Probabilistic Error Propagation in Logic Circuits Using the Boolean Difference Calculus

Nasir Mohyuddin, Ehsan Pakbaznia and Massoud Pedram
University of Southern California
Department of Electrical Engineering
Los Angeles, CA, USA
E-mail: {mohyuddi,pakbazni,pedram}@usc.edu

Abstract **- A gate level probabilistic error propagation model is presented which takes as input the Boolean function of the gate, the signal and error probabilities of the gate inputs, and the gate error probability and produces the error probability at the output of the gate. The presented model uses the Boolean difference calculus and can be applied to the problem of calculating the error probability at the primary outputs of a multi-level Boolean circuit with a time complexity which is linear in the number of gates in the circuit. This is done by starting from the primary inputs and moving toward the primary outputs by using a post-order traversal. Experimental results demonstrate the accuracy and efficiency of the proposed approach compared to the other known methods for error calculation in VLSI circuits.**

I. INTRODUCTION

As CMOS hits nano-scale regime, reliability is fast becoming a first order design concern. In future, designers will have to build reliable circuits using unreliable gates [1][2]. Circuit reliability will thus be an important tradeoff factor which has to be taken care of similar to other tradeoff factors such as performance, area, power, etc. Different implementations of the same logic function result in circuits with different reliability factors. Therefore, similar to timing analyzer and power estimator tools, it is important to develop robust tools that are capable of estimating circuit reliability at different design levels. In [3] authors have proposed a probabilistic transfer matrix (PTM) method to calculate the output signal error probability for a circuit; however, in [4] the author has proposed a method based on the probabilistic decision diagrams (PDDs) to perform this task.

In this paper we first introduce a probabilistic gate level error propagation model based on the concept of Boolean difference to propagate errors from inputs to output of a gate. We then apply this model to propagate the error in a given circuit and finally estimate the error probability at the circuit outputs. Note in the proposed model each gate is considered to be a faulty gate whose error (if happens) can sometimes cancel out errors at the gate inputs. This is taken into account by considering the gate's truth-table and how the output can change if some of the gate's inputs and/or the gate itself are erroneous. Note the internal gate error is modeled as an output flipping event. This means that, when a faulty gate makes an error, it basically flips (changes 1 to 0 or vise versa) its output value that it is supposed to generate given the inputs. To our knowledge, this is the first time that Boolean difference calculus is used to calculate the output error probability of a combinational gate in the presence of multiple input signal errors and the internal gate error itself. Reference [5] has used Boolean difference to estimate the switching activity of Boolean circuits when multiple inputs switch, but error estimation using Boolean difference in the presence of multiple faults is more involved due to the fact that certain combinations of faults may cancel each other. In the rest of the paper, we call our circuit error estimation technique the *Boolean Difference-based Error Calculator,* or BDEC for short. In BDEC we assume that a defective logic gate produces the wrong output value for every input combination. This is a more pessimistic defect model than the stuck-at-fault model.

Authors in [3] use a matrix called PTM for each gate to represent the error propagation from the inputs to the output(s) of a gate. They then define some operations such as matrix multiplication and tensor product to use PTMs for different gates in order to generate and propagate error probability for different nodes in a circuit level-by-level. Despite of its accuracy in calculating signal error probability, PTM technique suffers from the extremely large amount of the computations (regular and tensor matrix products) which makes this technique both very memory-intensive and slow. For larger circuits, size of the PTM matrices at each circuit level becomes very large making PTM an inefficient or even infeasible technique for error rate calculation. References [6] and [7] developed a methodology based on probabilistic model checking (PMC) to evaluate the circuit reliability. The issue of excessive memory requirement of PMC when the circuit size is large was successfully addressed in [8]. However, the time complexity stills remains a problem. In fact, the authors of [8] show that the run time for their more space-efficient approach is even worse than that of the original approach.

Boolean difference calculus was introduced and used for analyzing errors in the circuits by [9] [10] but it was mostly used to analyze single faults. It was then extended by [11][12] to handle multiple fault situations, however, they only consider stuck-at-faults. In [13] authors use Bayesian

networks to calculate the output error probabilities without considering the input signal probabilities.

The author in [4] uses probabilistic decision diagrams (PDD) to calculate the error probabilities at the outputs using probabilistic gates. While PDDs are on average much more efficient than PTM, the worst-case complexity of both PTM and PDD-based error calculators is exponential in the number of inputs in the circuit.

In contrast, we will show that BDEC calculates the circuit error probability much faster than what PTM does while achieving as accurate results as PTM's. We will show that BDEC requires a single pass over the circuit nodes using a post-order (reverse DFS) traversal to calculate the errors probabilities at the output of each gate as we move form the primary inputs to the primary outputs, hence complexity is *O(N)* where *N* is the number of the gates in the circuit.

The rest of the paper is organized as follows. In Section II we provide some important notations and definitions used in the rest of the paper. Section III explains the proposed gate error model and develops a technique to calculate the circuit reliability using this model. Section IV discusses the practical considerations. Section IV shows the simulation results and Section VI concludes the paper.

II. PRELIMINARIES

Some key concepts and notation that will be used in the remainder of this paper are discussed in this section.

