requirement is linear, i.e. $O(nf_{max})$ where n is the number of nodes and f_{max} is the maximum fan-in.

The time complexity, based on the stochastic inference scheme, is also linear in n, specifically, it is $O(nN)$, where N is the number of samples.

VI. Experimental Results

We demonstrate the experimental results using LGSynth'93 and ISCAS'85 benchmark circuits. Even though these benchmarks are for nano-CMOS, logical structures namely re-convergence and input-output behaviors are fundamental to most nano devices. Also, we do not have medium size standard designs to show the efficiency of our model for most of the emerging devices. Hence to study the scalability and time-accuracy trade-off, we choose these benchmark circuits that are used in many previous researches and are well understood.

We use exact inference scheme to estimate output error probabilities of small circuits from LGSynth '93 benchmarks and show that the our model is orders of magnitude less in comparison with PTM based model [4] in terms of space and time complexity. The approximate computation of the LIPEM using EPIS and PLS algorithms was performed by a tool named "SMILE" which is the API version of a tool named "GeNIe" [27]. The exact computation of the LIPEM using

TABLE II
AVERAGE OUTPUT ERROR PROBABILITIES [FROM EXACT INFERENCE]

	Nodes in LIPEM	Average output error probability for individual gate error probability p			time(s)
		=0.005	=0.05	=0.1	
c17	19	0.0148	0.1342	0.2398	0.0
parity	47	0.0699	0.3971	0.4824	0.0001
pcle	116	0.0179	0.1560	0.2702	0.07
decod	129	0.0068	0.0654	0.1251	0.14
cu	210	0.0232	0.1969	0.3327	0.56
pm1	225	0.0331	0.2627	0.4141	0.44
xor5	118	0.0925	0.4336	0.4900	0.26
alu4	222	0.0676	0.3906	0.4816	1.87
b9	392	0.0315	0.2475	0.3867	2.49
comp	415	0.0733	0.3828	0.4683	0.66
count	404	0.0203	0.1613	0.2632	1.14
malu4	280	0.0845	0.4253	0.4903	1.94
max_flat	70	0.0296	0.2151	0.3234	0.02
pc	261	0.0377	0.2794	0.4161	0.41
voter	134	0.0299	0.2178	0.3294	0.08

Clustering algorithm was performed using a tool named "HUGIN" [26]. It has to be noted that even though GeNIe/SMILE provides the junction tree algorithm, HUGIN forms the more minimal and optimal junction tree. That's the reason for using HUGIN for exact inference. The tests were performed on a Pentium IV, 2.00GHz, Windows XP computer.

Small Circuits: In table II, we report the average output error probabilities of benchmark circuits for different gate error probabilities. Column 2, 3 and 4 give the average output error probabilities when gate error probabilities are 0.005, 0.05 and 0.1 respectively. In column 5 we report the elapsed time for the error estimation. From this table, we can see that the decoder circuit has the lowest output error probability and circuits like, xor5, malu4 and parity have the worst error characteristics. For the XOR circuit, even a gate error probability of $p = 0.05$ is not acceptable, because with this gate error probability, the output error probability is 0.4336, which is the highest over all circuits.

The time complexity of PTM based model is given as $O(2^{n+m})$ where n is the number of inputs of the circuit under study and m is the number of outputs, whereas the time complexity of

TABLE III
COMPARISON OF THE RELIABILITY ESTIMATES OBTAINED FROM PTM MODEL AND LIPEM MODEL FOR GATE ERROR PROBABILITY $p = 0.05$

Circuit	Circuit reliability for individual gate error probability $p = 0.05$	
	LIPEM model	PTM model
c17	0.866	0.866
parity	0.603	0.603
pcle	0.844	0.844
decod	0.935	0.935
cu	0.803	0.803
pm1	0.737	0.737

TABLE IV
COMPARISON OF TIME COMPLEXITY BETWEEN PTM MODEL AND LIPEM MODEL

Circuit	Inputs n	Outputs m	$Cmax$	Time complexity	
				LIPEM model	PTM model
c17	5	2	6	$O(2^6)$	$O(2^7)$
parity	16	1	4	$O(2^4)$	$O(2^{17})$
pcle	19	9	7	$O(2^7)$	$O(2^{28})$
decod	5	16	13	$O(2^{13})$	$O(2^{21})$
cu	14	11	18	$O(2^{18})$	$O(2^{25})$
pm1	16	13	16	$O(2^{16})$	$O(2^{29})$

our model is given as $O(2^{|Cmax|})$ [19] as explained in Section V for exact inference scheme using clustering algorithm. Here $|Cmax|$ is the number of variables in the largest clique.

Table. IV shows the difference in time complexity between PTM model and LIPEM model for some of the smaller circuits where LIPEM models are inferenced using clustering algorithm. It can be clearly seen that the LIPEM model is much faster than the PTM model for all the circuits. Also the reliability estimates computed through PTM model and LIPEM model converge. Table. III gives the comparison of the reliability estimates obtained from PTM model and LIPEM model for gate error probability $p = 0.05$. These results are obtained by calculating the probability of correctness at the output. This can be given as 1 - *average output error probability*. Since both models exactly capture the circuit structure, the results produced are clearly similar to each other.

Han *et al.* [11] has given a comparison between their PGM model(Probabilistic Gate Model) and the PTM model. The PGM model, which is an approximate method for reliability modeling

in nano-circuits, gives reliability estimates very close to the results from the exact PTM model. Computational complexity analysis given in [11] shows that time and space complexity with PTM models is exponential with respect to the number of inputs *and* outputs, whereas complexity of PGM method is exponential *only* to the number of circuit inputs and it grows linearly with the number of outputs.

Scalability of the Error Model: We show that the error model and associated computations scales extremely well with circuit size by showing results with circuits of varying sizes. Table V shows the error probabilities for various ISCAS'85 benchmark circuits for three different gate errors of $p = 0.01, 0.001$ and 0.0001. Reported results were obtained from PLS with 10000 samples. As expected, all circuits exhibit higher overall error as individual gate error increases. It can be observed that in circuits like c432, c880 the increase in average output error probability is approximately 10 times. Also only two circuits(c499,c1355) show less than 10% error in the output for $p = 0.01$. This indicates that 0.01 is unacceptable gate error probability for most of the circuits. Additional gate level redundancy may be required for these circuits. In Table. VI we show the results with 1000, 10000 and 100000 samples for $p = 0.001$ and $p = 0.0001$. If we compare the estimates with 1000 samples and 10000 samples we can see that for some circuits like c499, c880, c1908, c7552 the estimates are close to the second decimal place. But other circuits like c432, c2670, c6288 are not close enough. If we compare the estimates with 10000 samples and 100000 samples, it can be clearly seen that for all circuits the estimates are equal to the second decimal place and close to the third decimal place. This shows that 10000 samples are adequate to generate the estimates.

To validate our model we performed an in-house logic simulation of the benchmarks with 500,000 random vectors obtained by changing seed after every 50,000 vectors and obtained the average output error probabilities for different gate error probabilities. Simulation is done by

TABLE V
AVERAGE OUTPUT ERROR PROBABILITIES FOR ISCAS'85 CIRCUITS -PLS WITH 10000 SAMPLES.

	No. of gates	Average output error probability for individual gate error probability p		
		=0.01	=0.001	=0.0001
c432	160	0.149	0.019	0.003
c499	202	0.034	0.005	0.001
c880	383	0.209	0.031	0.003
c1355	546	0.066	0.010	0.002
c1908	880	0.327	0.067	0.012
c2670	1193	0.380	0.075	0.011
c3540	1669	0.501	0.216	0.044
c5315	2307	0.379	0.071	0.011
c6288	2416	0.500	0.209	0.043
c7552	3512	0.497	0.142	0.027

TABLE VI
COMPARISON OF AVERAGE OUTPUT ERROR PROBABILITIES FOR ISCAS'85 CIRCUITS -PLS WITH 1000, 10000 AND 100000 SAMPLES.

