Scalable Probabilistic Computing Models using Bayesian Networks

Thara Rejimon and Sanjukta Bhanja
University of South Florida, Tampa, FL, USA.
E-mail: (rejimon, bhanja)@eng.usf.edu

Abstract

As technology scales below 100nm and operating frequencies Increase, correct operation of nano-CMOS will be compromised due reduced device-to-device distance, imperfections, and low noise and voltage margins. Unlike traditional faults and defects, these errors are expected to be transient in nature. Unlike radiation related upset errors, the propensity of these transient errors will be higher. Due to these highly likely errors, it is more appropriate to model nano-domain computing as probabilistic rather than deterministic events. We propose the formalism of probabilistic Bayesian networks (BNs), which also forms a complete joint probability model, for probabilistic computing. Using the exact probabilistic inference scheme known as clustering, we show that for a circuit with about 250 gates the output error estimation time is less than three seconds on a 2GHz processor. This is three orders of magnitude faster than a recently proposed method for probabilistic computing using transfer matrices.

1 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 arise due to temporary malfunction of nano-devices while operated near thermal limits. These errors are significant in nano-computing due to very low noise margin, reduced supply voltages and low stored charges in nodes. We term these errors as dynamic errors since they are not permanent damages. Such dynamic errors may occur anywhere in the circuit, but hard to detect by regular testing methodologies (since they are not permanent damage). They can be characterized only probabilistically. Each device (logic gate/interconnect) will have certain, non-zero, propensity for an output line error.

There have been works [1] that derived theoretical upper bounds of the output error, but without considering the specific circuit logic structure. Such general bounds can possibly used only for high architecture level designs issues, but better estimates are needed for logic level designs. A MATLAB based tool to compute and propagate probability distributions of fundamental gates is proposed in [7], which also evaluates reliability of defect tolerant architectures such as TMR and CTMR. However results were shown only for small circuits. Our model is used to 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.

Probabilistic computation with unreliable components is not new [1]. Recently, in [3] Chen *et. al* proposed Markov Random Field (MRF) based model which works only for toy circuits without re-convergent fanouts. Another primitive matrix based model proposed by Krishnaswami *et. al* [2], handles circuits that are really small with extremely high time complexity. In this work, we used BN based approach because logic circuits are inherently causal and a directed graph would be best to arrive at an edge minimal structure, the effect of which is evident in the observed time complexity of our model.

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

0-7803-9197-7/05/$20.00 © 2005 IEEE.

$Y_1, \cdots Y_K$. Let the corresponding versions of the internal lines and the 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 can be mathematically represented as the XOR of the error-free and the output with error.

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

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. For causal logical circuits, these dependencies can be modeled by a Bayesian Networks, which is known to be the exact, minimal probabilistic model for the underlying joint pdf. Probabilistic belief propagation on these Bayesian Networks can then to be used to estimate this probability.

2 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.

The exact joint probability distribution over a set of N random variables $X_1, X_2, \cdots, X_N$ in a Bayesian Network is given by Eq. 2.

$$\begin{aligned} P(x_1, \cdots, x_N) &= p(x_N|x_{N-1}, x_{N-2}, \cdots, x_1) \\ & \quad p(x_{N-1}|x_{N-2}, x_{N-3}, \cdots, x_1), \\ & \quad \cdots, p(x_1) \end{aligned} \tag{2}$$

Let X_{k-1} and X_{k-2} be the parents of a random variable X_k. Then X_k is independent of all other variables in the network given the states of its parent nodes, X_{k-1} and X_{k-2}. This *conditional independence* can be expressed by Eq. 3.

$$P(x_k|x_{k-1}, x_{k-2}, \cdots, x_1) = P(x_k|x_{k-1}, x_{k-2}) \tag{3}$$

Using the conditional independence in directional graph, we arrive at an optimal factorized form that involve conditional probabilities based on the parents (or direct causes, inputs) to a node (effect, output): $P(X) = \prod_{k=1}^{m} P(x_k|pa(x_k))$. Even though probabilistic inference is worst-case NP-Hard, these factorized forms can reduce complexity significantly for general cases.

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 specifically in switching model [4] and single stuck-at-fault/error model [5] in VLSI but their use in nano-domain dynamic-error modeling is new. We use an exact inference scheme, known as clustering [8] on the built Bayesian network representation to compute the output error probability.

3 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[1].

