Evaluating Circuit Reliability Under Probabilistic Gate-Level Fault Models

Ketan N. Patel, Igor L. Markov and John P. Hayes
University of Michigan, Ann Arbor 48109-2122
{knpatel,imarkov,jhayes}@eecs.umich.edu

Abstract

Circuit reliability is an increasingly important design consideration for modern logic circuits. To this end, our work focuses on the evaluation of circuit reliability under probabilistic gate-level fault models that can capture both soft errors, e.g., radiation-related, and spatially-uniform manufacturing defects. This basic task can, in principle, be used (i) by synthesis procedures to select more reliable circuits, and (ii) to estimate yield for electronic nanotechnologies where high defect density is expected.

We propose a matrix-based formalism to compute the error probability of the whole circuit based on probabilities of specific gate errors. This formalism is surprisingly related to that of quantum circuits, but also exhibits several new features. The numerical computation of error probabilities in large circuits runs into the same scalability problems as the simulation of quantum circuits. Therefore, we hope to adapt recent advances in quantum circuit simulation to the context of this work.

1 Introduction

Device geometries continue to shrink in deep sub-micron VLSI circuits as clock speeds increase and per-chip costs drop. Finer device features make circuits more susceptible to manufacturing defects, noise-related transient faults and interference from radiation. In particular, we distinguish (i) *soft errors* that occur non-deterministically and may be related to alpha-particles or gamma-radiation [4], and (ii) *manufacturing defects* that permanently damage circuit components. Both types manifest in the form of logic errors and may be difficult to prevent in the near-future technologies. Therefore, fault-tolerance and error-correction techniques appear increasingly important.

Previously, fault-tolerant circuits and architectures have been proposed to absorb soft errors without jeopardizing the function of computing systems. Examples include totally self-checking circuits [5] and dynamic verification of microprocessor operations [1]. In the latter approach, when a fault is detected, the operation is redone, with a serious runtime penalty. However, the checker circuit is small and does not slow down the processor until a fault is discovered. With dynamic verification, the main processor does not have to be fully verified because rare faults, not found during simulation, can be tolerated. However, dynamic verification assumes very low incidence of logic errors, which can only be guaranteed by circuit-level techniques.

The existing literature on circuit test and fault-tolerant circuits typically uses interconnect-based models, e.g., *stuck-at* and *bridging* faults. While these models enable fast and practically useful algorithms, they cannot adequately model more subtle behavior that faulty gates may exhibit. On the other hand, some interconnect faults can be captured by gate-level fault models with the help of fake buffers inserted at the right places.

The anticipation of high defect densities in circuit components is supported by recent work on chemically-assembled electronic nanotechnology [2]. The authors expect defect rates of 1-10% in first technology generations. A proposed solution is a programmable circuit fabric with test structures that would allow to discover all actual defects in each circuit manufactured. A logic circuit can then be mapped onto fully-functional gates and wires.

With less drastic means of tolerating errors in mind, our work focuses on circuits with faulty gates. In particular, we point out that technology-specific faults in logic gates can be modeled probabilistically: for a given input, every output can be observed with a certain probability. Such models capture not only soft errors, but also manufacturing defects that are equally likely to occur in any gate of a given type. Our work shows that the probability of erroneous output depends not only on the faultiness of gates, but also on circuit structure. Related logic optimization would require a reliable way to evaluate the error probability of a circuit. The main contribution of this work is an analytical technique for this task. This technique fundamentally differs from existing architectural [1, 4] and empirical approaches. We also point out a surprising connection between the evaluation of circuit error probability and numerical simulation of quantum circuits.

The remaining part of this manuscript is organized as follows. We introduce probabilistic gate-level fault models in Section 2 where we also compute error-properties of logic circuits with one-bit output. In Section 3 we show that multi-output circuits can achieve better fault-tolerance. Similarities of the error-probability computation to the simulation of quantum circuits are covered in Section 4. Conclusions and on-going work are discussed in Section 5.

2 Basic Methodology

Below we first introduce gate-level fault models and then explain how error properties of a logic circuit can be modeled in terms of those models. An example is given.

2.1 Probabilistic Gate-Level Fault Models

In error-free operation the function of a combinational logic circuit or gate can be represented by a *truth table*, which is a deterministic mapping of input values to output values. For example, the truth table for an AND gate maps the input value 01 to the output value 0. However, in the presence of soft errors or manufacturing defects this input may occasionally lead to a 1 at the output of the gate. If we know how often this is likely happen, we can model this behavior using a *probabilistic transfer matrix*. In this matrix, column indices represent input values, and row indices represent output values, while matrix elements capture pair-wise transition probabilities.