Circuit	Average output error probability for individual gate error probability p					
	=0.001			=0.0001		
	1000 samples	10000 samples	100000 samples	1000 samples	10000 samples	100000 samples
c432	0.024	0.019	0.020	0.002	0.003	0.003
c499	0.007	0.005	0.004	0.003	0.001	0.001
c880	0.032	0.031	0.028	0.005	0.003	0.004
c1355	0.014	0.010	0.008	0.004	0.002	0.001
c1908	0.066	0.067	0.068	0.013	0.012	0.011
c2670	0.065	0.075	0.070	0.011	0.011	0.011
c3540	0.212	0.216	0.211	0.051	0.044	0.043
c5315	0.071	0.071	0.068	0.012	0.011	0.011
c6288	0.220	0.209	0.205	0.049	0.043	0.041
c7552	0.146	0.142	0.147	0.030	0.027	0.026

comparing the outputs from the ideal logic block with corresponding outputs from an erroneous logic block. Both ideal and erroneous logic blocks are fed by the same primary inputs. To simulate the effect of erroneous operation, we inject an additional input, I_f to each gate in the faulty logic blocks. If this input to a gate in the erroneous logic block holds logic value one, the gate is outputs a wrong signal. If I_f to a gate is logic zero, it behaves as an ideal gate. Probability of I_f being at logic one is the gate error probability p. Note that in a single simulation run, the input I_f to any gate in the erroneous logic block can be either zero or one, depending on the gate error probability p and the output of the random number generator which determines the state of this input. Fig. 5(a)

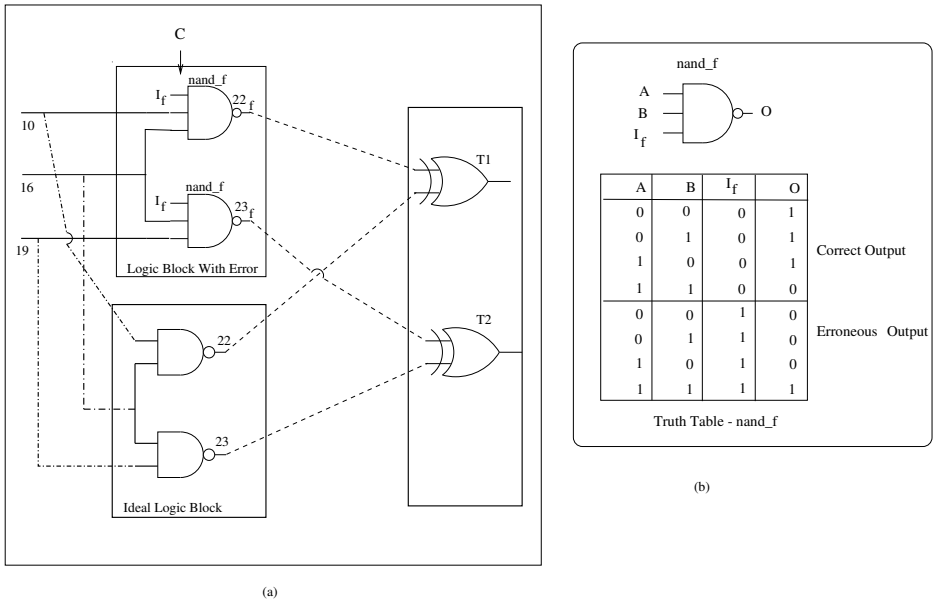


Fig. 5. (a)Dynamic Error simulation circuit (b) Truth-table of a NAND gate in the erroneous logic block

TABLE VII
COMPARISON OF BAYESIAN NETWORK MODELING AND LOGIC SIMULATION

	Nodes in LIPEM	1000 Samples		10000 samples	
		μ_e	T1(s)	μ_e	T2(s)
c432	475	0.0020	0.750	0.0020	7.453
c499	565	0.0010	0.875	0.0003	8.656
c880	956	0.0010	1.562	0.0004	15.250
c1355	1253	0.0020	2.369	0.0007	23.161
c1908	2172	0.0070	3.585	0.0060	34.617
c2670	3097	0.0104	4.831	0.0009	46.493
c3540	4038	0.0700	6.834	0.0700	65.515
c5315	6247	0.0077	10.875	0.0160	103.438
c6288	4896	0.0065	11.359	0.0022	107.234
c7552	8398	0.0650	14.843	0.0640	139.437

is a small dynamic error simulation circuit. Fig. 5(b) is the truth table of a NAND gate in the erroneous logic block of the simulation model.

In Table VII, we compare simulation results with Bayesian network results. We report the accuracy of our model in terms of average error, μ_e , between the exact output error probabilities and the estimated output error probabilities obtained from PLS with 1000 and 10,000 samples.

Column 2 of this table gives the number of nodes in the Bayesian network. We list μ_e and elapsed time for PLS with 1000 samples in column 3 and 4 respectively. Column 5 and 6 of this tables gives μ_e and elapsed time with 10,000 samples. Column 7 gives the logic simulation time. Mean estimation error with 1000 samples is 0.017 and with 10000 samples is 0.016. Average estimation time with 1000 samples and 10,000 samples are 5.79 seconds and 55.12 seconds respectively, whereas average simulation time is 161.43 seconds. These results show the effectiveness of our model in terms of accuracy and estimation time. We see that estimation time varies linearly with number of samples, whereas average estimation error is almost the same with 1000 and 10000 samples for most of the circuits. With PLS 1000 samples, circuit c2670 gave the lowest estimation error of 0.01 and circuits, c2670 and c5315, gave the highest estimation error of 0.07 over all circuits. With PLS 10000 samples, again c2670 gave lowest μ_e of 0.0009 and c3540 gave highest μ_e of 0.07.

Error Sensitivity: One of the attractive features of our model is that it is easy to incorporate error in each nodes. So having different node error probabilities at different nodes can be easily accomplished. To validate this aspect we have provided some results with variable node error probabilities. Fig. 6 gives the comparison of average output error probability for $c432$ with gate error probabilities $p = 0.001$, $p = 0.01$ and with each node getting a different gate error probability p chosen randomly from 0.001 to 0.01. It can be observed that the duration of inference is almost same making it evident that the complexity is not increased due to variable gate error probabilities. Also the average of the randomly chosen p values between 0.001 and 0.01 is 0.005. This shows that even when the gate error probabilities in the circuit are evenly distributed between 0.001 and 0.01, the average output error probability is closer to that with $p = 0.01$ than with $p = 0.001$. It indicates that the output is affected more by the gates with higher gate error probabilities.

Based on the above concept, we can provide error only to some specific nodes to study their

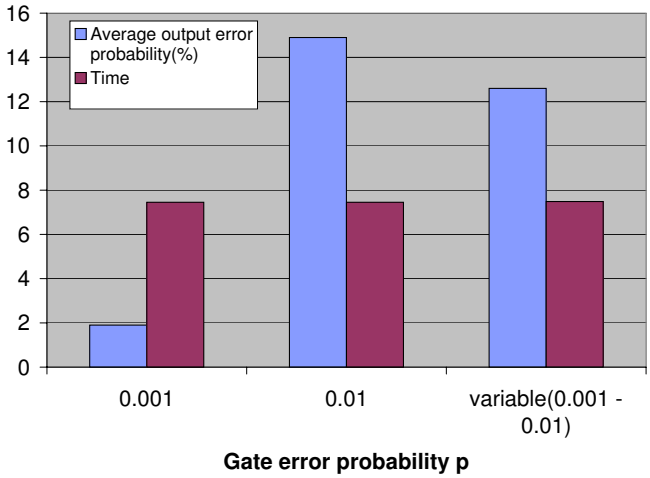
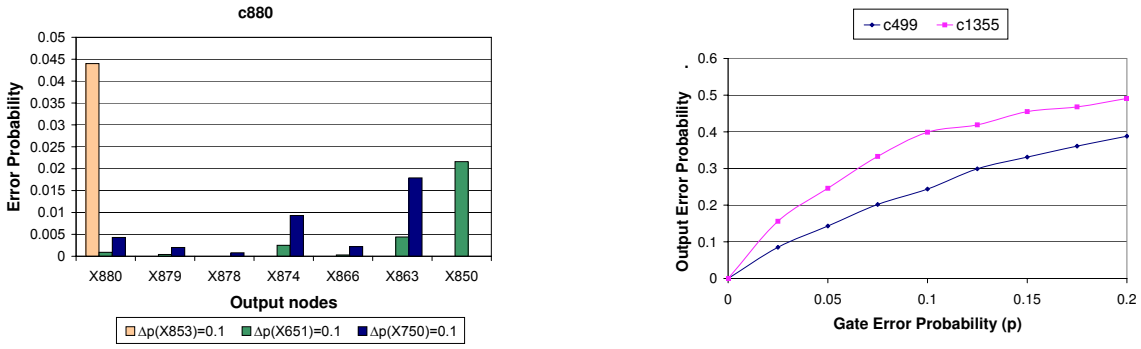


Fig. 6. Comparison of average output error probabilities for $c432$ with gate error probability $p = 0.001$, $p = 0.01$ and with p chosen randomly from 0.001 to 0.01 for each gate.

TABLE VIII
ERROR SENSITIVITY FOR C17

	Error probability at outputs	
Gate δp	22	23
Δp(X10)=0.1	0.0625	0
Δp(X11)=0.1	0.0375	0.075
Δp(X16)=0.1	0.075	0.0625
Δp(X19)=0.1	0	0.0625

sensitivity. We compute sensitivity of output error to a particular gate with a Δp change in gate error, rest of the gates having zero gate error probability. This is useful for determining the application of redundancy in achieving nano-domain fault tolerance. In Table VIII, we tabulate the output error probability corresponding to gate 22 and 23 of benchmark $c17$, when the error probabilities of gates 10, 11, 16, *and* 19 change by 0.1. It is clear that output error at gate 22 is most sensitive to gate 16. We also plot sensitivity of a few individual gates for benchmark c880 in Figure 7(a). As it can be seen that output error at node 880 is sensitive only to gate 853 out of the three intermediate gates shown here and gate output error at node 863 is reasonably sensitive to both node 651 and node 750.



(a) (b)

Fig. 7. (a)Sensitivity of output error probability with respect to individual gate errors for c880 (b) Output error profiles of two alternative logic implementation (c499 and c1355)

Design Space Exploration: Figure 7(b) shows the variation in output error with gate error p for two ISCAS benchmark c499 and c1355 that are logically equivalent. We see that c499 is clearly a better design of the logic for nano-domain in terms of resistance to dynamic errors. The circuit c1355 is more sensitive to dynamic-error almost for all individual gate error probabilities, specially when p=0.1, the output error probability become 0.4 as opposed to 0.25 for c499. The expected output dynamic error can be used, along with other design measures such as power and area for nano-domain circuits.