A. Partial Boolean difference

The partial Boolean difference of function $f(x_1, x_2, ..., x_n)$ with respect to subsets of its variables [12] is defined as:

$$\frac{\partial f}{\partial x_i} = f_{x_i} \oplus f_{\bar{x}_i}$$
$$\frac{\partial f}{\partial x_{i_1} x_{i_2} \ldots x_{i_k}} = \frac{\partial f}{\partial \left(x_{i_1} x_{i_2} \ldots x_{i_k} \right)} = f_{x_{i_1} x_{i_2} \ldots x_{i_k}} \oplus f_{\bar{x}_{i_1} \bar{x}_{i_2} \ldots \bar{x}_{i_k}} \quad (1)$$

where $\oplus$ represents XOR operator and f_{x_i} is the co-factor of $f(.)$ with respect to x_i, i.e.,

$$f_{x_i} = f(x_1, \ldots, x_{i-1}, x_i = 1, x_{i+1}, \ldots, x_n)$$
$$f_{\bar{x}_i} = f(x_1, \ldots, x_{i-1}, x_i = 0, x_{i+1}, \ldots, x_n) \quad (2)$$

Higher order co-factors of $f(.)$ can be defined similarly. Partial Boolean difference of the Boolean function f with respect to a single variable, x_i, returns the condition (on the rest of the variables) under which function f is sensitive to the input variable x_i. More precisely, if values of other input variables, other than x_i, are such that $\partial f/\partial x_i=1$, then by changing the input value x_i, the output value of function f will change. Whereas if values of input variables other than x_i are such that $\partial f/\partial x_i=0$, changing the input value x_i will not affect the output value of the function f.

Note the order-k partial Boolean difference defined in (1) is different from the k^{th} Boolean difference of function f as used in [11], which is denoted by $\partial^k f / \partial x_{i_1} \ldots \partial x_{i_k}$. For example, the 2nd Boolean difference of function f with respect to x_i and x_j is defined as:

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j} \right) = f_{x_i x_j} \oplus f_{\bar{x}_i x_j} \oplus f_{x_i \bar{x}_j} \oplus f_{\bar{x}_i \bar{x}_j} \quad (3)$$

Therefore, $\partial^2 f/\partial x_i \partial x_j \neq \partial f/\partial(x_i x_j)$.

B. Total Boolean difference

Similar to the partial Boolean difference that shows the condition under which a Boolean function is sensitive to change of any of its input variables, we can define *total Boolean difference* showing the conditions under which the output of the Boolean function $f(x_1, x_2, ..., x_n)$ is sensitive to the *simultaneous* changes in all the variables of a subset of input variables. For example, the total Boolean difference of function f with respect to $x_i x_j$ is defined as:

$$\frac{\Delta f}{\Delta(x_i x_j)} = \frac{\partial f}{\partial(x_i x_j)}\left(x_i x_j + \bar{x}_i \bar{x}_j \right) + \frac{\partial f}{\partial(\bar{x}_i x_j)}\left(\bar{x}_i x_j + x_i \bar{x}_j \right) \quad (4)$$

Note $\partial f/\partial(x_i x_j)$ describes conditions whereby if we go from $x_i=x_j=1$ to $x_i=x_j=0$ and vice versa, then the value of function f will be guaranteed to change. In contrast $\Delta f/\Delta(x_i x_j)$ describes conditions whereby if x_i and x_j change simultaneously (we go from $x_i=x_j=1$ to $x_i=x_j=0$ and vice versa OR we go from $x_i=1$, $x_j=0$ to $x_i=0$, $x_j=1$ and vice versa), then the value of function f will be guaranteed to change.

It can be shown that the total Boolean difference shown in (4) can be written in the following form:

$$\frac{\Delta f}{\Delta\left(x_i x_j \right)} = \frac{\partial f}{\partial x_i} \oplus \frac{\partial f}{\partial x_j} \oplus \frac{\partial^2 f}{\partial x_i \partial x_j} \quad (5)$$

Total Boolean difference with respect to three variables is:

$$\begin{aligned} \frac{\Delta f}{\Delta(x_1 x_2 x_3)} = & \frac{\partial f}{\partial(x_1 x_2 x_3)}\left(x_1 x_2 x_3 + \bar{x}_1 \bar{x}_2 \bar{x}_3 \right) \\ & + \frac{\partial f}{\partial(x_1 x_2 \bar{x}_3)}\left(x_1 x_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 x_3 \right) \\ & + \frac{\partial f}{\partial(x_1 \bar{x}_2 x_3)}\left(x_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 \bar{x}_3 \right) \\ & + \frac{\partial f}{\partial(\bar{x}_1 x_2 x_3)}\left(\bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 \bar{x}_3 \right) \end{aligned} \quad (6)$$

It is straightforward to verify that:

$$\begin{aligned} \frac{\Delta f}{\Delta(x_1 x_2 x_3)} = & \frac{\partial f}{\partial x_1} \oplus \frac{\partial f}{\partial x_2} \oplus \frac{\partial f}{\partial x_3} \oplus \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ & \oplus \frac{\partial^2 f}{\partial x_2 \partial x_3} \oplus \frac{\partial^2 f}{\partial x_1 \partial x_3} \oplus \frac{\partial^3 f}{\partial x_1 \partial x_2 \partial x_3} \end{aligned} \quad (7)$$