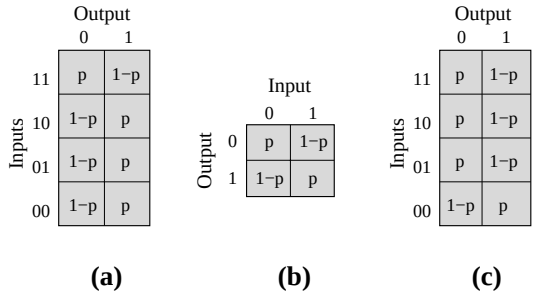

Figure 1: Probabilistic transfer matrices used for noisy logic gates: (a) AND, (b) NOT, and (c) OR. The variable p denotes the probability the gate will give an incorrect output. Note matrices a and c are rotated.

For example, consider the possible probabilistic transfer matrices shown in Figure 1 for the the standard AND, NOT and OR gates. In these examples the gates give the incorrect output value with probability p. In general, we could use any fixed probability distribution for the columns of the matrices. For example, if we knew that the AND gate was twice as likely to give the incorrect output for inputs 01 and 10 versus for inputs 00 and 11, we could reflect this is the matrix:

$$\begin{bmatrix} 1-p & 1-2p & 1-2p & p \\ p & 2p & 2p & 1-p \end{bmatrix}.$$

The only requirement is that each column sums to 1, since we assume that some value is always seen at the output. The number of rows and columns are exponential in the number of gate inputs and outputs respectively.

2.2 Derivation of Circuit Properties

The concept of the probabilistic transfer matrix is not restricted to gates. It can also represent the function of a noisy circuit. The element in the i-th row and j-th column represents the probability that the i-th output value will be observed when the j-th input value is given to the circuit. Now for convenience we also define the *ideal transfer matrix*. This is the probabilistic transfer matrix when there are no errors. Each column of this matrix contains

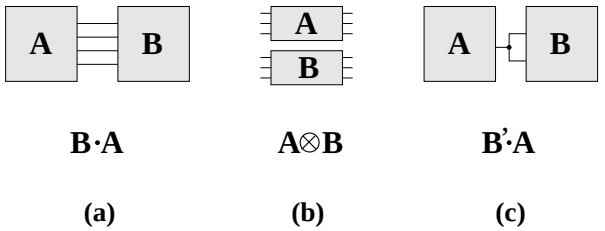

Figure 2: Basic ways to interconnect circuit components and the corresponding operation on the probabilistic transfer matrices. (a) serial (b) parallel (c) fanout

exactly one 1 with the remaining elements being 0. Now if we know the probability distribution of input values that will be given to the circuit, that is, if we know that the i-th input value will be given to the circuit with probability p_i, then we can calculate the error probability of the circuit from the ideal and probabilistic transfer matrices C and P:

$$\sum_{i=0}^{2^n-1} p_i \sum_{j:C(j,i)=0} P_{[j,i]},$$

where $P_{[i,j]}$ denotes the element in the i-th row and j-th column of matrix P. For the noisy AND gate in Figure 1a this calculation gives a probability p that the gate will give an incorrect output.

Therefore, if we can construct the probabilistic transfer matrix for a circuit, then we can calculate its error probability. Now we show that if we assume that gate errors occur independently and we know the probabilistic transfer matrices of the gates in the circuit, then by performing operations on these we can obtain the probabilistic transfer matrix of the circuit. The basic method is as follows: we construct the circuit by connecting together gates into sub-circuits then connecting together these sub-circuits; each time we connect two components, we derive the probabilistic transfer matrix of the combined circuit from those of the components.

There are three basic ways in which components in a well-formed circuit are combined: serially, in parallel, and through fanout. These are illustrated in Figure 2. We now show how each of these operations affects the probabilistic transfer matrices.

Serial Composition is the most straightforward. This occurs when a component acts on the outputs of another component. The probabilistic transfer matrix of the result is given by the product of those of the components.