Input Space Exploration: In this section, we explore input characteristics for a desired output behavior. Namely, what should be the input probabilities, entropy, likelihood etc for such a desired behavior. Many of the above results in the previous subsection is conducted under random inputs. However, inputs themselves might aggravate or eliminate some of the serious problems that we face. There are two key components for such characterization. How do we propagate probabilistically from a desired output characteristic (can be any node really)? Which input characteristics would be beneficial for the designers? The answer to the question lies in the unique

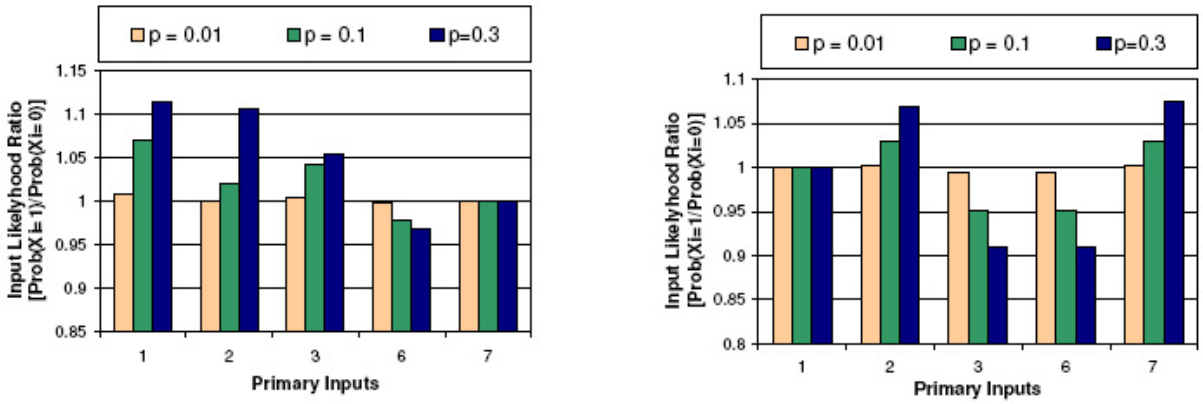


Fig. 8. c17 - Input space characterization by Likelihood ratio for (a) Zero error at output node 22 (b) For zero error at output node 23

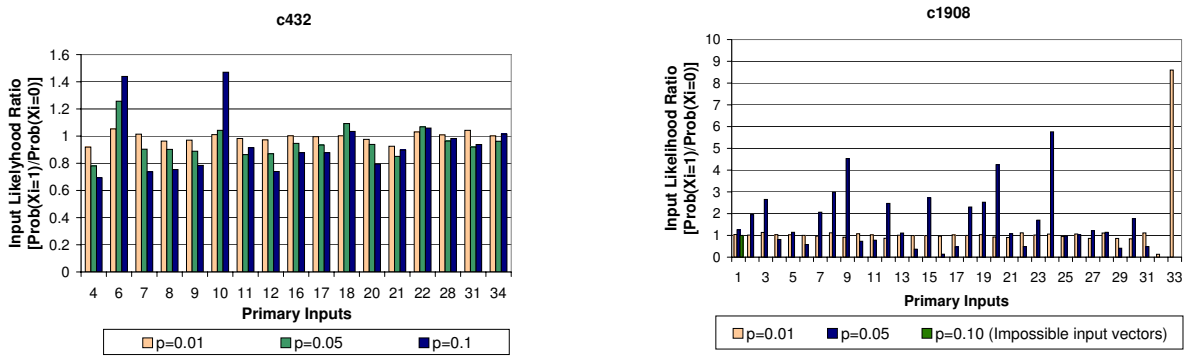
back-tracking capability of Bayesian network based models. In fact, BN models are unique for not only predictive inference but their main application is in diagnostic analysis. We show that exact inference scheme can be used for input space characterization of small circuits and approximate inference scheme for medium sized benchmark circuits. The logic sampling provides inaccurate estimates for backtracking as discussed in section B. Hence we resort to EPIS based inference for probabilistic diagnosis. We use likelihood of logic one as an input behavior. Designers are free to use any other metrics to characterize the inputs.

Input Space - Exact inference

We characterize the input space of c17 benchmark for obtaining some specific output behavior given the dynamic error probability of individual gates. We set some specific output error probabilities as evidence in the LIPEM model and back propagate these probabilities to obtain the corresponding input probabilities. In Fig 8(a) and (b), we plot the input space in terms of likelihood ratio $Prob(X_i = 1)/Prob(X_i = 0)$ for primary inputs 1, 2, 3, 6 and 7 by setting (a) error probability of output 22 at zero and (b) error probability of output 23 at zero, respectively. We plot the graphs for three different gate error probabilities, $p = 0.01$, $p = 0.1$, and $p = 0.3$. Likelihood ratio is equal to one if the input probability is 0.5 which indicates randomness of the input.

From Fig. 8(a) and (b), it can be seen that, with individual gate error probability $p = 0.01$, all the inputs have likelihood ratio equal to, or close to one. In other words, random inputs are likely to produce zero output error probability, if individual gate error probability is $p = 0.01$. Thus it can be concluded that if we have no information about the inputs, i.e. if the actual input space of this circuit is random, we don't have to apply any dynamic error mitigation technique to the gates in the circuit since the circuit has in-built error tolerance. As the gate error probability increases, we can see that input likelihood moves further away from one, indicating that the circuit requires some error mitigation schemes if the actual input pattern is different from the pattern shown in the graph. For example, with gate error probability $p = 0.3$, in order to have no error at output 22, input 1 should have more ones than zeros. If the actual input pattern has more zeros than ones for this input, we should definitely apply additional redundancy techniques to make the circuit error-free. From Fig. 8(a), it is evident that inputs 1 and 2 follow almost same pattern as gate error probability increases, whereas input 6 follows an opposite pattern. Likelihood of input 6 being at logic 0 increases as the gate error probability increases from 0.01 to 0.3 in order to get zero output error at node 22. Hence in the actual input space of the circuit, if input 6 has more 1 values than 0's, circuit requires more stringent redundancy techniques and vice-versa. From Fig. 8(b), it can be seen that likelihood of inputs 2 and 7 being at logic one increase with increase in gate error probability, whereas inputs 3 and 6 exhibit the opposite behavior. Input 1 has likelihood ratio equal to one for all gate error probabilities which shows that the logic level of this input does not influence the output error probability at node 23 for any gate error probabilities. Since $c17$ is a very small benchmark circuit, the effect of inputs on the overall output error is not very important. To demonstrate the usefulness of our model, we experiment input space characterization on mid-size benchmarks, which is explained below.

Input Space - Approximate Inference



(a) (b)

Fig. 9. Input space characterization by Likelihood ratio for benchmarks (a) c432 (b) c1908

To handle medium sized benchmark circuits, we use the approximate inference scheme, namely EPIS, for input space characterization. In the following three set of experiments on c432 and c1908, we make the gate-error probability of all gates in the circuit to 0.01 and forced all the output error probabilities to zero (also termed evidence), then we back-track and find out the input probabilities that will give the desired output characteristic of zero error.

These input probabilities are then used to calculate input likelihood ratios. In Figure 9(a), we plot the likelihood ratios $Prob(X_i = 1)/(Prob(X_i = 0)$ of selected input of benchmark c432. As it can be seen that for this circuit, with gate error probability $p = 0.01$, the likelihood is close to one for most of the inputs. This implies that to have no output error, primary inputs would have equal probability of being at logic 0 or at logic 1, if the gate error probability $p = 0.01$. We then repeat this by increasing the individual gate error probability values to p=0.05 and then to p=0.1. It is clear that input likelihoods are further away from one in both directions, for gate error probability $p = 0.1$. For example, inputs 6 and 10 have likelihood ratio, close to 1.5 for achieving zero output error with a gate error probability $p = 0.1$, indicating that probability of these inputs being at logic one should be higher than that of being at logic zero. Hence it is clear that if these

inputs have more 0's than 1's in the actual input pattern, then the circuit is less resistant to dynamic errors and needs mitigation schemes to make it error-tolerant. However, inputs 4 and 12 have low likelihood ratios, close to 0.7, for gate error probability $p = 0.1$. Circuit behavior with respect to these inputs are exactly opposite to that with respect to inputs 6 and 10. We can also observe that some inputs like, 4, 8, 9, 11, 12, 20 and 21 have likelihood ratio less than one for all three gate error probabilities whereas some other inputs like, 6, 10 and 22 always have likelihood ratio greater than one. From these observations, it can be concluded that if inputs in the *first* category have more number of *zeros* than *ones* and inputs in the *second* category have more number of *ones* than *zeros*, in the input pattern, this circuit possesses in-built error tolerance with respect to dynamic errors and doesn't require severe redundancy techniques. However, if the input pattern is different from above, the circuit gives erroneous outputs and designers have to incorporate error tolerant schemes for reliable operation.

