Probabilistic Maximum Error Modeling for Unreliable Logic Circuits

Karthikeyan Lingasubramanian
Department of Electrical Engineering
University of South Florida, Tampa, FL

Sanjukta Bhanja
Department of Electrical Engineering
University of South Florida, Tampa, FL

ABSTRACT

Reliability modeling and evaluation is expected to be one of the major issues in emerging nano-devices and beyond 22nm CMOS. Such devices would have inherent propensity for gate failures due to the underlying device variabilities. Many of these failures would be transient in nature, necessitating the need for probabilistic logic based analysis. Current research in this area is concerned with computing error bounds, but they do not account for circuits structures or are usually derived for specific logic gate types. In addition, the usual focus is on computing the *average* error behavior. In this work, we propose an exact probabilistic error model to compute the *maximum* error in a circuit-specific manner and can handle various types of logical components in the same circuit. We model the error estimation problem as a maximum *a posteriori* estimate (MAP) over the joint error probability function of the entire circuit. Using this model, we can not only compute the maximum error, but can also identify the input vector that cause the maximum output error. We demonstrate this model using MCNC and ISCAS circuits. We observe that for some circuits, maximum error probabilities are significantly larger than the average likelihood error, thus making a case for the consideration of maximum error metric as an essential design guideline rather than just average-case estimates. We also find that the error estimates depend on the specific circuit structure. Lastly, we observe that the maximum error probabilities are sensitive to the individual gate failure probabilities.

Categories and Subject Descriptors: C.4Performance of Systems:Reliability, availability, and serviceability

General Terms: Reliability

Keywords: Maximum error, MAP

1. INTRODUCTION

In this work, we present a formalism to study the *maximum* output error over all possible input space. This problem is essentially the Maximum *a posteriori* (MAP) estimate that maximizes the probability of evidence (in our case output error) for all possible primary inputs. As the first step in our model, we convert the circuit into a corresponding probabilistic model to represent the interdependence of the random variables of interest in a composite joint probability distribution function $P(Y_1, Y_2, \cdots, Y_N)$, where the graph structure models all structural dependencies in an edge-minimal fashion. The initial probabilistic model consists of three blocks, (i) ideal error free logic, (ii) logic network with erroneous gate and (iii) a detection unit that compares outputs from error-free and erroneous blocks for each output. Both the ideal logic and error free logic would be fed by the primary inputs **I**. We denote all the internal nodes, both in the error-free and erroneous portions, by **X** and the comparator outputs as **O**. The output logic is based on XOR and hence a value "1" would signify error at the output. The MAP problem estimates the maximum probability of individual output error $P(o_i)$ by sum-marginalizing the underlying joint probability distribution function over all the internal nodes **X** *and* maximizing over the primary inputs **I**.

The specific steps for the estimation process involves the conversion of the probabilistic model to a join tree(JT) through a series of transformations for local computation in presence of re-convergence. An upper bound of MAP probability is then obtained by message-passing between the neighboring clusters. We then obtain the exact MAP probability by a systematic depth-first search over the input instantiation tree and using the upper bound obtained from the join tree to branch and bound and prune some part of the search tree. We also can obtain the inputs that cause the maximum output error. We want to point out that since the probabilistic logic network will work in a stochastic manner, even with a fixed input vector, probability of output error will be non-zero and could exceed the expected error value for many cases. We study this behavior for various gate error probabilities.

2. PROBABILISTIC ERROR MODEL

The underlying model compares error-free and error-prone outputs. Our model contains three sections, (i) error-free logic where the gates are assumed to be perfect, (ii) error-prone logic where each gate goes wrong independently by an error probability ε and (iii) XOR-logic based comparators that compare between the error-free and error-prone outputs. These sections are modeled using a probabilistic network. Fig. 1 illustrates our model. In Fig. 1(a) block 1 is the error-free logic, block 2 is the error-prone logic with

GLSVLSI'07, March 11–13, 2007, Stresa-Lago Maggiore, Italy.