Higher order total Boolean differences may be defined similarly. In general total Boolean difference of function f with respect to an n-variable subset of its inputs can be written as:

$$\frac{\Delta f}{\Delta(x_{i_1} x_{i_2} \ldots x_{i_n})} = \sum_{j=0}^{2^{n-1}-1} \left. \frac{\partial f}{\partial \vec{x}} \right|_{m_j} \left(m_j + m_{2^n - j - 1} \right) \quad (8)$$

where m_j's are defined as follows:

$$
\begin{aligned}
m_0 &= \bar{x}_{i_1}\bar{x}_{i_2}\ldots\bar{x}_{i_{n-1}}\bar{x}_{i_n}\\
m_1 &= \bar{x}_{i_1}\bar{x}_{i_2}\ldots\bar{x}_{i_{n-1}}x_{i_n}\\
&\vdots\\
m_{2^n-1} &= x_{i_1}x_{i_2}\ldots x_{i_{n-1}}x_{i_n},
\end{aligned}
\tag{9}
$$

and we have:

$$
\left.\frac{\partial f}{\partial \vec{x}}\right|_{m_j} = \frac{\partial f}{\partial\left(x_1^* x_2^* \ldots x_{n-1}^* x_n^*\right)} \text{ where } m_j = x_1^* x_2^* \ldots x_{n-1}^* x_n^* \tag{10}
$$

C. Signal and error probabilities

Signal probability is defined as the probability for a signal value to be "1". That is:

$$
p_i = \Pr\{x_i = 1\} \tag{11}
$$

Gate error probability is shown by ε_g and is defined as the probability that a gate generates an erroneous output, *independent* of its applied inputs. Signal error probability is defined as the probability of error on a signal line. If the signal line is output of a gate, the error can be either due to input gate error or gate error itself. We show the error probability on signal line x_i by ε_i.

We are interested in determining the circuit output error rates, given the circuit input error rates under the assumption that each gate in the circuit can fail independently with a probability of ε_g. We account for multiple simultaneous gate failures which is the general case.

III. Proposed Error Model

In this section we propose our gate error model based on the Boolean difference calculus. The gate error model is then used to calculate the error probability and reliability at outputs of a circuit.

A. Gate Error Model

Figure 1 shows a logic gate realizing Boolean function f, with gate error probability of ε_g. The signal probabilities at the inputs, i.e., probabilities for input signals being 1, are p_1, p_2,…, p_n while the input error probabilities are ε_1, ε_2,…, ε_n. The output error probability is ε_z.

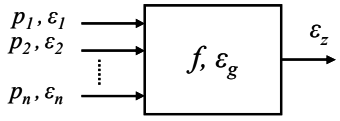

Figure 1. Gate implementing function f.

First consider the error probability equation for a buffer gate shown in Figure 2. The error at the output occurs if the input is erroneous and the gate is not or when the gate is erroneous and the input is not. Therefore, assuming independent faults for the input and the gate, the output error probability for buffer can be written as:

$$
\varepsilon_z = \varepsilon_{in}(1-\varepsilon_g) + (1-\varepsilon_{in})\varepsilon_g = \varepsilon_g + (1-2\varepsilon_g)\varepsilon_{in} \tag{12}
$$

where, ε_{in} is the error probability at the buffer input. It can be seen from the equation above that the output error probability for buffer is independent of the input signal probability. Note (12) can also be used to express the output error probability of an inverter gate.

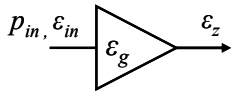

Figure 2. A faulty buffer with erroneous input.

Now we model each faulty gate with erroneous inputs as an ideal (no fault) gate with the same functionality and the same inputs in series with a faulty buffer as shown in Figure 3.

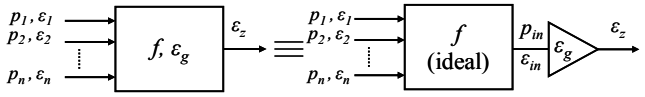

Figure 3. The proposed model for a general faulty gate.

Now consider a general two-input gate. Using the fault model discussed above, we can write the output error probability considering all the cases of no error, single error and double errors at the input and the error in the gate itself. We can write the general equation for the error probability at the output, ε_z, as:

$$
\varepsilon_z = \varepsilon_g + (1-2\varepsilon_g)\underbrace{\left(\begin{array}{l}\varepsilon_1(1-\varepsilon_2)\Pr\left\{\dfrac{\partial f}{\partial x_1}\right\} + (1-\varepsilon_1)\varepsilon_2\Pr\left\{\dfrac{\partial f}{\partial x_2}\right\}\\ +\varepsilon_1\varepsilon_2\Pr\left\{\dfrac{\Delta f}{\Delta(x_1x_2)}\right\}\end{array}\right)}_{\varepsilon_{in}} \tag{13}
$$

where $\Pr\{.\}$ represents the signal probability function and returns the probability of its Boolean argument to be "1". The first and the second terms in ε_{in} account for the error at the output of the ideal gate due to single input errors at the first and the second inputs, respectively. Note error at each input of the ideal gate propagates to the output of this gate only if the other inputs are not masking it. The non-masking probability for each input error is taken into account by calculating the signal probability of the partial Boolean difference of the function f with respect to the corresponding input. The first two terms in ε_{in} only account for the cases when we have single input errors at the input of the ideal gate, however, error can also occur when both inputs are erroneous simultaneously. This is taken into account by multiplying the probability of having simultaneous errors at both inputs, i.e., $\varepsilon_1\varepsilon_2$, with the probability of this error to be propagated to the output of the ideal gate, i.e., the signal probability of the total Boolean difference of f with respect to x_1x_2.