For example, consider the serially connected components shown in Figure 2a. Suppose A and B denote the probabilistic transfer matrices of the two components. Let component A have n inputs and m outputs, and component B have m inputs and l outputs. Let us calculate the probability that when an input i is given to the first component, we observe an output j. This is by definition the element in the j-th row and i-th column of the overall error transfer matrix. If an input i is given to the first component, the probability distribution of the outputs of the first component are given by the i-th column of A. The vector with the probabilities that each of the possible inputs to the second component give an output j is given by the j row of B. Therefore, the probability that the output of the second component will be j, given that i is input to the first component, is equal to the inner product of the i-th column of A and the j-th row of B. Thus, the overall probabilistic transfer matrix is given by $B \cdot A$.

Parallel Composition involves two or more circuit components acting on disjoint sets of bits. The corresponding matrix operation is the tensor product of the probabilistic transfer matrices of the components.

Consider the example shown in Figure 2b. Suppose A and B denote the probabilistic transfer matrices of the two components. Let component A have n inputs and m outputs and let component B have l inputs and k outputs. Consider an input i to the overall circuit, which can be partitioned into inputs i_A and i_B to the respective components A and B. Since the two components are independent, the probability distribution of the outputs for the first component are given by the i_A-th column of A and those of the second component are given by the i_B-th column of B. So the probability that an output $j = (j_A\ j_B)$ is observed is given by the product

$$A_{[j_A,i_A]} \cdot B_{[j_B,i_B]},$$

which is the element in the i-th row and j-th column of the combined probabilistic transfer matrix. Writing out the entire matrix we have

$$\begin{bmatrix} A_{[0,0]} \cdot B & A_{[0,1]} \cdot B & \cdots & A_{[0,n]} \cdot B \\ A_{[1,0]} \cdot B & A_{[1,1]} \cdot B & \cdots & A_{[1,n]} \cdot B \\ \vdots & & & \vdots \\ A_{[m,0]} \cdot B & A_{[m,1]} \cdot B & \cdots & A_{[m,n]} \cdot B \end{bmatrix},$$

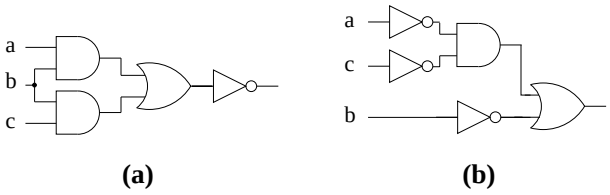

Figure 3: Two circuits that compute the function $f(a,b,c) = \overline{b} + \overline{a} \cdot \overline{c}$. The circuit to the right is more robust to transient gate faults than the one to the left, assuming a uniform gate failure probability.

which is equal to the tensor (Kronecker) product $A \otimes B$.

Fanout When components are connected with fanout an output of one component is connected to multiple inputs of the following component. In order to represent this type of connection, we simply eliminate those columns in the probabilistic transfer matrix of the second component that correspond to different values on the copied wire.

For example, consider the circuit in Figure 2c. If component B has two inputs both from the same output of component A, we would eliminate columns $1 = \{01\}$ and $2 = \{10\}$ from component B's probabilistic transfer matrix.

2.3 Circuit Example

Using the three basic operations above, we can find the overall probabilistic transfer matrix for any well-formed combinational circuit. Now we illustrate this method for two different circuits shown in Figure 3, both computing the function $f(a,b,c) = \overline{b} + \overline{a} \cdot \overline{c}$. For convenience, we denote the probabilistic transfer matrices of the AND, NOT, and OR gates by the names of the gates and to simplify the algebra we define $q = 1 - p$. First we take the circuit given in Figure 3a. At depth 1 we take the tensor of the two AND gates:

$$T_1 = \text{AND} \otimes \text{AND} = \begin{bmatrix} q & q & q & p \\ p & p & p & q \end{bmatrix} \otimes \begin{bmatrix} q & q & q & p \\ p & p & p & q \end{bmatrix}.$$

This yields a 4×16 matrix with column labels *abbc*. In order to reduce out the duplicate copy of b we eliminate the columns where the two copies are not the same, i.e., columns with labels of the form $X01X$ and $X10X$. This

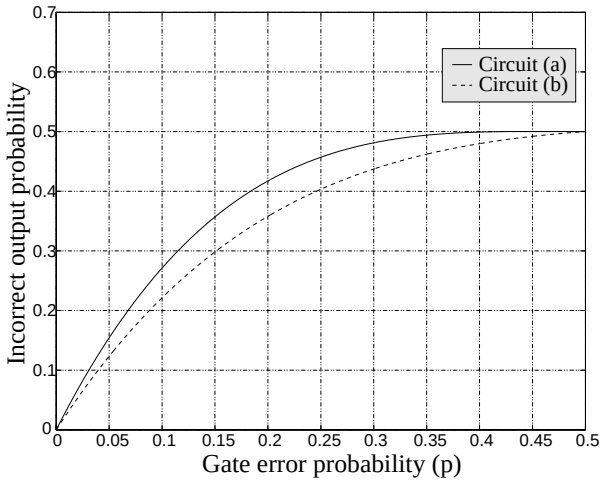

Figure 4: Probability of incorrect output as a function of gate error probability for the two circuits in Figure 3.