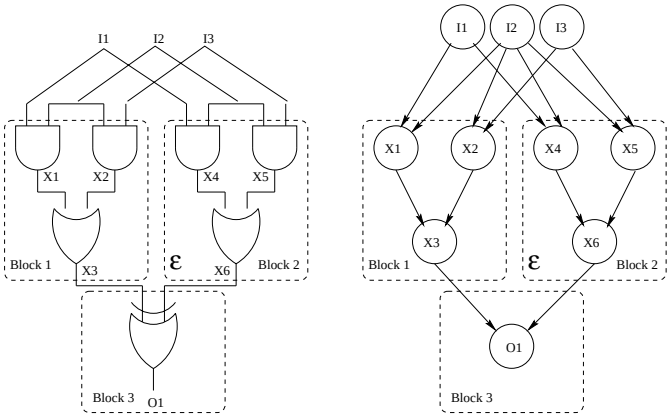

Figure 1: (a) Digital logic circuit (b) Corresponding probabilistic model

gate error probability ε and block 3 is the comparator logic. Fig. 1(b) represents the equivalent probabilistic network.

The probabilistic network is a conditional factoring of a joint probability distribution. The nodes in the network represent random variables. Let us define the following random variables say $\mathbf{Y}=\{\mathbf{I} \cup \mathbf{X} \cup \mathbf{O}\}$ are composed of the three disjoint subsets $\mathbf{I}$, $\mathbf{X}$ and $\mathbf{O}$ where each one of them have two states, logic state $''0''$ and logic state $''1''$ and where

1. $I_1, \cdots, I_k \in \mathbf{I}$ are the set of k primary inputs.
2. $X_1, \cdots, X_m \in \mathbf{X}$ are the internal logic signals for both the erroneous (every gate has a failure probability ε) and error-free ideal logic elements.
3. $O_1, \cdots, O_n \in \mathbf{O}$ are the n comparator outputs each one signifying the error in one of the primary outputs of the logic block.
4. $N = k+m+n$ are the total number of network random variables.

We would consistently denote smaller-case variables for instantiation of the Upper-case random variables. Any probability function $P(y_1, y_2, \cdots, y_N)$ can be written as[1]

$$\begin{aligned} P(y_1, \cdots, y_N) &= P(y_N|y_{N-1}, y_{N-2}, \cdots, y_1) \\ & \quad P(y_{N-1}|y_{N-2}, y_{N-3}, \cdots, y_1) \\ & \quad \cdots P(y_1) \end{aligned} \tag{1}$$

This expression holds for any ordering of the random variables. In most applications, a variable is usually not dependent on all other variables. There are lots of conditional independencies embedded among the random variables, which can be used to reorder the random variables and to simplify the conditional probabilities.

$$P(y_1, \cdots, y_N) = \prod_v P(y_v|Pa(Y_v)) \tag{2}$$

where $Pa(Y_v)$ are the parents of the variable Y_v, representing its direct causes. This factoring of the joint probability function can be denoted as a graph with links directed from the random variable representing the inputs of a gate to the random variable representing the output. The probability distribution of each random variable is given in a conditional probabilistic table based on the logic function which governs the signal. In this setup it is easier to incorporate the individual gate error probability ε by just changing the probabilities in the conditional probabilistic table. For example for a given gate error probability ε for a NAND gate, the conditional probability of the output equaling one (zero) given that the inputs are zero is $1-\varepsilon$ (ε).

[1]Probability of the event $Y_i = y_i$ will be denoted simply by $P(y_i)$ or by $P(Y_i = y_i)$.

3. MAXIMUM *A POSTERIORI* (MAP) ESTIMATE

The Maximum *a posteriori* estimate over a joint probability distribution function P, and three disjoint subsets of random variables $\mathbf{I}$, $\mathbf{X}$ and $\mathbf{O}$ is a way to find an instantiation $\mathbf{i}$ of variables in $\mathbf{I}$ that maximizes the $P(\mathbf{i}, \mathbf{o})$ given some evidence $\mathbf{o}$ in $\mathbf{O}$. Hence, MAP problem involves both summation (over $\mathbf{X}$) and maximization operators over($\mathbf{I}$), simultaneously. The average likelihood estimator works only to find the probability of evidence or $P(\mathbf{o})$, the maximum likelihood estimator estimates the maximum probability of $P(\mathbf{i}, \mathbf{x}, \mathbf{o})$.

The method to compute MAP probabilities, described in this section, is based on the works by Park and Darwiche [13] [14]. The reader is encouraged to refer [13] and [14] for more detailed explanation. Here we provide rough sketch of the computations. Again, we restate that probabilistic network variables say $\{Y\}$ can be divided into three subsets $\mathbf{I}$, $\mathbf{X}$ and $\mathbf{O}$. Let $I_1, \cdots, I_k \in \mathbf{I}$; $X_1, \cdots, X_m \in \mathbf{X}$; $O_1, \cdots, O_n \in \mathbf{O}$.