For 2-input AND gate ($f=x_1x_2$) of Figure 4 we have:

$$
\begin{aligned}
&\Pr\left\{\frac{\partial f}{\partial x_1}\right\} = \Pr\{x_2\} = p_2,\ \Pr\left\{\frac{\partial f}{\partial x_2}\right\} = \Pr\{x_1\} = p_1\\
&\Pr\left\{\frac{\Delta f}{\Delta(x_1x_2)}\right\} = \Pr\{\bar{x}_1\bar{x}_2 + x_1x_2\} = (1-p_1)(1-p_2) + p_1p_2\\
&\qquad = 1-(p_1+p_2)+2p_1p_2
\end{aligned}
\tag{14}
$$

Plugging (14) into (13) and after some simplifications, we obtain:

$$\varepsilon_{AND2} = \varepsilon_g + (1-2\varepsilon_g)\left(\varepsilon_1 p_2 + \varepsilon_2 p_1 + \varepsilon_1\varepsilon_2\left(1-2\left(p_1+p_2\right)+2p_1p_2\right)\right) \quad (15)$$

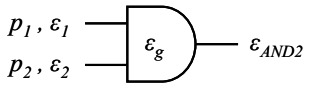

Figure 4. Faulty 2-input AND gate with erroneous inputs.

Similarly, the error probability for the case of 2-input OR can be calculated as:

$$\varepsilon_{OR2} = \varepsilon_g + (1-2\varepsilon_g)\left(\varepsilon_1\left(1-p_2\right)+\varepsilon_2\left(1-p_1\right)+\varepsilon_1\varepsilon_2\left(2p_1p_2-1\right)\right) \quad (16)$$

And for 2-input XOR gate we have:

$$\varepsilon_{XOR2} = \varepsilon_g + \left(1-2\varepsilon_g\right)\left(\varepsilon_1+\varepsilon_2-2\varepsilon_1\varepsilon_2\right) \quad (17)$$

It is interesting to note that the error probability at the output of the XOR gate is independent of the input signal probabilities. Everything else being equal, the 2-inpout XOR gate exhibits larger output error compared to 2-input OR and AND gates. This is expected since XOR gates show maximum sensitivity to input errors (XOR, like inversion, is an entropy preserving function). The output error expression for a 3-input gate is:

$$\varepsilon_z = \varepsilon_g + (1-2\varepsilon_g)\begin{pmatrix} \varepsilon_1(1-\varepsilon_2-\varepsilon_3+\varepsilon_2\varepsilon_3)\Pr\left\{\frac{\partial f}{\partial x_1}\right\} \\ +\varepsilon_2(1-\varepsilon_1-\varepsilon_3+\varepsilon_1\varepsilon_3)\Pr\left\{\frac{\partial f}{\partial x_2}\right\} \\ +\varepsilon_3(1-\varepsilon_1-\varepsilon_2+\varepsilon_1\varepsilon_2)\Pr\left\{\frac{\partial f}{\partial x_3}\right\} \\ +\varepsilon_1\varepsilon_2(1-\varepsilon_3)\Pr\left\{\frac{\Delta f}{\Delta(x_1x_2)}\right\} \\ +\varepsilon_2\varepsilon_3(1-\varepsilon_1)\Pr\left\{\frac{\Delta f}{\Delta(x_2x_3)}\right\} \\ +\varepsilon_1\varepsilon_3(1-\varepsilon_2)\Pr\left\{\frac{\Delta f}{\Delta(x_1x_3)}\right\} \\ +\varepsilon_1\varepsilon_2\varepsilon_3\Pr\left\{\frac{\Delta f}{\Delta(x_1x_2x_3)}\right\} \end{pmatrix} \quad (18)$$

And a general expression for a 4-input logic gate is:

$$\varepsilon_z = \varepsilon_g + (1-2\varepsilon_g)\begin{pmatrix} \sum_i \varepsilon_i\left(1-\sum_{j\neq i}\varepsilon_j+\sum_{(j,k)\neq i}\varepsilon_j\varepsilon_k-\sum_{(j,k,l)\neq i}\varepsilon_j\varepsilon_k\varepsilon_l\right)\Pr\left\{\frac{\partial f}{\partial x_i}\right\} \\ +\sum_{(i,j)}\varepsilon_i\varepsilon_j\left(1-\sum_{k\neq i,j}\varepsilon_k+\sum_{(k,l)\neq i,j}\varepsilon_k\varepsilon_l\right)\Pr\left\{\frac{\Delta f}{\Delta(x_ix_j)}\right\} \\ +\sum_{(i,j,k)}\varepsilon_i\varepsilon_j\varepsilon_k\left(1-\sum_{l\neq i,j,k}\varepsilon_l\right)\Pr\left\{\frac{\Delta f}{\Delta(x_1x_2x_3)}\right\} \\ +\varepsilon_1\varepsilon_2\varepsilon_3\varepsilon_4\Pr\left\{\frac{\Delta f}{\Delta(x_1x_2x_3x_4)}\right\} \end{pmatrix} \quad (19)$$

The general expression for the output error probability for a k-input gate can be easily written in the similar fashion. However, since this general form is too lengthy, we do not provide it in this paper.