These experiments are repeated for c1908 and we observe that most inputs have a likelihood of one for p=0.01 as seen in Fig. 9(b). For p=0.05, we have many inputs with likelihood much higher than one indicating that logic one at those inputs is more probable to cause zero output errors. If the actual input patterns have more zeros, the circuit will give erroneous outputs and hence gates are to be targeted for redundancy application. We can see another important result from Fig. 9(b). With p=0.1, no input vectors can account for a zero output error, which means that no input combination can give zero output error if the gate error probability reaches 0.1. Hence $p = 0.1$ is an upper bound on gate error probability, beyond which reliable computation is impossible. We need to have effective error mitigation schemes for circuit c1908 to achieve reliable computing if the individual gate error probability is 0.1 or above.

Input error modeling: Handling input errors can be easily accomplished in our model. We attached a buffer node, which is represented as input error node, with each primary input and

TABLE IX
AVERAGE OUTPUT ERROR PROBABILITIES FOR ISCAS'85 CIRCUITS WITH GATE ERROR PROBABILITY $p = 0.01$ AND FOR DIFFERENT INPUT ERROR PROBABILITIES $p_{in} = 0.01$, $p_{in} = 0.001$ AND $p_{in} = 0.0001$.

Circuit	Avg. Output error prob. for individual gate error prob. $p = 0.01$		
input error prob. p_{in}	=0.01	=0.001	=0.0001
c432	0.239	0.221	0.216
c499	0.061	0.052	0.052
c880	0.339	0.310	0.302
c1355	0.100	0.093	0.093
c1908	0.396	0.385	0.385

induced error in them. The induction of error is done in the same way as we have done for the internal gate nodes. The individual input error probabilities p_{in} for all input error nodes are fixed to 0.01, 0.001 and 0.0001 with the individual gate error probabilities p for all gate nodes fixed to 0.01. The results are shown in Table. IX. It can be noted that in some circuits like $c499, c1355$ and $c1908$ a small increase in p_{in} from 0.0001 to 0.001 is not affecting the output. Also increase in p_{in} from 0.001 to 0.01 increases the average output error probability for all the circuits indicating the effect of input error.

Reliability Enhancement: The effects of dynamic errors in logic circuits can be mitigated by application of various error-tolerant techniques thereby achieving a given level of reliability. Application of redundancy to all gates in a circuit will result in very high area overhead and excess power dissipation in addition to increased cost. We need to choose gates that are highly sensitive to dynamic errors and apply selective redundancy measures to achieve trade-off between redundancy, reliability, area overhead and cost.

We first estimate dynamic error sensitivities of selected gates in a circuit. Gates which are logically closer to the primary outputs are considered to be more sensitive to dynamic errors than those gates which are at higher logical depths from primary outputs. Hence we choose gates which are within a specific logical depth from primary outputs. We compute error sensitivities of each of these selected gates and then compute percentage reduction in overall output error probabilities by

applying selective redundancy to highly sensitive nodes. This is explained below:

- Step 1: Estimate the overall output error probabilities with a given dynamic error probability (say $p = 0.01$) for ALL gates in the circuit ($\{P_{E_1}, P_{E_2}, \cdots, P_{E_N}\}$).
- Step 2: Give a higher error probability (say $p = 0.05$) for the gate whose error sensitivity is to be determined keeping all other gate error probabilities same as before (0.01) and estimate the overall output error probabilities ($\{P'_{E_1}, P'_{E_2}, \cdots, P'_{E_N}\}$).
- Step 3: Compute the difference in estimated output error probabilities from step 1 and step 2 ($\Delta P = \{|P_{E_1} - P'_{E_1}|, |P_{E_2} - P'_{E_2}|, \cdots, |P_{E_N} - P'_{E_N}|\}$).
- Step 4: Determine the set of gates $\{S\}$ which give $\max(\Delta P) \geq$ a threshold value (δp_{th}). These gates are to be selected for application of redundancy. Value of δp_{th} is chosen depending on the desired error tolerance.
- Step 5: We estimate overall output error probabilities by giving a low dynamic error probability (say $p = 0.001$) for gates selected for redundancy ($gates \in \{S\}$) and original error probability $p = 0.01$ for all other gates ($gates \notin \{S\}$) in the circuit. We compare these values with output error probabilities for 100% redundancy (by giving $p = 0.001, \forall gates$) and compute reliability/redundancy trade-off in terms of percentage reduction in average output error probabilities.

Results for circuits c17 and c432 in Table X and Table XI respectively. Column 1 of these tables gives the δp_{th} values and column 2 gives the percentage of gates selected for redundancy for each δp_{th}. In column 3, we report the percentage reduction in output error probabilities by applying selective redundancy. Number of gates to be selected for redundancy application depends on the error threshold δp_{th}. With decrease in δp_{th}, more number of gates are selected for redundancy, thus reducing the output error probability. For example, from Table XI, it can be seen that by decreasing δp_{th} from 0.020 to 0.001, we achieve reliability improvement from 11% to 60% measured in terms

TABLE X
C17-SELECTIVE REDUNDANCY

δp_{th}	% of Gates selected for redundancy	Percentage reduction in average output error probability
0.025	60%	69.91
0.000	100%	89.80

TABLE XI
C432-SELECTIVE REDUNDANCY

δp_{th}	% of Nodes selected for redundancy	Percentage reduction in average output error probability
0.020	8%	11%
0.015	14%	17%
0.010	24%	32%
0.005	37%	50%
0.001	59%	60%
0.000	100%	88%

of reduction in output error probability. Thus our modeling tool can be used for choosing the gates to be selected for application of error-tolerant techniques and also for determining the amount of such techniques required for achieving a desired error tolerance.

VII. DISCUSSION AND ONGOING WORK

We presented an exact probability model, based on Bayesian networks, to capture the inter-dependent effects of dynamic errors at each gate. Dynamic error at each gate is modeled through the conditional probability specifications in the Bayesian network. The expected output error can be used to select between different implementation of the same logical function, so as to result in reliable nano-level circuits.

We theoretically proved that the dependency model for probabilistic gate errors is a Bayesian Network and hence is an exact, minimal representation. We used exact inference scheme for small circuits and show that our model is extremely time and space efficient compared to existing PTM based technique. To handle even larger benchmarks, we used two stochastic sampling algorithms,

PLS and EPIS. Even though many of the futuristic techniques do not yet offer benchmarks, we demonstrate scalability of our estimation tool by using the logic level specifications of the ISCAS'85 benchmarks, which would be valid for uncertainty modeling in nano-CMOS. We studied the input characteristics of logic circuits for a desired output behavior by exploring the unique back-tracking feature of Bayesian networks. By conducting a sensitivity analysis of gates in the circuits, we were able to identify gates that affect the overall circuit behavior due to dynamic errors and present a technique to determine the amount of redundancy to be applied for achieving a desired improvement in circuit performance.

We are currently working on modeling dynamic error tolerant designs by applying TMR redundancy on selected nodes having high dynamic error probabilities based on their switching characteristics and other device features. We intend to pursue the dynamic error for QCA logic blocks and model interconnect gate error in future. A relative study on various technologies like CNT, RTD with respect to dynamic error would be easily extended by this framework. The other more difficult task would be to handle sequential logic in terms of dynamic errors.

REFERENCES

[1] S. Spagocci and T. Fountain, "Fault rates in nanochip devices," in *Electrochemical Society*, pp. 582–593, 1999.

[2] J. Han and P. Jonker, "A defect- and fault-tolerant architecture for nanocomputers," *Nanotechnology*, vol. 14, pp. 224–230, 2003.

[3] K. Nikolic, A. Sadek, and M. Forshaw, "Fault-tolerant techniques for nanocomputers," *Nanotechnology*, vol. 13, pp. 357–362, 2002.

[4] S. Krishnaswamy, G. S. Viamontes, I. L. Markov, and J. P. Hayes, "Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices", *Design Automation and Test in Europe (DATE)*, March 2005.

[5] R. Iris Bahar, J. Mundy, and J. Chan, "A Probabilistic Based Design Methodology for Nanoscale Computation", *International Conference on Computer Aided Design*, 2003.

[6] G. Noman, D. Parker, M.Kwiatkowska and S. K. Shukla, "Evaluating the reliability of defect-tolerant architectures for nanotechnology with probabilistic model checking", *International Conference on VLSI Design*, 2004.

[7] J. Han, J. B. Gao,P. Jonker, Yan Qi and J.A.B. Fortes, "Toward hardware-Redundant Fault-Tolerant Logic for Nanoelectronics" *IEEE Transactions on Design and Test of Computers*, vol. 22-4 pp. 328–339, July-Aug. 2005.

[8] J. B. Gao, Yan Qi and J.A.B. Fortes, "Bifurcations and Fundamental Error Bounds for Fault-Tolerant Computations" *IEEE Transactions on Nanotechnology*, vol. 4-4 pp. 395–402, July 2005.

[9] J. Chen, J. Mundy, I. Bahar, and J. M. Xu, "Fault-tolerance Carbon Nanotube-based Computer Logic Design," *Foresight Conference on Molecular Nanotechnology*, Oct. 2003.

[10] J. B. Gao, Yan Qi and J.A.B. Fortes, "Markov Chains and Probabilistic Computation - A general Framework for Multiplexed Nanoelectronic Systems" *IEEE Transactions on Nanotechnology*, vol. 4-2 pp. 395–402, march 2005.

[11] Jie Han, Erin Taylor, Jianbo Gao and Jose Fortes, '"Reliability Modeling of Nanoelectronic Circuits" *IEEE Conference on Nanotechnology*, July 2005.