Consider that the variables in $\mathbf{I}$ are the MAP variables i.e., the variables whose most probable configuration $\mathbf{i}$ has to be found out, and we give evidence $\mathbf{o}$ to the variables in $\mathbf{O}$. The MAP estimate calculates the probability $MAP(\mathbf{i}, \mathbf{o})$.

$$MAP(\mathbf{i}, \mathbf{o}) = \max_{\mathbf{I}} \sum_{\mathbf{X}} P(\mathbf{i}, x_1, \cdots, x_m, \mathbf{o}) \tag{3}$$

For example, consider Fig 1. In the probabilistic model shown in Fig 1(b), $\{I1, I2, I3\} \in \mathbf{I}$; $\{X1, X2, X3, X4, X5, X6\} \in \mathbf{X}$; $\{O1\} \in \mathbf{O}$. For an evidence $\mathbf{o} = \{O1 = o_1\}$ the MAP will provide the instantiations $\mathbf{i} = \{I1 = i_1, I2 = i_2, I3 = i_3\}$.

The maximization and summation operators in Eq. 3 are non-commutative.

$$[\sum_{\mathbf{X}} \max_{\mathbf{I}} P](\mathbf{y}) \geq [\max_{\mathbf{I}} \sum_{\mathbf{X}} P](\mathbf{y}) \tag{4}$$

This is the basis for the first step of MAP evaluation. In this step a variable elimination algorithm is used which basically computes an upper bound on MAP probability. A valid elimination order of variables should have the summation variables first followed by the maximization variables. Subsequently an invalid elimination order will give the upper bound for MAP solution based on Eq. 4. Eventually, the closer the chosen invalid elimination order to the valid elimination order, the tighter will be the upper bound. This variable elimination mechanism can be incorporated using a jointree algorithm.

A jointree is a tree where individual nodes are clusters of variables of the basic probabilistic model shown in Fig 2b [12]. There are many algorithms for this transformation. Interested reader is referred to [12]. The neighboring clusters will have one or more common variables which enables message passing. With each cluster node, j, in the join tree we associate a function, $\phi(j)$, also termed as the probability

potential function, over the variables in the cluster node, constructed out of conditional probabilities in the probabilistic model (Eq. 5). For each conditional probability, $P(y_v|Pa(Y_v))$, we find one and only one cluster node, j, that contain the node set $\{y_v\} \cup Pa(Y_v)$. The potential function for a cluster node is the product of the conditional probability functions mapped to that cluster node. Thus,

$$\phi(j) = \prod_{\{y_v\} \cup Pa(Y_v) \in j} P(y_v|Pa(Y_v)) \quad (5)$$

The joint probability function, which was expressed as product of conditional probabilities, can now be expressed equivalently as the product of these individual ϕ potentials.

A message sent from a cluster j to cluster l can be described as,

$$M_{jl} = \max_M \sum_S \phi_j \prod_{k \neq l} M_{kj} \quad (6)$$

where $M \subseteq I$ and $S \subseteq X$ for all variables in cluster j and not in l. Given this we can say that for any cluster j the potential that contains the upper bound for the MAP solution can be given as,

$$\max_M \sum_S \phi_j \prod_k M_{kj} \quad (7)$$

where $M \subseteq I$ and $S \subseteq X$.

Once the join tree is formed, we designate a cluster as root cluster and with respect to that cluster, we find the leaf clusters. In the first phase, the MAP algorithm uses a message passing algorithm from the leaf to the root cluster. In this pass an upper bound of the MAP probability $\lceil P(\mathbf{i}, \mathbf{o}) \rceil$ is obtained for a given evidence. This is done by subsequently obtaining the upper bound $\lceil P(I \setminus I_i, \mathbf{o}, i_i) \rceil$ of the MAP probability by removing I_i from the maximizing variables (inputs) and adding the variable to the set of already known evidences.

In the next step, we create a binary search tree for the input instantiations. One path from the root to the leaf node of this search tree gives one input vector choice. At an internal node $\mathbf{i}$, we have partial instantiations of the maximizing input variables. Children of this node $\mathbf{i}$ contains the instantiations of $I \setminus \mathbf{i}$.