B. Circuit Error Model

In this section we use the gate error model proposed in sub-section A to calculate the error probability at the output of a given circuit.

Given a multi-level logic circuit composed of logic gates, we start from the primary inputs and move toward the primary outputs by using a post-order (reverse DFS) traversal. For each gate, we calculate the output error probability using input signal probabilities, input error probabilities, and gate error probability and utilizing the error model proposed in sub-section A. The signal probability for the output of each gate is also calculated based on the input signal probabilities and the gate function. The process of output error and signal probability calculation is continued until all gates are visited. For each node z in the circuit, reliability is defined as:

$$\chi_z = 1-\varepsilon_z \quad (20)$$

After processing all the gates in the circuit and calculating error probabilities and reliabilities for all the circuit primary outputs, we can calculate the overall circuit reliability. Assuming that different primary outputs of the circuit are independent, the overall circuit reliability can be calculated as the product of all the primary outputs reliabilities, that is:

$$\chi_{circuit} = \prod_i \chi_{PO_i} \quad (21)$$

The case of dependent primary outputs (which is a more realistic scenario) requires calculation of spatial correlation coefficient as will be outlined further on in the paper. The detailed treatment of spatial correlation coefficient calculation however falls outside the scope of the present work

This error propagation algorithm has $O(2^kN)$ complexity where k is the maximum number of inputs to any gate in the circuit (which is small and can be upper bounded *a priori* in order to give $O(N)$ complexity) and N is the number of gates in the circuit. This complexity should be contrasted to that of the PTM based or the PDD-based approaches, that have a worst case complexity of $O(2^N)$. The tradeoff is that our proposed approach based on post-order traversal of the circuit netlist and application of Boolean difference operator results in only approximate output error and signal probability values due to the effect of re-convergent fanout structures in the circuit, which create *spatial correlations* among input signals to a gate. This problem has been addressed in the literature on improving the accuracy of signal probability calculators [14][15]. Our future implementation of BDEC shall focus on utilizing similar techniques (including efficient calculation of spatial correlation coefficients) to improve the accuracy of proposed Boolean difference-based error calculation engine.

IV. Practical Considerations

A. Output error expression

In this section we use the error models introduced in previous section to calculate the exact error probability expression at the output of a tree-structured circuit. For the sake of

elaboration, we choose a 4-input AND gate implemented as a balanced tree of 2-input AND gates as shown in Figure 5. We can calculate the output error of this circuit by expressing the error at the output of each gate using (15).

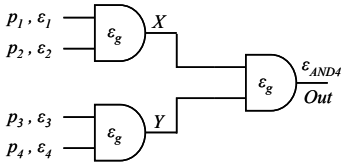

Figure 5. Balanced tree implemenation of 4-input AND gate.

Equation (22) provides the exact output error probability of the circuit shown in Figure 5 in terms of the input signal probabilities and input error probabilities where, similar to [16], high order exponents of the signal probabilities are reduced to first order exponents (this is called *high order exponent suppression* and is essential in order to avoid duplicate counting of terms in the final expression). In (22), without loss of generality, we assume $\varepsilon_{g=0}$ in order to reduce the length of the expression.

$$
\begin{aligned}
\varepsilon_{AND4} = {} & (p_2p_3p_4\varepsilon_1 + p_1p_3p_4\varepsilon_2 + p_1p_2p_4\varepsilon_3 + p_1p_2p_3\varepsilon_4) \\
& + p_3p_4(1-2(p_1+p_2)+2p_1p_2)\varepsilon_1\varepsilon_2 \\
& + p_2p_4(1-2(p_1+p_3)+2p_1p_3)\varepsilon_1\varepsilon_3 \\
& + p_2p_3(1-2(p_1+p_4)+2p_1p_4)\varepsilon_1\varepsilon_4 \\
& + p_1p_4(1-2(p_2+p_3)+2p_2p_3)\varepsilon_2\varepsilon_3 \\
& + p_1p_3(1-2(p_2+p_4)+2p_2p_4)\varepsilon_2\varepsilon_4 \\
& + p_1p_2(1-2(p_3+p_4)+2p_3p_4)\varepsilon_3\varepsilon_4 \\
& + p_4(1-2(p_1+p_2+p_3)+4(p_1p_2+p_1p_3+p_2p_3)-6p_1p_2p_3)\varepsilon_1\varepsilon_2\varepsilon_3 \\
& + p_2(1-2(p_1+p_3+p_4)+4(p_1p_3+p_1p_4+p_3p_4)-6p_1p_3p_4)\varepsilon_1\varepsilon_3\varepsilon_4 \\
& + p_3(1-2(p_1+p_2+p_4)+4(p_1p_2+p_1p_4+p_2p_4)-6p_1p_2p_4)\varepsilon_1\varepsilon_2\varepsilon_4 \\
& + p_1(1-2(p_2+p_3+p_4)+4(p_2p_3+p_2p_4+p_3p_4)-6p_2p_3p_4)\varepsilon_2\varepsilon_3\varepsilon_4 \\
& + \begin{pmatrix} 1-2(p_1+p_2+p_3+p_4) \\ +4(p_1p_2+p_1p_3+p_1p_4+p_2p_3+p_2p_4+p_3p_4) \\ -8(p_1p_2p_3+p_1p_2p_4+p_1p_3p_4+p_2p_3p_4) \\ +14p_1p_2p_3p_4 \end{pmatrix} \varepsilon_1\varepsilon_2\varepsilon_3\varepsilon_4
\end{aligned}
\qquad (22)
$$