[12] R. Martel, V. Derycke, J. Appenzeller, S. Wind, and Ph. Avouris, "Carbon Nanotube Field-Effect Transistors and Logic Circuits," *Proceedings of the 39th Conference on Design Automation*, 2002.

[13] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," in *Automata Studies* (C. E. Shannon and J. McCarthy, eds.), pp. 43–98, Princeton Univ. Press, Princeton, N.J., 1954.

[14] S. Winograd and J. D. Cowan, *Reliable Computation in the Presence of Noise*. The MIT Press, 1963.

[15] T. Rejimon and S. Bhanja, "Probabilistic Error Model for Unreliable Nano-logic Gates," *Proc. IEEE Conference Nanotechnology*, July 2006.

[16] R. K. Kummamuru, A. O. Orlov, R. Ramasubramanyam, C. S. Lent, G. H. Bernstein, and G. L. Snider, "Operation of Quantum-Dot Cellular Automata (QCA),Shift Registers and Analysis of Errors" *IEEE Transactions on Electron Devices*, vol. 50-59, pp. 1906–1913, June 1993.

[17] M. B. Tahoori, M. Momenzadeh, and J. Huang, and F. Lombardi, "Defects and Faults in Quantum Cellular Automata at Nano Scale" *Workshop on Fault Tolerance in Parallel and Distributed Systems*, 2004.

[18] J. P. Sun, G. I. Haddad, and P. Mazumder, "Resonant Tunneling Diodes: Models and Properties" *Proceedings of the IEEE*, vol. 86-2, pp. 641–661, April 1998.

[19] S. Bhanja and N. Ranganathan, "Cascaded Bayesian inferencing for switching activity estimation with correlated inputs," *IEEE Transaction on VLSI*, vol. 12-12, pp. 1360–1370, Dec. 2004.

[20] S. Bhardwaj, S.B.K. Vrudhula and D. Blaauw, "*t*AU: Timing analysis under uncertainty" *International Conference on Computer Aided Design*, Nov. 2003.

[21] T. Rejimon and S. Bhanja, "An Accurate Probabilistic Model for Error Detection ," *Proc. IEEE International Conference on VLSI Design*, pp. 717–722, Jan. 2005.

[22] J. Cheng, "Efficient Stochastic Sampling Algorithms for Bayesian Networks," *PhD Dissertation, University of Pittsburgh*, 2001.

[23] C. Yuan and M. J. Druzdzel, "An Importance Sampling Algorithm Based on Evidence Prepropagation," *Proceedings of the 19th Annual Conference on Uncertainty on Artificial Intelligence*, pp. 624-631, 2003.

[24] M. Henrion, "Propagation of uncertainty by probabilistic logic sampling in Bayes' networks,"*Uncertainty in Artificial Intelligence*, 1988.

[25] K.P. Murphy, Y. Weiss and M.I. Jordan "Loopy belief propagation for approximate inference: an empirical study," *In Proceedings of Uncertainty in AI*, pp. 467–475, 1999.

[26] URL http://www.hugin.com

[27] ''GeNIE'' URL

http://www.sis.pitt.edu/~genie/genie2

[28] R. G. Cowell, A. P. David, S. L. Lauritzen, D. J. Spiegelhalter, "Probabilistic Networks and Expert Systems," Springer-Verlag New York, Inc., 1999.

[29] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference", Morgan Kaufmann Publishers, Inc., 1988.

[30] K. Murphy, "A Brief Introduction to Graphical Models and Bayesian Networks", http:/http. cs. Berkeley. edu/murphyk/Bayes/bayes.html, 2001.

[31] N. Pippenger, "Reliable Computation by Formulas in the Presence of Noise", *IEEE Trans on Information Theory*, vol. 34(2), pp. 194-197, 1988.

[32] S. Bhanja and K. Lingasubramanian and N. Ranganathan, "A stimulus-free graphical probabilistic switching model for sequential circuits using dynamic bayesian networks", *ACM Transactions on Design Automation of Electronic Systems*, vol. 11(3), pp. 773-796, 2004.

VIII. Response to Reviewers

Associate Editor's Comments: Both our reviewers are impressed with your work but have lengthy comments. One reviewer is a young researcher and the other, a matured scientist with many years of experience. Together they represent the typical reader group for your paper once it is published. Note that there are many things that are not expressed well. I would like you to take their comments very seriously and revise the paper. In addition, the abstract of the paper

should be rewritten. A well written abstract should contain (a) problem statement, (b) your specific method that allowed you to solve the stated problem, and (c) main theoretical and numerical results including some comparative benchmark example(s). Other things, such as, motivation, background, etc., do not belong in the abstract and should be placed at appropriate places in the paper.

We thank the AE for compiling all the key issues and concerns about our journal paper. We have rewritten the abstract according to AE's suggestions. We have revised our manuscript significantly based on the reviewers' comments addressing all the relevant issues. We first provide the summary of all the revisions and then the responses describing the detailed revisions made for each of the reviewers' comments.

A. Summary of Revisions

- We have rewritten the abstract according to the AE's suggestions.
- We have updated Table. V with results using 10000 samples instead of 1000 samples. We also provide justification that 10000 samples are adequate. In Table. VI we show the results with 1000, 10000 and 100000 samples for $p = 0.001$ and $p = 0.0001$. If we compare the estimates with 1000 samples and 10000 samples we can see that for some circuits like c499, c880, c1908, c7552 the estimates are close to the second decimal place. But other circuits like c432, c2670, c6288 are not close enough. If we compare the estimates with 10000 samples and 100000 samples, it can be clearly seen that for all circuits the estimates are equal to the second decimal place and close to the third decimal place. This shows that 10000 samples are adequate to generate the estimates.
- We have added results to show that our model is not limited to networks with fixed p values but can also handle networks with variable p values. Fig. 6 gives the average output error probability for $c432$ gate error probability $p = 0.001$, $p = 0.01$ and with each node getting

a different gate error probability p chosen randomly from 0.001 to 0.01. It can be observed that the duration of inference is almost same making it evident that the complexity is not increased due to variable gate error probabilities. Also the average of the randomly chosen p values between 0.001 and 0.01 is 0.005. This shows that even when the gate error probabilities in the circuit are evenly distributed between 0.001 and 0.01, the average output error probability is closer to that with $p = 0.01$ than with $p = 0.001$. It indicates that the output is affected more by the gates with higher gate error probabilities.

- We have added results to show that our model can handle input errors (Table. IX). We attached a buffer node with each primary input in the error encoded circuit block and induced error in them in the same way as we did for the intermediate nodes. So the model would not change drastically. The individual input error probabilities p_{in} for all input error nodes are fixed to 0.01, 0.001 and 0.0001 with the individual gate error probabilities p for all gate nodes fixed to 0.01. It can be seen that the output error probability steadily increases with the increase in input error probability. This shows that our model can effectively handle input errors.
- We have removed the table comparing the simulation time between PTM model and LIPEM model and instead added a theoretical discussion on the relative time and space complexity of both models and through this discussion we have shown the efficiency of our model (see pages. 23-24). We have also added a table comparing the reliability estimates obtained from PTM model and LIPEM model for gate error probability $p = 0.05$ (Table. III). These results are obtained by calculating the probability of correctness at the output. These results indicate that our model and PTM model produces similar estimates.
- We have added a discussion that explains the advantage of using Bayesian Networks based model over MRF based model. (see Page. 9)

MRF are undirected graphs and Bayesian Networks are directed graphs. This directional feature adds significant impact in modeling conditional independencies, also called I-map [19] which are the underlying rationale to arrive at a minimal (sparse) structure. The d-separation or directional separation (in a DAG) is directly coupled with I-maps in a causal system which let us model in an edge-minimal fashion [30], [29]. If the information flow through an edge is not known, then independencies would not be captured properly [19].

Fig. 11 [29] illustrates a few important aspects of dependency models and underlying graph structures. First, not all dependency models can be embedded in a graph. Also, it can be observed that dependencies that a BN can model might not be captured in MRF and vice versa [29], [30]. In logical scenario where inputs drive output, i.e. causal setup, BN is the natural choice and is the best model encapsulating all the dependencies retaining all the independencies (related to sparsity or minimality) [29], [30].

For an example, suppose we have two input AND gate. If the output is "0", the inputs becomes dependent (if one of the input is "1", the other is forced to be a "0" to maintain the evidence at output node "0"). Even though two inputs can be independent, the observation of the child imposes dependence in the parents. This is true in logic and causal networks. Since d-separation in a DAG models this dependence, we do not need an additional link between parents where as in an undirected environment, additional links between parents becomes necessary and hence the graph is not as sparse as it should be.

If we may politely point out, MRF is not an undirected Bayesian network. Bayesian networks are inherently directed, due to the use of conditional probabilities. If one removes the directions in a BN one does not get an MRF. One has to go through a series of reasoned transformations to preserve all the dependences and most of the independencies.

One can see that decomposable graphs (chordal graphs) can embed dependencies expressed

by both directed and undirected graph and is the intersection of BN and MRF model. Every chordal graph can possess at least one junction tree. One might see that if we add all nodes together, it is a junction tree of one clique containing all nodes. As complexity of inference is exponential to the nodes in the largest clique of the junction tree, the key is to find a junction tree that has the smallest number of nodes in its largest clique (again drives us to the I-map and sparsity) and then we will have the computational leverage over marginalizing the numerical joint pdf. For modeling a causal network, junction tree cliques can be largely minimized when arrived through parent-child relationship as opposed to an MRF. Superiority of BN over MRF model is not determined by arriving at a junction tree but by arriving at a smallest junction tree.