While performing the depth first search in the input tree, at every node $\mathbf{i}$, we perform $\lceil P(\mathbf{I} \setminus \mathbf{i}, \mathbf{o}, \mathbf{i}) \rceil$ using the message passing algorithm on JT as discussed above. If this upper bound is less or equal to the largest stored MAP probability, then the children of $\mathbf{i}$ are not considered further. The process is initialized by obtaining a MAP upper bound by performing a local search using hill climbing technique [14].

4. EXPERIMENTAL RESULTS

Our main goal is to provide the maximum output error probabilities for different gate error probabilities ε. To get the maximum output error probabilities every output signal of a circuit has to be examined through MAP estimation. First, an evidence has to be provided to one of the output signal variables in set $\mathbf{O}$ such that $P(o_i = 0) = 0$ and

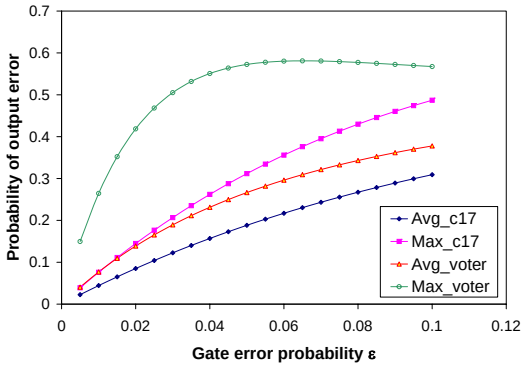

Figure 2: Sensitivity of average and maximum error probabilities with reference to the change in gate error probability ε for $c17$ and *voter*

$P(o_i = 1) = 1$. Recall that these variables have a probability distribution based on XOR logic and so giving an evidence like this is similar to forcing the output to be wrong. The outputs are evidenced individually and the corresponding input instantiations $\mathbf{i}$ are obtained by performing MAP. Then the input variables in the probabilistic model are instantiated with each instantiation $\mathbf{i}$ and inferenced to get the output probabilities. For performing MAP and inference in the probabilistic model, we use a tool called SAMIAM [2]. $P(o_i = 1)$ is noted from all the outputs for each $\mathbf{i}$ and the maximum value gives the maximum output error probability. In order to give a comparison we also present the average output error probabilities by providing the probability distribution, $P(i_i = 0) = 0.5$ and $P(i_i = 1) = 0.5$, to all the input variables. The entire operation is repeated for different ε values.

Our model also supports to have variable ε in a circuit. Each gate in a circuit can have an ε selected in random from a fixed range, say 0.005 - 0.1. We have presented the result in Fig. 3 for *max_flat*. Here we compare the average and maximum output error probability and run time for ε=0.005, ε=0.1 and variable ε ranging for 0.005 - 0.1. It can be seen that the output error probabilities for variable ε are closer to those for ε=0.1 than for ε=0.005.

Fig. 2 shows the change in both average and maximum error probabilities with reference to the change in gate error probability ε. These graphs are obtained by performing the experiment for different ε values ranging from 0.005 to 0.1. The notable result is that the maximum error probabilities are much larger than average error probabilities making them a very crucial design parameter to consider. We also can see that even for small circuits such as $c17$, the gap between the average and maximum error increases with ε (Fig. 2).

Table 1 tabulates the average and maximum output error probabilities with $\varepsilon = 0.05$ and $\varepsilon = 0.1$. The circuit *count* shows large difference between the average and maximum output error probability thereby indicating the importance of maximum error. Also based on average error, c17 would appear more damaging in terms of error tolerance than circuit *count* , where as considering maximum error, *count* appears more error-prone for vulnerable inputs with respect to c17. Table 1 also presents the run time for MAP computation. It is performed on a Windows PC with 2.26GHz Pentium 4 CPU with 1GB of RAM. The run time does not

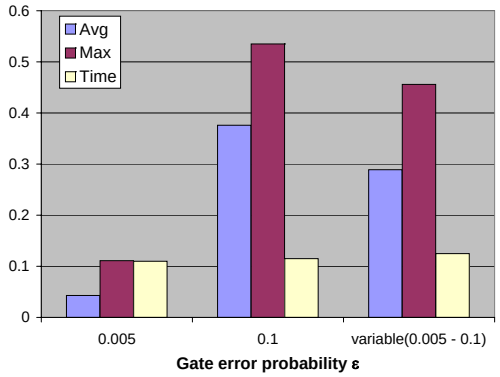

Figure 3: Comparison between the average and maximum output error probability and run time for ε=0.005, ε=0.1 and variable ε ranging for 0.005 - 0.1 for max_flat