Using symbolic formula with high order exponent suppression, the model presented in section III computes the exact output error probability in circuits with no reconvergent fanout. In our experimental results, we resort to signal error propagation using numerical values instead of the symbolic notation. Consequently, the computational complexity of error computation for a circuit becomes a linear function of the number of gates in that circuit. This, of course, comes at the expense of introducing some inaccuracy in our computational model. However, the exponential improvement in time complexity justifies the small loss in accuracy.

B. Reconvergent Fanout

Figure 6 is an example of a circuit with reconvergent fanout. It is clear from the figure that inputs to the final logic gate are not independent. Therefore, if the BDEC technique discussed in Section III is applied to this circuit, the calculated output error probability will not be accurate. We describe a modification to BDEC which improves the probability of error calculation in a circuit with reconvergent fanouts.

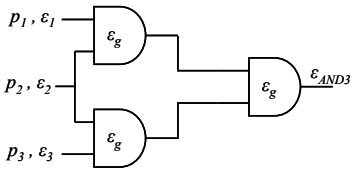

Figure 6. Balanced tree implemenation of 3-input AND gate.

Local reconvergent fanout like the one depicted in Figure 6 can be handled by collapsing levels of logic. This means that we consider all the three gates in Figure 6 as a single circuit block and then apply the BDEC technique for this circuit block. In this case the BDEC technique only sees a 3-input AND function without having any information about how this 3-input AND function has been implemented. Additionally, the original implementation information can be taken into account by properly calculating the ε_g value for this new 3-input AND block. The ε_g value for the collapsed gate is calculated assuming that the input error probabilities are zero.

Table I shows that BDEC + logic collapsing produces accurate results for the circuit in Figure 6. If the reconvergent fanout extends over multiple circuit levels, then multiple level collapsing can be used. Note that as the number of collapsed levels gets larger, the computational complexity of computing output error probability of a super gate with many inputs becomes higher and a trade-off between accuracy and complexity will have to be made.

TABLE I. OUTPUT ERROR PROBABILITY WITH RE-CONVERGENT FANOUT

Circuit	PTM	BDEC	BDEC+ Collapsing
3-Input AND Gate	0.1092	0.1121	0.1092

V. SIMULATION RESULTS

In this section we present some simulation results for the proposed circuit reliability technique and we compare the results of our approach with those of PTM and PGM [17]. For simplicity, but without loss of generality, we assume all gates in a circuit have the same gate error probability ε_g. All primary inputs are assumed to be error free and spatiotemporally uncorrelated. Moreover, signal probability for all the inputs were set to 0.5. The gate error probability was set to 0.05.

We implemented the proposed error calculator and algorithm (BDEC) in SIS [18]. We thus present results that show how efficiently BDEC can calculate the output reliability for circuits with high primary input count. Running our MATLAB 7.1-based implementation of PTM on a computer system with 2GBytes of RAM, we observed that typically for circuits with 16 or more inputs, PTM reported out of memory error. BDEC, however, does the calculations much faster and more efficiently than PTM.

TABLE II shows results for reliability calculation in tree-structured circuits. Note that "8-Input XOR BT" (BT for Balanced Tree) refers to 8-input XOR function implemented

using 2-input XOR gates in three levels of logic whereas "8-Input XOR Chain" refers to the same function realized as a linear chain of seven 2-input XOR gates. We also show results for two 16-input circuits with balanced tree implementation of 2-input gates having layers of 2-input AND, OR or XOR gates. First letter of gate name is used to show the gates used in each level. For example, AOXO means that the circuit consists of four levels of logic with AND, OR, XOR and OR gates at the first thru the fourth logic level.

Since the complexity of the PTM approach increases exponentially with primary input count, all circuits in TABLE II are chosen to have relatively small number of primary inputs. It is seen that the BDEC technique achieves highly accurate reliability values, i.e., the reliability values are different than PTM ones by at most 0.1% for the circuits reported in TABLE II. More importantly, TABLE II shows the difference between the scaling trend of the execution time in both PTM and BDEC techniques. In PTM, the execution time increases exponentially when we move from smaller circuits to larger circuits in TABLE II, whereas in BDEC the change in the execution time when we move from smaller circuits to the larger ones in TABLE II is really small. For two cases, 16-input XOR chain and 16-input AND chain, the system runs out of memory while executing PTM technique. This shows that execution of PTM technique for even relatively small circuits needs a huge amount of system memory.