leaves the following transfer matrix:

$$T_1' = \begin{bmatrix} q^2 & q^2 & q^2 & qp & q^2 & q^2 & qp & p^2 \\ qp & qp & qp & q^2 & qp & qp & p^2 & qp \\ qp & qp & qp & p^2 & qp & qp & q^2 & qp \\ p^2 & p^2 & p^2 & qp & p^2 & p^2 & qp & q^2 \end{bmatrix}.$$

Finally we apply the OR and NOT gates:

$$T_3 = \text{NOT} \cdot T_2 = \text{NOT} \cdot \text{OR} \cdot T_1'$$

For the circuit in Figure 3b, we have

$$T_3 = \text{OR} \cdot (\text{AND} \otimes \text{NOT}) \cdot (\text{NOT} \otimes \text{NOT} \otimes I_2),$$

where I_2 is the probabilistic transfer matrix for an ideal wire, i.e., the 2×2 identity matrix.

We can now calculate the error probability for each circuit given the input value distribution. Assuming that all input values to the circuits are equiprobable, Figure 4 shows the probability of an incorrect output for both circuits with respect to the gate error probability p. The second circuit has a lower probability of giving an incorrect output over the entire range of gate error probabilities. The basic intuition here is that long narrow circuits are

generally more error-prone than short wide ones. In this example, all paths to the output in the first circuit must go through the last two gates. Therefore, faults in either of these gates affect all possible inputs. By contrast, in the second circuit, only the last gate is used by all paths from an input to the output.

This example should only be taken as an illustration of our method, not as a realistic evaluation of the relative robustness of the circuits in Figure 3. In a real application the probability that a gate fails would probably be correlated with both the gate type and the gate inputs. Both of these could be incorporated into our method by using more accurate probabilistic transfer matrices for the gates in the circuits.

3 Better Fault-Tolerance using Multiple Outputs

Thus far our primary goal has been to evaluate error properties of different circuit structures that compute the same function. However the achievable error properties for the circuits we have considered have been limited by the error-resistance of the gate driving the output: if this gate gives a faulty output so does the circuit. This is a general limitation of single-output circuits.

One way to decrease the probability of errors is to encode input and output using redundant wires. In this case, a logical bit (or set of bits) is represented by a larger number of physical bits. For example, a simple output encoding could map the logical value 0 to three physical bits all with value 0, similarly with the logical 1 value. A circuit with a single logical output bit would actually have three physical bits as outputs. If one of the output gates fails, the correct logical output is still retrievable from the remaining two correct output bits. Of course, decoding circuitry is necessary at some point, only delaying the inevitable single gate failure limit, however this may be postponed to the end of the computation where this relatively small decoding logic can be made physically robust. This is in contrast to making each of the circuit's gates physically more robust.

The use of encoded inputs and output fits relatively seamlessly into our methodology. For representing the input encoding, we have two choices: we can either choose the input value distribution to reflect the encoding, i.e., have probability 0 for non-code values; or we can actually eliminate the columns of the probabilistic transfer matrices representing non-code values. The latter is very similar to our method for representing fanout in Section 2.2. In general the first method is conceptually simpler, but may require extra unnecessary computations. At the output we need to represent the mapping of physical output bits to logical outputs; in our three bit example above, we would map 000, 100, 010, and 001 all to the logical output value 0. This is easily done by modifying the ideal transfer matrix: rather than a single 1 in each column of this matrix representing the correct output value, we have multiple 1's representing each of the possible correct output values.

Another way to increase the error-resistance of a single output circuit is to increase the reliability of the limiting gate, namely the gate driving the output. This may be possible in some implementation and for some noise environments; for example, two basic gate sizes may be available in some implementations with the larger being less susceptible to radiation effects. Carrying this argument further, we could add the flexibility to choose gate sizes for any of the gates in the circuit, given perhaps some limit on the area overhead. However, in general it may not be obvious which gates sizes should be increased to maximize error-resistance. Our method for evaluating the circuit's error-resistance could be very useful for this.

