

Chapter 2

Probabilistic Transfer Matrices

In this chapter, we present a general framework for reliability analysis that treats circuits entirely probabilistically. While this is useful for analyzing soft errors, it is also useful for analyzing devices that periodically fail or behave probabilistically during regular operation. Quantum dot cellular automata (QCA), where gates and wires are made from “quantum dots”, are examples of such devices; see Fig. 1.5. Each dot consists of a pair of electrons that can be configured in two different ways to represent a single bit of information. In QCA, both gates and wires are created from planar arrangements of dots. QCA have an inherent propensity for faults because the electrons can easily be absorbed into the atmosphere or arrange themselves in an ambiguous configuration [1, 2]. Other examples of inherently probabilistic devices include probabilistic CMOS, molecular logic circuits, and quantum computers.

Historically, the probabilistic analysis of circuits has centered around signal-probability estimation, which was motivated by random-pattern testability concerns [3–5]. In short, the probability of a signal being a 0 or 1 gives some indication of the difficulty in controlling (and therefore testing) the signal. In this chapter, we treat circuits probabilistically to analyze circuit reliability. As opposed to signal probability estimation, reliability analysis deals with complex probabilistic failure modes and error propagation conditions.

In general, accurate reliability analysis involves computing not just a single output distribution but, rather, the output error probability for each input pattern. In cases where each gate experiences input-pattern dependent errors—even if the input distribution is fixed—simply computing the output distribution does not give the overall circuit error probability. For instance, if an XOR gate experiences an output bit-flip error, then the output distribution is unaffected, but the wrong output is paired with each input. Therefore, we need to separately compute the error associated with each input vector.

Consider the circuit in Fig. 2.1. Given that each gate experiences an error with probability $p = 0.1$, the circuit’s output error probability for the input combination 000 is 0.244. The input combination 111 leads to an output error probability of 0.205. The overall error rate of the circuit is the sum of the error probabilities, weighted by

the input combination probabilities. The probability of error for the circuit in Fig. 2.1, given the uniform input distribution, is therefore 0.225. Note that joint probabilities of input combinations, rather than individual input probabilities, are necessary to capture correlations among inputs.

We analyze circuit reliability and other aspects of non-deterministic behavior, using a representation called a probabilistic transfer matrix (PTM). A PTM for a gate (or a circuit) gives the probability of each output combination, conditioned upon the input combinations. PTMs can model gates exhibiting varying input-dependent error probabilities. PTMs form an algebra—a set closed under specific operations—where the operations in question are matrix multiplication and tensor products. These operations may be used to compute overall circuit behavior by combining gate PTMs to form circuit PTMs. Matrix products capture serial connections, and tensor products capture parallel connections.

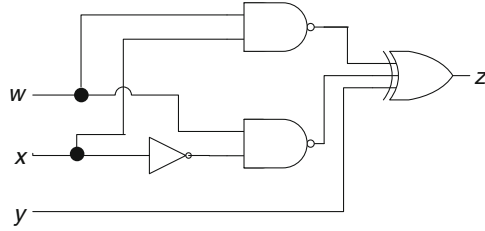
For those familiar with Bayesian inference, a PTM for a gate is essentially a conditional probability table and reliability analysis is a specific, although, complex form of Bayesian inference. Our aim is to compute the joint probability of the primary outputs given joint probabilities of primary inputs. Circuits offer a very natural way of decomposing the joint probability distribution because essentially, the output probability of a gate only depends on its immediate input probability. Therefore, each gate can be represented by a conditional probability table. However, unlike traditional Bayesian analysis, in this chapter we discuss operations necessary to combine these probability distributions to form a joint probability distribution utilizing algebraic analysis. In the next chapter, we show how to scale this computation to simple circuits using decision-diagram-based compression. Most of the concepts and results described in this chapter also appear in [6, 7].

2.1 PTM Algebra

This section describes the PTM algebra and some key operations for manipulating PTMs. First, we discuss the basic operations needed to represent circuits and to compute circuit PTMs from gate PTMs. Next, we define additional operations to extract reliability information, eliminate variables, and handle fan-out efficiently.

Consider a circuit C with n inputs and m outputs. We order the inputs for the purposes of PTM representation and label them in_0, \dots, in_{n-1} ; similarly, the m outputs are labeled out_0, \dots, out_{m-1} . The circuit C can be represented by a $2^n \times 2^m$ PTM M . The rows of M are indexed by an n -bit vector whose values range from $\underbrace{000 \dots 0}_n$ to $\underbrace{111 \dots 1}_n$. The row indices correspond to input vectors, i.e.