TABLE II. CIRCUIT RELIABILITY FOR TREE-STRUCTURED CIRCUITS HAVING RELATIVELY SMALL NUMBER OF PRIMARY INPUTS.

Circuit	# of Gates	Execution Time (ms)		Circuit Reliability	
		PTM	BDEC	PTM	BDEC
8-Input XOR BT	7	0.790	0.011	0.7391	0.7391
16-Input XOR BT	15	1664.5	0.017	0.6029	0.6029
16-Input XOR Chain	15	Out of Memory	0.015	Out of Memory	0.6029
8-Input AND BT	7	0.794	0.010	0.9392	0.9382
16-Input AND BT	15	1752.2	0.017	0.9465	0.9462
16-Input AND Chain	15	Out of Memory	0.016	Out of Memory	0.9091
16-input AOXO BT	15	1769.3	0.017	0.7622	0.7616
16-input OXAX BT	15	1593.1	0.017	0.7361	0.7361

Another important advantage of the proposed BDEC technique which can be observed from TABLE II is that the complexity of this technique mainly depends on the number of the gates in the circuit; however, the complexity of PTM technique depends on several other factors such as number of the inputs, width and depth of the circuit, number of the wire crossovers, etc. In other words, efficiency (execution time and memory usage) of PTM depends not only on the number of the gates in the circuit, but on the circuit topology. This is a big disadvantage for PTM making it an infeasible solution for large and/or topologically complex circuits.

Although the complexity of Boolean difference equations increases exponentially with the number of the inputs of the function; this does not increase the complexity of the BDEC technique. The reason is the fact that using gates with more than few inputs, say 4, in the actual implementation of any Boolean function is not considered as a good design practice. This makes the complexity of calculating Boolean difference equations small. On the other hand for a fixed library of gates, all the Boolean difference equations can be calculated offline, so, there is no computational overhead due to calculating the Boolean difference equations in BDEC.

Table III shows the execution time and reliability calculation for some of synthesized tree-structured circuits with relatively larger number of inputs. Since the complexity of the PTM is really high for these circuits we only show the results for BDEC. Some of the circuits in Table III are the larger versions of the circuits reported in TABLE II. We have also included 16 and 32-bit ripple carry adder (RCA) circuits. Results for two benchmark circuits, I1 and C18, are also included in this table. TABLE IV compares the results for PTM, PGM [17], and BDEC for some more general circuits.

TABLE III. CIRCUIT RELIABILITY FOR TREE-STRUCTURED CIRCUITS HAVING RELATIVELY LARGE NUMBER OF PRIMARY INPUTS.

Circuit	# of Gates	Execution Time (ms)	Circuit Reliability
64-Input XOR (BT)	63	0.046	0.5007
64-Input XOR (Chain)	63	0.043	0.5007
64-Input AND (BT)	63	0.054	0.9475
64-Input AND (Chain)	63	0.051	0.9091
64-Input AOXAOXBT	63	0.054	0.6314
64-Input XAOXAOBT	63	0.053	0.9475
16-Bit RCA	80	0.115	0.0132
32-Bit RCA	160	0.216	0.0002
I1	46	0.054	0.3580
C18	6	0.013	0.8032

TABLE IV. CIRCUIT RELIABILITY AND EFFICIENCY OF BDEC COMPARED TO PGM AND PTM.

Circuit	Execution Time (ms)		Circuit Reliability (ε_g=0.05)			% Error compared to PTM	
	BDEC	PTM	BDEC	PGM	PTM	BDEC	PGM
2-4 Decoder	0.014	6.726	0.741	0.740	0.748	0.98	1.1
FA1	0.013	2.399	0.790	0.790	0.810	2.76	2.5
FA2	0.017	3.318	0.648	0.593	0.653	3.16	9.2
C17	0.012	2.304	0.766	0.762	0.784	2.62	2.8
COMP	0.014	0.937	0.751	0.730	0.826	9.24	11.8
Avg. Err.	-	-	-	-	-	3.75	5.5

Note that FA1 and FA2 are two different implementations of a full adder circuit. The former is a NAND only realization while the latter is an XOR/AND implementation. COMP is a two-bit comparator circuit. We report the results for our

implementation of PTM and BDEC; however, since we were not able to produce the results of PGM, we took the reported results in [17]. As seen from this table, BDEC achieves better accuracy compared to PGM.

TABLE V shows the results of running BDEC for somewhat larger benchmark circuits. In the last column, we report the results for some of the circuits that were analyzed in [3] to compare the run times of running PTM with that of BDEC. PTM results were reported for technology independent benchmarks where as BDEC results are for benchmark circuits mapped to a cell library in 65nm CMOS technology. PTM results were generated using a system with 3GHz Pentium 4 processor where as BDEC results are generated from a system with 2.4GHz dual core processor. One can see that BDEC (which has very low memory usage) is orders of magnitude faster than PTM. TABLE VI shows how BDEC execution times linearly scale with the gate counts.

TABLE V. RUNTIME COMPARISAON BDEC AND PTM FOR SOME LARGE BENCHMARK CIRCUITS .