4 Quantum Circuits and Scalability

Quantum computations are represented by complex unitary matrices [6] (multiplying a unitary matrix by its conjugate-transpose gives the identity matrix). The square of the amplitude of each element gives the probability of a transition from one basis state to another. Such matrices capture the functions of individual quantum gates and also whole quantum circuits (note that quantum gates have as many inputs as outputs, ditto for quantum circuits). Quantum states, including circuit input and output, are complex vectors. Given a unitary matrix of a quantum gate, the output is produced by input using matrix-vector multiplication. Because of this, sequential composition is represented by the regular matrix-matrix multiplication and parallel composition by the tensor (Kronecker) product.

This is similar to how we compute the probabilistic transfer matrix of a circuit from gate matrices.

There are a few important differences between our problem and the quantum circuit problem.

1. Quantum circuits do not have fanouts. Thus, one of our operations does not have a quantum analog.
2. All quantum gates have the same number of outputs as inputs, and therefore all of the matrices are square. This is usually not the case for our problem.
3. The matrix elements in our case are real, while they may be complex in the quantum circuit case. On the other hand, our matrices do not have to be real-unitary (a.k.a. orthogonal) – the only requirement is that all elements in every column add up to one.
4. Our computation differs greatly from the operation of quantum circuits in the last step. Quantum measurement is applied after a quantum circuit, but we use an unrelated operation to calculate the error probability from the probabilistic transfer matrix.

The matrices in both our problem and the quantum circuit problem are double-exponential in size with respect to the number of inputs and outputs. This leads to obvious scalability concerns for numerical methods. However, these concerns are already being addressed in the field of simulation of quantum circuit with BDD-based techniques [3, 7] which we hope to adapt to our domain. To simulate a quantum circuit, one starts with an input vector and computes an output vector. Quantum measurement — the last step, irrelevant to our work — is simulated by computing the probabilities of all possible outcomes. The simulation of quantum circuits [3, 7] is important, because, among other things, it provides a method to evaluate designs prior to implementation. A recent proposal to deal with the scalability problem detects and eliminates common arithmetic sub-expressions, drastically simplifying the computation [7]. Computationally, this approach is based on Binary Decision Diagrams, particularly the QuIDD datastructure adapted from Algebraic Decision Diagrams (ADDs). All matrices and vectors are represented by decision diagrams and relevant operations are performed using DD traversals. This approach appears applicable to our problem as well, with the caveat that we must develop a new BDD-based operation to model fanout. This is one of directions of on-going work.

5 Conclusions and On-going Work

This work is preliminary, and we are investigating several further directions. These include more detailed analyses of the analogy with quantum circuit simulation.

Through the adaptation of the QuIDD Pro quantum circuit simulator [7] we hope to significantly improve the performance of our basic method. We also exploring ways to simplify the calculations by using approximations at critical stages, while still preserving the integrity of the result.

We also plan incorporate our circuit reliability evaluation method into synthesis algorithms, particularly fault-tolerant synthesis methods acting on encoded inputs and outputs. This has the potential of significantly improving circuit reliability.

A better understanding of fault-tolerant logic may be possible by analytical means. For example, one somewhat obvious property we've observed is that long narrow circuits tend to be more susceptible to gate faults than short wide ones. Further studies of error properties of logic circuits, e.g., trees, may lead to new circuit optimization strategies and heuristics usable during synthesis.

References

[1] T. M. Austin. "DIVA: A Dynamic Approach to Microprocessor Verification," *Journal of Instruction Level Parallelism*, May 2000.

[2] S. C. Goldstein and M. Budiu. "NanoFabrics: Sparial Computing Using Molecular Electronics," *Intl. Symp. on Comp. Arch.*, June 2001.

[3] D. Greve. "QDD: a quantum computer emulation library," http://home.plutonium.net/~dagreve/qdd.html

[4] S. Kim and A. K. Somani. "Soft error sensitivity characterization for microprocessor dependability enhancement strategy," *Proc. Intl. Conf. on Dependable Systems and Networks*, 2002.

[5] P. K. Lala. *Self-Checking and Fault-Tolerant Digital Design*. Academic Press, Inc., 2001.

[6] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[7] G. F. Viamontes, M. Rajagopalan, I. L. Markov, and J. P. Hayes. "Gate-level Simulation of Quantum Circuits," *Proc. ASPDAC 2003*, pp. 295-301.