- We have addressed all the reviewer's questions and doubts. We have also corrected all the mistakes that the reviewers had pointed out.

B. Detailed Response

In this subsection, we respond to each of the reviewers comments individually and also present the corresponding changes in the manuscript. We sincerely thank you for all your detailed analysis and comments.

Response to reviewer 1's comments:

We thank the reviewer for all the constructive suggestions.

1. *When execute exact inference for small circuit and stochastic inference, the stochastic inference need large enough samples to get correct result, when the node error probability is very small.*

   We agree with the reviewer and acknowledge the suggestion. We have provided the results with 10000 samples instead of 1000 samples (see Table. V). In Table. XII we show the results with 1000, 10000 and 100000 samples for $p = 0.001$ and $p = 0.0001$. If we compare

TABLE XII
COMPARISON OF AVERAGE OUTPUT ERROR PROBABILITIES FOR ISCAS'85 CIRCUITS -PLS WITH 1000, 10000 AND 100000 SAMPLES.

Circuit	Average output error probability for individual gate error probability p					
	=0.001			=0.0001		
	1000 samples	10000 samples	100000 samples	1000 samples	10000 samples	100000 samples
c432	0.024	0.019	0.020	0.002	0.003	0.003
c499	0.007	0.005	0.004	0.003	0.001	0.001
c880	0.032	0.031	0.028	0.005	0.003	0.004
c1355	0.014	0.010	0.008	0.004	0.002	0.001
c1908	0.066	0.067	0.068	0.013	0.012	0.011
c2670	0.065	0.075	0.070	0.011	0.011	0.011
c3540	0.212	0.216	0.211	0.051	0.044	0.043
c5315	0.071	0.071	0.068	0.012	0.011	0.011
c6288	0.220	0.209	0.205	0.049	0.043	0.041
c7552	0.146	0.142	0.147	0.030	0.027	0.026

the estimates with 1000 samples and 10000 samples we can see that for some circuits like c499, c880, c1908, c7552 the estimates are close to the second decimal place. But other circuits like c432, c2670, c6288 are not close enough. If we compare the estimates with 10000 samples and 100000 samples, it can be clearly seen that for all circuits the estimates are equal to the second decimal place and close to the third decimal place. This shows that 10000 samples are adequate to generate the estimates.

2. *The author presents the output error probability varies when gate error probabilities are 0.005, 0.05 and 0.1 respectively, is there a possibility that two nodes have error simultaneously? If it is possible, then those two errors may mask each other based on the circuit structure.*

   We agree with the reviewer. Two errors occurring simultaneously will mask each other due to logic masking. Our model handles logic masking and this is the primary reason for the results to match with logic simulation results.

3. *And in table III, the time collapsed for this model is much smaller than PTM (ref. [4] in the paper) mode, different execution environment is noticed.*

   We have removed the table comparing the simulation time between PTM model and LIPEM

TABLE XIII
COMPARISON OF THE RELIABILITY ESTIMATES OBTAINED FROM PTM MODEL AND LIPEM MODEL FOR GATE ERROR PROBABILITY $p = 0.05$

Circuit	Circuit reliability for individual gate error probability $p = 0.05$	
	LIPEM model	PTM model
c17	0.866	0.866
parity	0.603	0.603
pcle	0.844	0.844
decod	0.935	0.935
cu	0.803	0.803
pm1	0.737	0.737
xor5	0.566	0.566

model and instead added a theoretical discussion on the relative time and space complexity of both models and through this discussion we have shown the efficiency of our model (see pages. 23-24). We have also added a table comparing the reliability estimates obtained from PTM model and LIPEM model for gate error probability $p = 0.05$ (Table. XIII). These results are obtained by calculating the probability of correctness at the output. Since both models exactly capture the circuit structure, the results produced are similar to each other.

4. *The comparison of BN model results and logic simulation result (in Table V) to show the accuracy of BN model should be re-considered. In a circuit, one logic dynamic error may be masked by 1) temporal masking, 2) electrical masking and 3) logic masking. In logic simulation, we only consider the logic masking thus take into account of the property of each gate, the error has some probability to be masked. So in the logic simulation result the output error rate should be smaller than BN model. In logic simulation, input vector also play an important role in activate & propagate the dynamic error to the primary output, random vector set should be large enough to make the result more reasonable.*

   Our model is an abstract probabilistic model. The masking effects like temporal and electrical masking which are sensitive to device parameters cannot be modeled probabilistically [4], [5], [6], [11]. But our LIPEM model handles logic masking and thats one of the

TABLE XIV
COMPARISON OF AVERAGE OUTPUT ERROR PROBABILITIES FOR ISCAS'85 CIRCUITS -LOGIC SIMULATION WITH 500000 AND 5000000 RANDOM INPUT VECTORS.

Circuit	Average output error Probability for individual gate error probability $p = 0.01$		
	LIPEM model	Logic simulation	
		500000 random input vectors	5000000 random input vectors
c432	0.149	0.151	0.150
c499	0.034	0.030	0.030
c880	0.209	0.207	0.207
c1355	0.066	0.061	0.060
c1908	0.327	0.324	0.324
c2670	0.380	0.391	0.391
c3540	0.501	0.496	0.496
c5315	0.379	0.376	0.376
c6288	0.500	0.496	0.496
c7552	0.497	0.499	0.499

reasons for the results to match with logic simulation. Also we are sure that 500,000 random vectors are good enough to generate the logic simulation results. In Table. XIV, we have shown the logic simulation results for 500,000 random vectors and 5 million random vectors. It can be clearly seen that the results converge perfectly for all circuits and this shows that 500,000 random vectors are adequate to generate logic simulation results.

5. *Also, in BN model, if more accuracy needed error probability on every nodes should not be equal (Should take into account of node sensitivity to errors).*

   We thank the reviewer for pointing this out. One of the attractive features of our model is that it is easy to incorporate error in each node. So having different node error probabilities at different nodes can be easily accomplished. To validate this aspect we have provided some results with variable node error probabilities. Fig. 10 gives the average output error probability for $c432$ gate error probability $p = 0.001$, $p = 0.01$ and with each node getting a different gate error probability p chosen randomly from 0.001 to 0.01. It can be observed that the duration of inference is almost same making it evident that the complexity is not increased due to variable gate error probabilities. Also the average of the randomly chosen p values between 0.001 and 0.01 is 0.005. This shows that even when the gate error probabilities in the circuit are evenly distributed between 0.001 and 0.01, the average output

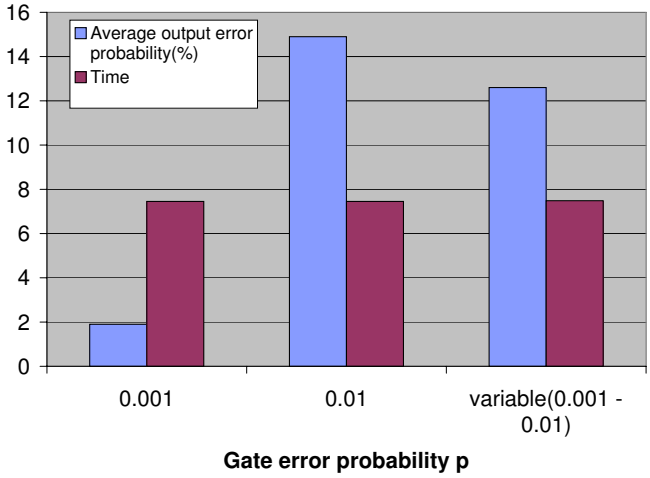


Fig. 10. Comparison of average output error probabilities for $c432$ with gate error probability $p = 0.001$, $p = 0.01$ and with p chosen randomly from 0.001 to 0.01 for each gate.

error probability is closer to that with $p = 0.01$ than with $p = 0.001$. It indicates that the output is affected more by the gates with higher gate error probabilities. Based on the above concept, we can provide error to selected nodes to study their sensitivity. (Pages. 28-29)

6. *Input vector space search based on both Exact Inference and Approximate Inference, shows that the input vectors can be well chosen to mitigate the dynamic errors. In Figure 7, what is the algorithm to find the input likelihood when the error probability is zero and all gates probability error rate is 0.01 by back-tracking? We see, for most input the likelihood is almost "1", what can we take use of this information? And for combinational circuit, the vectors are not time-correlated, only searching the likelihood of input space is not enough.* Back-tracking is one of the useful applications in Bayesian Networks. In Bayesian Networks we can, not only find the effect given the cause, we can also find the cause given the effect. We can evidence the output nodes and find the probability distribution in the input nodes using reverse inference. For input vector space search we used Evidence Pre-propagated Importance Sampling (EPIS), which is described in section B at pages 20-21. Coming to Fig. 9, when the input likelihood ratio for all the inputs gets close to "1", it

means that for a completely random input space the probability of getting an error in the output is less or almost zero. In other words, the entire input space has a high probability to provide a correct output. So the uncertainty in the input space is not affecting the output. If the input likelihood ratio gets away from ”1”, it means that for a completely random input space there is high probability for an error to occur at the output. In other words, only a part of the input space has a high probability to give a correct output and the rest of the input space have a low probability to give a correct output. So we have to take care of the uncertainty in the input space. This study can provide very useful data for input space exploration during testing.