0/1 truth assignments of the circuit's input signals. Therefore, if $\mathbf{i} = i_0 i_1 \dots i_{n-1}$ is an n -bit input vector, then row $M(\mathbf{i})$ gives the output probability distribution for n input values $in_0 = i_0, in_1 = i_1 \dots in_{n-1} = i_{n-1}$. Similarly, column indices correspond to truth assignments of the circuit's m output signals. If \mathbf{j} is an m -bit vector, then entry $M(\mathbf{i}, \mathbf{j})$ is the conditional probability that the out-



$$(F_2 \otimes F_2 \otimes I)(I \otimes \text{swap} \otimes \text{NOT}_p \otimes I)(\text{NAND}_{2p} \otimes \text{NAND}_{2p} \otimes I)(\text{XOR}_{3p})$$

Fig. 2.1 Sample logic circuit and its symbolic PTM formula

Fig. 2.2 a ITM for the circuit in Fig. 2.1; **b** circuit PTM where each gate experiences error with probability $p = 0.1$

	0	1		0	1
000	1	0	000	0.756	0.244
001	0	1	001	0.244	0.756
010	1	0	010	0.756	0.244
011	0	1	011	0.244	0.756
100	0	1	100	0.295	0.705
101	1	0	101	0.705	0.295
110	0	1	110	0.295	0.705
111	1	0	111	0.705	0.295
(a)			(b)		

puts have values $\text{out}_0 = j_0, \text{out}_1 = j_1 \dots \text{out}_{m-1} = j_{m-1}$ given input values $\text{in}_0 = i_0, \text{in}_1 = i_1 \dots \text{in}_{n-1} = i_{n-1}$, i.e, $P[\text{outputs} = \mathbf{j} | \text{inputs} = \mathbf{i}]$. Therefore, each entry in M gives the conditional probability that a certain output combination occurs given a certain input combination.

Definition 2.1 Given a circuit C with n inputs and m outputs, the *PTM* for C is a $2^n \times 2^m$ matrix M whose entries are conditional probabilities of the form shown here: $M(\mathbf{i}, \mathbf{j}) = P[\text{outputs} = \mathbf{j} | \text{inputs} = \mathbf{i}]$.

Definition 2.2 A fault-free circuit has a PTM called an *ideal transfer matrix (ITM)* in which the correct logic value of each output occurs with probability 1.

The PTM for a circuit represents its functional behavior for all input and output combinations. An input vector for an n -input circuit is a row vector with dimensions $2^n \times 1$. Entry $v(\mathbf{i})$ of an input vector v represents the probability that the input values $\text{in}_0 = i_0, \text{in}_1 = i_1 \dots \text{in}_{n-1} = i_{n-1}$ occur. When an input vector is right-multiplied by the PTM, the result is an output vector of size 1×2^m . The output vector gives the resulting output distribution. Examples of an ITM and PTM are shown in Fig. 2.2.

2.1.1 Basic Operations

PTMs can be defined for all the gates of a logic circuit by taking into account errors affecting the gates. A PTM for the entire circuit can then be derived from the PTMs

of the gates and their interconnections. The basic operations needed to compute the circuit PTM from component PTMs are the matrix and tensor products.

Consider the circuit C formed by connecting two gates g_1 and g_2 in series, i.e., the outputs of g_1 are connected to the inputs of g_2 . Suppose these gates have PTMs M_1 and M_2 ; then the entry $M(\mathbf{i}, \mathbf{j})$ of the resulting PTM M for C represents the probability that g_2 produces output j , given g_1 has input i . This probability is computed by summing over all values of intermediate signals (outputs of g_1 which are also inputs of g_2) for input \mathbf{i} of g_1 and output \mathbf{j} of g_2 . Therefore, each entry $M(\mathbf{i}, \mathbf{j}) = \sum_{\text{all } \mathbf{h}} M_1(\mathbf{i}, \mathbf{h}) M_2(\mathbf{h}, \mathbf{j})$. This operation corresponds to the ordinary matrix product $M_1 M_2$ of the two component PTMs.

Now suppose that circuit C is formed by two parallel gates g_1 and g_2 with PTMs M_1 and M_2 . An entry in the resulting matrix M should represent the joint conditional probability of a pair of input–output values from g_1 and a pair of input–output values from g_2 . Each such entry is therefore a product of independent conditional probabilities from M_1 and M_2 , respectively. These joint probabilities are given by the tensor product operation.

Definition 2.3 Given two matrices M_1 and M_2 , with dimensions $2^k \times 2^l$ and $2^m \times 2^n$, respectively, the *tensor product* $M = M_1 \otimes M_2$ of M_1 and M_2 is a $2^{km} \times 2^{ln}$ matrix whose entries are:

$$M(i_0 \dots i_{k+m-1}, j_0 \dots j_{l+n-1}) = M_1(i_0 \dots i_{k-1}, i_0 \dots j_{l-1}) \\ \times M_2(i_k \dots i_{k+m-1}, j_l \dots j_{l+n-1})$$

Figure 2.3 shows the tensor product of an *AND* ITM with an *OR* ITM. Note that the *OR* ITM appears once for each occurrence of a 1 in the *AND* ITM; this is a basic feature of the tensor product.

Besides the usual logic gates (AND, OR, NOT, etc.), it is useful to define three special gates for circuit PTM computation. These are (i) the n -input identity gate with ITM denoted I_n ; (ii) the n -output fan-out gate F_n ; and (iii) the swap gate *swap*. These wiring PTMs are shown in Fig. 2.4.