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

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

$$\begin{aligned} &= P(z_1) \cdots P(z_N) \\ & \quad \sum_{\forall z} \sum_{\forall x, \forall x^e} P(e_i, y_i, y_i^e, z_1, \cdot z_N, \\ & \quad x_1, \cdots x_M, x_1^e, \cdots x_M^e) \end{aligned} \tag{6}$$

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.

We model, both error-free logic and the one 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 output error probability, $P(E_i = 1)$ of output Y_i of the circuit.

Figure 1(a) shows the (conceptual) representation of a error detection circuit for a simple logic involving two NAND gates, represented by block C. The other block involves 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

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

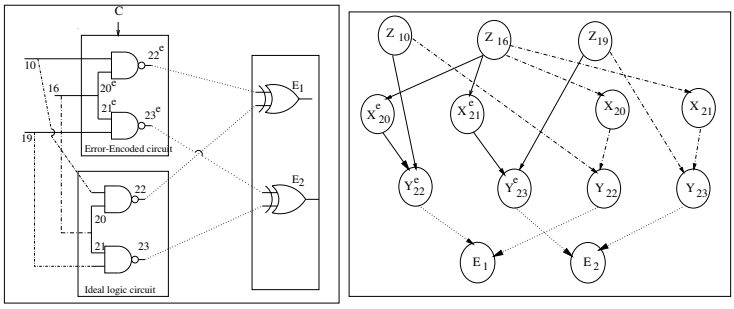

Figure 1. (a) Conceptual circuit representation of the logic used to detect errors involving the ideal logic and the unreliable logic components. (b) The corresponding Bayesian network representation.

output of the comparator represents error in computation. Note that this is just a conceptual representation, we do not actually propose synthesizing the circuit. From the conceptual circuit design, we can construct the Bayesian network representation, which we call the LIPEM-DAG model. Each node in the LIPEM is a line in circuit and the links denote a connection between the lines via gates. Figure 1(b) shows the LIPEM corresponding to the circuit in Figure 1(a). It can be easily proven that the LIPEM DAG structure corresponding to the combinational circuit C is a minimal I-map of the underlying dependency model and hence is a Bayesian network.

Bayesian Network Quantification: LIPEM-BN 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 $p(x_v|x_{\text{parent}(v_i)})$, where $parent(v_i)$ is the set of nodes that has directed edges to 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. By specifying a detailed conditional probability we ensure that the spatial dependencies among sets of nodes (not only limited pair-wise) are effectively modeled.

4 Computing the Error Probability

Junction Tree Based Inference: We demonstrate this inference scheme with an example shown In Fig 2. The combinational circuit is shown in Fig. 2a and Fig 2b is its equivalent Bayesian Network representation.

The steps involved in the exact inference scheme are described below. Moralization: Create an undirected graph structure called the *moral graph* from the Bayesian network DAG structure by adding undirected edges between the parents of a common child node and dropping the directions of the links. The moral graph represents the Markov structure of the underlying joint function [6]. The dependencies that are preserved in the original DAG are also preserved in the moral graph [6]. The dashed edges in Fig. 2c are added at this stage. This step ensures that every parent child set is a complete sub graph. Triangularization: In this step, every cycles of length greater than or equal to 4 is reduced to cycles of 3 by inserting additional links (chords) to the moral graph. The moral graph is said to be triangulated if it is chordal [6]. Note that in this particular example, moral graph is chordal and no additional links are needed. Message passing in Junction Tree: A *junction tree* is defined as a tree of cliques (collection of completely connected sub graph) of the choral graph (cliques are connected by unique path as in Fig 2c). Junction tree possesses running intersection property [6] 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. Fig. refbn1d is the junction tree derived from the chordal graph of Fig. 2c in this example. Interested readers are referred to [4] for a detailed description of how local message passing is performed in junction trees.

5 Experimental Results

We demonstrate the ideas using LGSynth'93 and some of the ISCAS'85 benchmark circuits. Gates with more than two inputs are reduced to two-input gates by introducing additional dummy nodes, without changing the circuit structure. While modeling the dummy nodes the gate-error probabilities of the dummy nodes are considered zero. Hence their presence do not modify the output error profiles.

In table 1, we report the maximum output error probabilities of benchmark circuits for different gate error probabilities. Column 2, 3 and 4 give the maximum output error probabilities when gate error probabilities are 0.005, 0.05 and 0.1 respectively. In column 5 we report the elapsed time.