Response to reviewer 2’s comments:

We thank the reviewer for all the constructive suggestions.

1. *Overall, I liked this paper. I generally enjoy interesting applications of Bayesian networks, especially as related to fault analysis. I also appreciated some of the interesting uses of the model for additional analyses beyond just error analysis (e.g., the sensitivity and redundancy analyses). Even so, I think there are a number of areas where the paper can be improved.*

   We thank the reviewer for the encouraging remarks.

2. *It is unclear what error/fault model is being used. For example, are we assuming so kind of non-deterministic or transient stuck-at model, or are we simply attempting to account for noise in a signal due to the physical limitations that arise at nano scales. I think they are applying some kind of stochastic ”line error” model, but this is not particularly clear. This should be clarified up front and used as one of the key assumptions in the formal development of the work.*

   We thank the reviewer for pointing this out. Our model is indeed a line error model. We

TABLE XV
AVERAGE OUTPUT ERROR PROBABILITIES FOR ISCAS'85 CIRCUITS WITH GATE ERROR PROBABILITY $p = 0.01$ AND FOR DIFFERENT INPUT ERROR PROBABILITIES $p_{in} = 0.01$, $p_{in} = 0.001$ AND $p_{in} = 0.0001$.

Circuit	Avg. Output error prob. for individual gate error prob. $p = 0.01$		
input error prob. p_{in}	=0.01	=0.001	=0.0001
c432	0.239	0.221	0.216
c499	0.061	0.052	0.052
c880	0.339	0.310	0.302
c1355	0.100	0.093	0.093
c1908	0.396	0.385	0.385

thank the reviewer for pointing this out. Our model is an abstract model where the error created in the output signal, due to an error in the corresponding component, is represented. Providing an error in a node implies that the corresponding signal is erroneous.

3. *This is a two-part comment related to the completeness of the LIPEM model. First, I don't see where the model includes error that might come from the inputs. Specifically, given the comparator circuit, if the inputs are erroneous but the circuit is fine, this will not be detected because of the use of common inputs. How would the model change (and would the change be useful) if models of input error were included as well?*

   We agree with the reviewer that we are not handling input errors. We thank the reviewer for pointing this out. Handling input errors can be easily accomplished in our model. We attached a buffer node with each primary input in the error encoded circuit block and induced error in them in the same way as we did for the intermediate nodes. So the model would not change drastically. The individual input error probabilities p_{in} for all input error nodes are fixed to 0.01, 0.001 and 0.0001 with the individual gate error probabilities p for all gate nodes fixed to 0.01. The results are shown in Table. XV. It can be seen that the output error probability steadily increases with the increase in input error probability. This shows that our model can effectively handle input errors.

4. *Second, how are errors from the comparators handled? This seems to relate back to the error/fault model in that it appears the assumed model is a "line" error. In reality, the*

components themselves could lead to errors arising, and this is certainly true of the comparator circuit as well. That, in fact, is one of the tradeoffs that must be considered when incorporating self-test or fault tolerant components in a system ? what are the effects on system-level reliability given the added complexity of system components intended to improve reliability? Do you really get the desired improvement, especially given the associated cost? They discuss some of this in their experimental section, but without including error models for the added components, I don't see how you can answer that question directly.

We use the comparator as a study element used to study the differences between the error encoded logic and its ideal counterpart. The only real-time hardware component is the error encoded logic circuit. We are comparing this circuit with its ideal counterpart using comparators. The ideal logic and comparator logic are strictly study elements and are **not** additional hardware components.

5. *I wish the work was not limited to combinational circuits. While the approach to the problem is interesting in its own right, the limitation of its current use to combinational circuits raises questions of scalability and usability with more complex, real-world, sequential circuits.*

   We agree with the reviewer. Redesigning our model to handle sequential circuits is one of the future research directions of this work. Sequential circuits have several design issues like spatio-temporal interdependencies and latching window masking effects. Our model, the LIPEM, can handle the spatial dependencies but not the temporal dependencies. We need a different kind of Temporal Networks to handle this aspect. One interesting feature of sequential circuit though, the input output flow is still preserved and hence modeling with MRF might not yield the minimal model. We have already seen this effect before in

modeling power for sequential circuits [32].

6. *Related to this is the fact that, at nano scales, the digital abstraction may itself be problematic, raising a question of whether or not a Bayesian network model of a digital circuit is the right way to go. I, for one, am in favor of the Bayesian approach. I am just curious about whether the LIPEM is the right (or best) Bayesian model. Again, I think the best solution to this problem is to provide more detailed justification for the model within the context of nano scale circuits up front.*

   We thank the reviewer for this useful remark. For any computing system if the inputs drive the outputs then BN model will give the minimal representation of the underlying system by the definition of Bayesian Network (DAG D is minimal I-map of dependency model P). If the circuit structure is known then LIPEM is the best (minimal with respect to edges) DAG that captures all the dependency and hence is the Bayesian model representing the underlying input-output behaviors of the circuit. So if the device level non-functional interactions are precisely known, we can arrive at a more accurate model for the device interactions. This work is at the circuit level and assumes device level non-functional errors (cross-talk, supply noise, capacitive coupling etc) are not precisely known.

7. *In the discussion on the rationale (note spelling of the word "rationale") for using Bayesian networks, the authors make a claim that is not backed up, and I am not even sure I agree with it. Specifically, the claim is made that, because of certain causal dependencies, a Bayesian network is superior "for causal networks" over a Markov random field. First, no reference or argument is given to justify this claim. Given this, why is it necessarily the case that the BN formulation is superior to the MRF formulation? Second, an MRF is basically an undirected Bayesian network. What is interesting is that one can relate a junction tree derived from a BN (which is needed for exact inference) to an MRF. Sure, it*

may be easier to construct the BN because of the causal structure, but that says nothing about "superiority" of the model.

We thank the reviewer for pointing this out. The following discussion will help validating our claim. Also, we have added reference [29], [30] to support our claim.

It is true that MRF are undirected graphs and Bayesian Networks are directed graphs, however, this directional feature adds significant impact in modeling conditional independencies, also called I-map [19] which are the underlying rationale to arrive at a minimal (sparse) structure. The d-separation or directional separation (in a DAG) is directly coupled with I-maps in a causal system which let us model in an edge-minimal fashion [30], [29]. If the information flow through an edge is not known, then independencies would not be captured properly [19].

Fig. 11 [29] illustrates a few important aspects of dependency models and underlying graph structures. First, not all dependency models can be embedded in a graph. Also, it can be observed that dependencies that a BN can capture, might not be covered in MRF and vice versa [29], [30]. In logical scenario where inputs drive output, i.e. causal setup, BN is the natural choice and is the best model encapsulating all the dependencies retaining all the independencies (related to sparsity or minimality) [29], [30].

For an example, suppose we have two input AND gate. If the output is "0", the inputs becomes dependent (if one of the input is "1", the other is forced to be a "0" to maintain the evidence at output node "0"). Even though two inputs can be independent, the observation of the child imposes dependence in the parents. This is true in logic and causal networks. Since d-separation in a DAG models this dependence, we do not need an additional link between parents where as in an undirected environment, additional links between parents becomes necessary and hence the graph is not as sparse as it could be.

If we may politely point out, MRF is not an undirected Bayesian network. Bayesian networks are inherently directed, due to the use of conditional probabilities. If one removes the directions in a BN one does not get an MRF. One has to go through a series of reasoned transformations to preserve all the dependences and most of the independencies.

One can see that decomposable graphs (chordal graphs) can embed dependencies expressed by both directed and undirected graph and is the intersection of BN and MRF model. Every chordal graph can possess at least one junction tree, which leads us to the second point the reviewer have raised. One might see that if we add all nodes together, it is a junction tree of one clique containing all nodes. As complexity of inference is exponential to the nodes in the largest clique of the junction tree, the key is to find a junction tree that has the smallest number of nodes in its largest clique (again drives us to the I-map and sparsity) and then we will have the computational leverage over marginalizing the numerical joint pdf. For modeling a causal network, junction tree cliques can be largely minimized when arrived through parent-child relationship as opposed to an MRF. Superiority of BN over MRF model is not determined by arriving at a junction tree but by arriving at a smallest junction tree.

The reviewer is right in pointing that constructing Bayesian Networks and cascading Bayesian Networks are much simpler than specifying conditional distribution in MRF and cascading two MRF.

We have added a discussion on explaining our point in the manuscript at page 9 paragraph 1.