Benchmark	# of Gates	PIs	POs	BDEC Exec Time (sec)	PTM Exec Times (sec)
C17	6	5	2	7.00E-06	0.313
pcle	71	19	9	2.40E-05	4.300
z4ml	74	7	4	2.20E-05	0.840
mux	106	21	1	2.80E-05	2.113
9symml	252	9	1	5.20E-05	696.211

TABLE VI. CIRCUIT RELIABILITY FOR LARGE BENCHMARK CIRCUITS .

Benchmark	# of Gates	PIs	POs	BDEC Exec Time (μ sec)	BDEC Reported Reliability
majority	22	5	1	9.0	0.6994
decod	66	5	16	18.0	0.2120
count	139	31	16	38.0	0.0707
frg1	143	28	3	48.0	0.6135
C880	442	60	26	96.0	0.0038
C3540	1549	50	22	358.0	0.0003
alu4	2492	12	8	577.0	0.0232
t481	4767	16	1	1710.0	0.8630

VI. CONCLUSION

As technology scales down circuit reliability is becoming one of the main concerns in VLSI design. In nano-scale CMOS regime circuit reliability have to be considered in the early design phases. This shows the need for fast reliability calculator tools that are accurate enough to estimate overall circuit reliability. The presented error/reliability calculator, BDEC, takes primary input signal and error probabilities and gate error probabilities and computes the reliability of the circuit. BDEC benefits from a linear-time complexity with number of the gates in the circuit. Compared to PTM which generates accurate reliability results, BDEC generates highly accurate results that are very close to PTM ones. We showed that the efficiency, execution time and memory usage, of BDEC is much better than those for PTM.

REFERENCES

[1] C. Hu, "Silicon nanoelectronics for the 21st century," *Nanotechnology*, vol. 10, no. 2, 1999, Page(s): 113-116.

[2] R.I. Bahar, C. Lau, D. Hammerstrom, D. Marculescu, J. Harlow, A. Orailoglu, W.H Joyner Jr., M. Pedram, "Architectures for silicon nanoelectronics and beyond," *Computer Magazine,* vol. 40, no. 1, Jan. 2007, Page(s): 25-33.

[3] S. Krishnaswamy, G.F. Viamontes, I.L. Markov, J.P. Hayes, "Accurate reliability evaluation and enhancement via probabilistic transfer matrices," *Proc. Design, Automation and Test in Europe,* 2005, Page(s): 282-287.

[4] A. Abdollahi, "Probabilistic Decision Diagrams for Exact Probabilistic Analysis," *Proc. Int'l Conference on Computer Aided Design,* 2007.

[5] H. Mehta, M. Borah, R.M. Owens, M.J. Irwin, "Accurate estimation of combinational circuit activity," *Proc. Design Automation Conference,* 1995, Page(s): 618-622.

[6] D. Bhaduri and S. Shukla, "NANOPRISM: A tool for evaluating granularity versus reliability trade-offs in nano architectures," in *Proc. 14th ACM Great Lakes Symp. VLSI*, 2004, Page(s): 109–112.

[7] G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla, "Evaluating the reliability of NAND multiplexing with PRISM," *IEEE Trans. on Computer Aided Design*, vol. 24, no. 10, Oct. 2005, Page(s): 1629–1637.

[8] D. Bhaduri, S.K Shukla, P.S. Graham, M.B. Gokhale, "Reliability Analysis of Large Circuits Using Scalable Techniques and Tools", *IEEE Trans. on Circuits and Systems*, vol. 54, no. 11, Nov. 2007 Page(s): 2447 – 2460.

[9] F. F. Sellers, M. Y. Hsiao, and L. W. Bearnson, "Analyzing Errors with the Boolean Difference," *IEEE Trans. on Computers,* vol. 17, no. 7, July 1968, Page(s): 676-683.

[10] S. B. Akers Jr, "On the theory of Boolean functions," *SIAM J. Appl. Math.*, vol. 7, Page(s):487-498, 1959.

[11] C-T. Ku and G.M. Masson, "The Boolean Difference and Multiple Fault Analysis," *IEEE Trans. on Computers*, vol. c-24, no. 1, Jan. 1975, Page(s): 62-71.

[12] S. R. Das, P. K. Srimani, and C. R. Dutta, "On Multiple Fault Analysis in Combinational Circuits by Means of Boolean Difference," *Proc. of the IEEE*, vol. 64, no. 9, Sept. 1976, pp. 1447-1449.

[13] T. Rejimon, S. Bhanja, "An accurate probabilistic model for error detection," *Proc. 18th International Conference on VLSI Design*, 2005, Page(s):717-722.

[14] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco, "Testability Measures in Pseudorandom Testing," *IEEE Trans. on Computer Aided Design*, vol. 11, no. 6, Jun. 1992, Page(s):794-800.

[15] R. Marculescu, D. Marculescu and M. Pedram, "Probabilistic modeling of dependencies during switching activity analysis," *IEEE Trans. on Computer Aided Design,* Vol. 17, no. 2, Feb. 1998, Page(s):73-83.

[16] K. P. Parker, E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks," *IEEE Trans. on Computers*, vol. 24, no. 6 (June 1975) Pages 668-670.

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

[18] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. SangiovanniVincentelli, "SIS: A system for sequential circuit synthesis," *U.C. Berkeley, Tech. Rep.,* May 1992.