In Table 2, we compare the time and space complexity of our model with those of Probabilistic Transfer Matrix [PTM] based method proposed in [2]. Column 2 and 3 of this table give the runtime and memory requirement

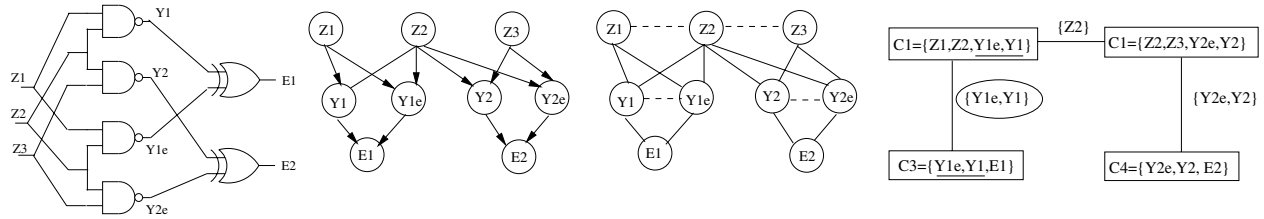

Figure 2. (a) A small Error Model (b) Bayesian Network representation (c) Chordal Graph (d) Junction Tree

Table 1. Output error probabilities.

	Maximum Output error Probability for individual gate error probability p			time(s)
	=0.005	=0.05	=0.1	
c17	0.0148	0.1342	0.2398	0.0
parity	0.0699	0.3971	0.4824	0.0001
pcle	0.0179	0.1560	0.2702	0.07
decod	0.0068	0.0654	0.1251	0.14
cu	0.0232	0.1969	0.3327	0.56
pm1	0.0331	0.2627	0.4141	0.44
xor5	0.0925	0.4336	0.4900	0.26
alu4	0.0676	0.3906	0.4816	1.87
b9	0.0315	0.2475	0.3867	2.49
comp	0.0733	0.3828	0.4683	0.66
count	0.0203	0.1613	0.2632	1.14
malu4	0.0845	0.4253	0.4903	1.94
max_flat	0.0296	0.2151	0.3234	0.02
pc	0.0377	0.2794	0.4161	0.41
voter	0.0299	0.2178	0.3294	0.08

Table 2. Comparison of Modeling using Bayesian Networks and Probabilistic Transfer Matrix

	PTM [2]		BN	
	time(s)	memory(MB)	time(s)	memory(MB)
c17	0.076	0.003	0.0	0.096
parity	0.35	0.144	0.0001	0.14
pcle	74.9	24.2	0.07	0.34
decod	56.9	11.8	0.14	0.42
cu	93.87	10	0.56	2.28
pm1	7169	160	0.44	1.38
xor5	1337	57.3	0.26	2.48

of PTM model, where simulations were performed using a 3GHz Pentium 4 processor. Column 4 and 5 give the runtime and memory requirement of our model. We used a 2GHz Pentium 4 processor and the listed elapsed times are obtained by the $ftime$ command in the WINDOWS environment, and is the sum of CPU, memory access and I/O time. These results show the effectiveness of our model in terms of estimation time and memory usage. The time and space complexity of BN based model does not depend on gate error probability values, whereas experimental results form [2] show that the runtime and memory requirement for PTM based modeling are not the same for different gate error probabilities.

6 Conclusion

We presented an exact probability model, based on Bayesian networks, to capture the inter-dependent effects of dynamic errors at each gate through the conditional probability specifications in the Bayesian network. We used an exact inference scheme to compute the overall output error probability due to individual gate errors. To handle larger circuits, we use an approximate inference scheme based on stochastic sampling which gave accurate estimates. Due to space limitation, we do not include those results in this paper. Among the other uses of the BN based probabilistic computing model, is (i) the calculation of the sensitivity of node errors on the output, which is useful for identifying gates for selective redundancy, and (ii) the characterization the input space in terms of propensity for errors, which has not been shown to be possible with the transfer matrix approach. 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.

References

[1] 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.

[2] 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.

[3] J. Chen, J. Mundy, Y. Bai, S.-M. C Chan, P. Petrica, and R. I. Bahar, "A Probabilistic Approach to Nano-computing," *Workshop on Non-Silicon Computation*, June 2003.

[4] S. Bhanja and N. Ranganathan, "Switching Activity Estimation of VLSI Circuits using Bayesian Networks" *IEEE Transactions on VLSI Systems*, pp. 558–567, Feb. 2003.

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

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

[7] D. Bhaduri and S. K. Shukla, "NANOLAB: A Tool for Evaluating Reliability of Defect-Tolerant Nano Architectures", *International Symposium on VLSI Design*, 2004.

[8] `URL http://www.hugin.com`