8. *Continuing this line of reasoning, would it be possible to handle sequential circuits (something identified as needed in the future work discussion) with MRFs? Or would the inclusion of component models help this? One of the issues that frequently arises in Bayesian*

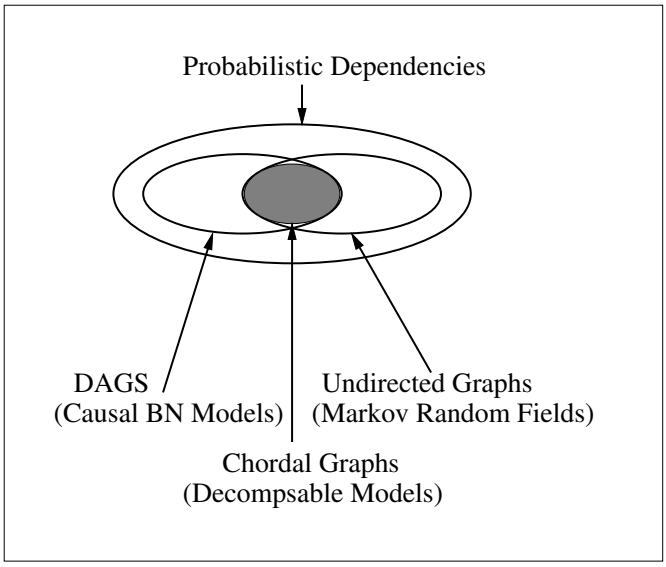


Fig. 11. Probabilistic models and graph structure [29]

formulations is that a Markov assumption hits (thus the Markov blanket notion and the Markov random field model). Unfortunately, addressing state issues becomes tricky because one loses (at least in an obvious way) the Markov property. Since this paper is not about sequential circuits, it would be nice to have a discussion on the impact of the model choice on the ability to extend beyond combinational logic because of the resulting violation of the Markov property.

The current formalism does not handle sequential circuits as mentioned before. Sequential circuits are dynamic models and are causal in one time frame and hence we feel the dynamic BN is more appropriate. We believe that arriving and specifying the conditional distribution using MRF would be a real challenge for sequential circuits.

9. *There is a bit of ambiguity when the authors talk about "diagnosis." There is the statistical version of the term where one attempts to identify causal factors leading to a set of observations, and there is the fault diagnosis version (which one can claim is a subset of the statistical) where one is trying to determine the specific fault leading to an observed failure. Which of these is being discussed? This actually relates back to the comment about the*

underlying fault/error model. If we are simply worrying about noise, then a fault diagnosis is not necessary. If we are trying to find transient fault conditions leading to the cause of the error, then fault diagnosis is necessary.

We agree with the reviewer and acknowledge the suggestion. Using our model we are trying to find out the noise signals due to transient faults and not the conditions that give rise to transient faults. We are basically finding the causal factors that might result in some specific observations. For example, we can evidence a set of output nodes and study the input and intermediate nodes using their probability distributions. The evidence would be the observation and the probability distributions would be the causal factors. So the term "diagnosis", in our context, is the statistical version where we are studying the conditions that might affect the output. We have rephrased those sentences.

10. *On page 10 (just prior to section IV), the discussion about using Bayesian networks and the inference algorithms used is basically a repeat of the discussion on the top of page 6. Please avoid such repetition.*

    We thank the reviewer for pointing this out. We have removed the repetitive sentences. (see Page 12 Paragraph 2)

11. *There seems to be a notational problem in equations (5) and (6). Specifically, in both equations, there is a single dot between z_1 and z_N. I assume that should be an ellipsis. If so, please fix. If not, please explain the notation.*

    We thank the reviewer for pointing this out. We were trying to represent all nodes from z_1 to z_N. We have corrected that. (see Page 12)

12. *I really don't like the "theorem" presented on page 13. First, that theorem seems obvious. Based on the underlying assumptions of what the model represents (which, by the way, are not well described), the "causal" structure obviously falls out as a minimal I-map. So*

what does the theorem add? Second, the theorem appears to depend heavily on the "line error" fault model since it does not allow for "non-functional" causes of error such as sneak circuits. If, indeed, some kind of error mode existed like a sneak, then that suggests additional dependencies not captured in the model. Also, should we move up the hierarchy such that models other than simple logic gates are represented, joint dependencies become more complex, and one might argue that the independence assumptions captured by the model no longer apply. I suggest eliminating the theorem.

The theorem might be redundant and obvious for an advanced reader like the reviewer but this theorem explains the basic advantage of using Bayesian network model and connects it to I-map and d-separation through the boundary DAG. Even though it might look trivial, for the sake of completeness of the paper and considering wider background of our readers, we would like to retain it.

13. *In the first paragraph of IV.B, the notation of $p(x_v|x_{parent(v_i)})$ is used. What is x_v? As far as I can tell, this is the first use of this notation, and it needs to be explained.*

    We thank the reviewer for pointing this out. This was a typo. The correct notation is x_{v_i} instead of x_v. (see Page 15)

14. *On page 17, the comment is made that the junction tree approach "is in sharp contrast with simulative approaches where flow of information always propagate (sic.) from input to the outputs." While I agree that the junction tree algorithm is very different from a simulation-based approach, I disagree that the reason is because of flow of information. MCMC-based simulation approaches do not require the flow to be from input to output. Even belief propagation as in Pearl's algorithm or EPIS permits propagation against the flow. It may be more natural to simulate following flow, but all of the "directions" can be reversed.*

We thank the reviewer for pointing this out and completely agree with the reviewer. We intended to differentiate our method to input-driven simulative approaches (like HSpice) that are input pattern sensitive that are common to core VLSI CAD community. We have rephrased that sentence. (see Page 18)

15. *Also on page 17, the last paragraph says, "This algorithm has been proven to converge to the correct probability estimates." Which algorithm? The prior sentence mentions both PLS and EPIS. Reference [24] seems to suggest it is PLS, but that is not the natural conclusion to be drawn from the sentence.*

    We thank the reviewer for pointing this out. We wanted to address both the algorithms. We have corrected that sentence. (see Page 19 Paragraph 2)

16. *On page 20, the authors state, "Yuan et al. proved that for a poly-tree, the local loopy belief propagation is optimal." That is not what they proved. First, loopy belief propagation is a generalization of Pearl's message passing algorithm that can already be applied to perform exact inference on a poly-tree. There is no need for importance sampling with a poly-tree. Yuan et al. are attempting to find the optimal importance function to be used in importance sampling for non-poly-tree networks. What they prove is that, should one want to apply importance sampling to a poly-tree network, then you can derive the optimal importance function for a poly-tree network and then use that in a loopy network with an importance-sampled variation of loopy belief propagation. The offending sentence needs to be rewritten to more accurately characterize the result from Yuan et al. and to explain why this result is even relevant to their discussion.*

    We agree with the reviewer. We thank the reviewer for pointing this out. We have removed that sentence from the discussion. (see Page 21 Paragraph 2)

17. *As a rule of thumb, providing algorithm comparisons based on run time is problematic*

and should be strongly discouraged. The primary reasons for this include the inability to control for variations in implementations, the effects of additional processes running on the experimental machine, and characteristics in architectures, languages, etc., that might effect the outcome of the experiment. In this paper, the authors attempt to compare the run time performance of their approach running on a 2 GHz P4 to the PTM algorithm written in C running on a 3 GHz P4. Their results show that their algorithm is much faster than PTM, even while running on a slower machine. They acknowledge that the LIPEM models were processed using GeNIe from the University of Pittsburgh, so they had no control over implementation. Unfortunately, this result is meaningless because we cannot tell whether the time improvements can be attributed to the algorithm or the implementation and run-time characteristics of the algorithm. Of course, the differences are pretty large, and those differences can be explained by an underlying complexity analysis, so the benchmarks become even less interesting. I suggest removing all of the time-based results since they add nothing to the discussion and tend to annoy experimental computer scientists like me. Either that or caveat the daylights out the results, explaining they are included to provide a frame of reference for a couple implementations.

We agree with the reviewer and acknowledge the comment. We have removed the table comparing the simulation time between PTM model and LIPEM model and instead added a theoretical discussion on the relative time and space complexity of both models and through this discussion we have shown the efficiency of our model (see pages. 23-24).

18. *Speaking of comparisons, since this paper is focusing on error estimates, a more useful comparison would have been between the error estimates of the LIPEM approach and PTM. Since the two approaches were used, it would have been nice to see the results of both rather than just focusing on the runtime performance of the two.*

TABLE XVI
COMPARISON OF THE RELIABILITY ESTIMATES OBTAINED FROM PTM MODEL AND LIPEM MODEL FOR GATE ERROR PROBABILITY $p = 0.05$

Circuit	Circuit reliability for individual gate error probability $p = 0.05$	
	LIPEM model	PTM model
c17	0.866	0.866
parity	0.603	0.603
pcle	0.844	0.844
decod	0.935	0.935
cu	0.803	0.803
pm1	0.737	0.737
xor5	0.566	0.566

We have added a table comparing the reliability estimates obtained from PTM model and LIPEM model for gate error probability $p = 0.05$(Table. XVI). These results are obtained by calculating the probability of correctness at the output. Since both models exactly capture the circuit structure, the results produced are similar to each other.

19. *I was also intrigued by the fact the authors used two different Bayesian network tools? GeNIe and Hugin. (By the way, GeNIe is the graphical front end to SMILE, so technically they used SMILE. This should probably be fixed.) It is apparent that they did this because GeNIe/SMILE does not include the junction tree algorithm where Hugin does. They need to point this out.*

We agree with the reviewer. We have used SMILE, which is the API version of GeNIe to inference large circuits, since HUGIN does not provide approximate inference algorithms. Even though GeNIe/SMILE provides the junction tree algorithm, HUGIN forms the junction tree with cliques that are much smaller than GeNIe. As we mentioned before, getting a junction tree is not the goal and would not provide computational leverage but junction tree is to be constructed properly such that number of nodes in the largest clique is much smaller than number of nodes in the graph. That's the reason for using HUGIN for exact inference. We have pointed this out in the beginning of the experimental results section. (see Page 23 Paragraph 1)