Table 1: Average and maximum output error probabilities at $\varepsilon = 0.05$ and $\varepsilon = 0.1$

	ε=0.05		ε=0.1		Time
Circuit	Avg	Max	Avg	Max	
c17	0.188	0.312	0.309	0.487	0.047s
max_flat	0.272	0.457	0.376	0.535	0.110s
voter	0.266	0.573	0.378	0.568	0.641s
pc	0.299	0.533	0.43	0.589	225.297s
count	0.179	0.492	0.282	0.507	36.610s
alu4	0.472	0.517	0.5	0.604	58.626s
malu4	0.481	0.587	0.499	0.725	588.702s

change significantly for different ε and we show that in Fig. 3 where it can be clearly seen that the run times are very close to each other. This is expected as MAP complexity is determined by number of inputs, and number of variables in the largest cluster. Table 2 validates our proposed model by comparing it with an in-house simulator. It is clearly evident that the results from our model almost exactly coincides with the simulator results.

Table 3 gives the maximum-error input combinations got from MAP i.e., the input combinations that gives maximum output error probability. In max_flat and *voter* the maximum-error input vectors from MAP changes with ε, while in $c17$ it does not change. In the range {0.005-0.2} for ε, max_flat has three different maximum-error input combinations while *voter* has two. It implies that these maximum-error input combinations not only depend on the circuit structure but could dynamically change with the ε. This could be of concern for designers as the maximum-error inputs might change after gate error probabilities reduce due to error mitigation schemes. Hence, explicit MAP computation would be necessary to judge the maximum error probabilities after every redundancy schemes are applied.

Table 2: Comparison between Maximum output error probabilities achieved from the proposed model and the in-house simulator at $\varepsilon = 0.05$ and $\varepsilon = 0.1$

	ε=0.05		ε=0.1	
Circuit	Model	Simulation	Model	Simulation
c17	0.312	0.312	0.487	0.487
max_flat	0.457	0.457	0.535	0.535
voter	0.573	0.573	0.568	0.568
pc	0.533	0.535	0.589	0.604
count	0.492	0.495	0.507	0.516
alu4	0.517	0.521	0.604	0.609
malu4	0.587	0.592	0.725	0.732

Table 3: Maximum-error input combinations from MAP

Circuits	No. of Inputs	Input vector	Gate error probability ε
c17	5	01111	0.005 - 0.2
max_flat	8	00010011	0.005 - 0.025
		11101000	0.03 - 0.05
		11110001	0.055 - 0.2
voter	12	000100110110	0.01 - 0.19
		111011100010	0.2

5. CONCLUSION

We have proposed a probabilistic model that computes the exact maximum output error probabilities for a logic network and map this problem as maximum *a posteriori* hypothesis of the underlying joint probability distribution function of the network. We have demonstrated our model with standard ISCAS and MCNC benchmarks. The results show significant difference between the maximum and average output error probabilities. Using our model we can also compute the most probable input instantiations that can generate the maximum error probabilities. We also study the change in output error with respect to the individual gate error ε and the results show steady increase in output error.

6. 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] "Sensitivity Analysis, Modeling, Inference and More" URL http://reasoning.cs.ucla.edu/samiam/

[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] T. Rejimon and S. Bhanja, "Scalable Probabilistic Computing Models using Bayesian Networks", *IEEE Midwest Symposium on Circuits and Systems*, pp. 712-715, July 2005

[8] Z. Wang, K. Chakrabarty and M. Goessel, "Test Set Enrichment using a Probabilistic Fault Model and the Theory of Output Deviations", *Design, Automation and Test in Europe(DATE)*, pp. 1-6, 2006

[9] W. Evans and N. Pippenger, "On the Maximum Tolerable Noise for Reliable Computation by Formulas" *IEEE Transactions on Information Theory*, vol. 44-3 pp. 1299–1305, 1998.

[10] S. Roy and V. Beiu, "Majority Multiplexing-Economical Redundant Fault-tolerant Designs for Nano Architectures" *IEEE Transactions on Nanotechnology*, vol. 4-4 pp. 441–451, 2005.

[11] 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.

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

[13] J. D. Park and A. Darwiche, "Solving MAP Exactly using Systematic Search", *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence*, 2003.

[14] J. D. Park and A. Darwiche, "Approximating MAP using Local Search", *Proceedings of 17th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 403-410, 2001.