An n -input *identity gate* simply outputs its input values with probability 1. It corresponds to a set of independent wires or buffers and has the 2×2 identity matrix as its ITM. Larger identity ITMs can be formed by the tensor product of smaller identity ITMs. For instance, the ITM for a 2-input, 2-output identity gate is $I_2 = I \otimes I$. More generally, $I_{m+n} = I_m \otimes I_n$. An n -output *fan-out gate*, F_n , copies an input signal to its n outputs. The ITM of a 2-output fan-out gate, shown in Fig. 2.4b, has entries of the form $F_2(i_0, j_0 j_1) = 1$, where $i_0 = j_0 = j_1$ and all other entries are 0. Therefore, the 5-output fan-out ITM, F_5 , has entries $F_5(0, 00000) = F_5(1, 11111) = 1$, with all other entries 0. Wire permutations such as crossing wires are represented by *swap gates*. The ITM for an adjacent-wire swap (a simple two-wire crossover) is shown in Fig. 2.4c. Any permutation of wires can be modeled by a network of swap gates.

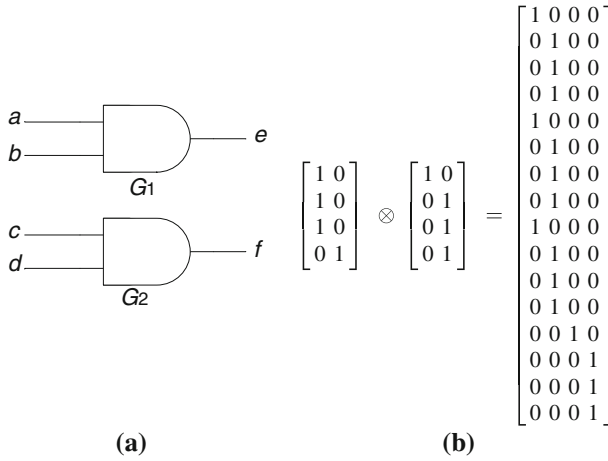


Fig. 2.3 Illustration of the tensor product operation: **a** circuit with parallel AND and OR gates; **b** circuit ITM formed by the tensor product of the AND and OR ITMs

Fig. 2.4 Wiring PTMs: **a** identity gate I ; **b** 2-output fan-out gate F_2 ; **c** wire-swap gate denoted $swap$

$$\begin{array}{ccc}
 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 \text{(a)} & \text{(b)} & \text{(c)}
 \end{array}$$

Example 2.1 Consider the circuit in Fig. 2.5—this is the circuit of Fig. 2.1 with its wiring gates made explicit. The PTMs for the gates with error probability p are as follows:

$$\begin{array}{ccc}
 \begin{bmatrix} p & 1-p \\ p & 1-p \\ p & 1-p \\ 1-p & p \end{bmatrix} & \begin{bmatrix} 1-p & p \\ p & 1-p \\ p & 1-p \\ 1-p & p \\ p & 1-p \\ 1-p & p \\ 1-p & p \\ p & 1-p \end{bmatrix} & \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix} \\
 NAND_{2p} & XOR_{3p} & NOT_p
 \end{array}$$

The corresponding circuit PTM is expressed symbolically by the formula in Fig. 2.5. Each parenthesized term in this formula corresponds to a level in the circuit. The advantage of evaluating the circuit PTM using such an expression is that the error probabilities for the entire circuit can be extracted from it.

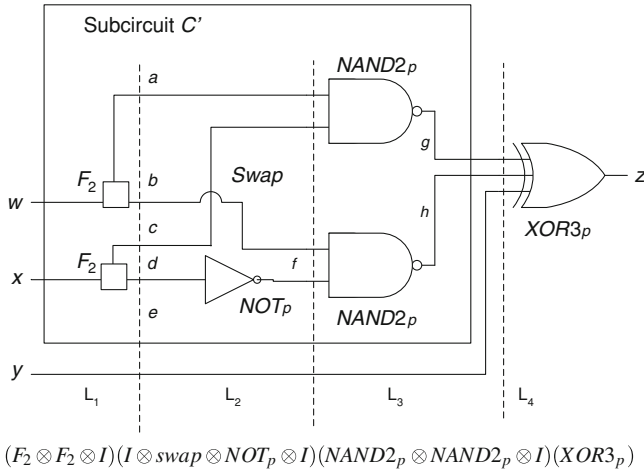


Fig. 2.5 Circuit to illustrate PTM calculation; vertical lines separate levels of the circuit; the parenthetical subexpressions correspond to logic levels

2.1.2 Additional Operations

In addition to the basic operations of matrix multiplication and tensor product, we introduce the following three operations to increase the scope and efficiency of PTM-based computation:

- *fidelity*. This operation measures the similarity between an ITM and a corresponding PTM. It is used to evaluate the reliability of a circuit.
- *eliminate_variables*. This operation computes the PTM of a subset of inputs or outputs, starting from a given PTM. It can also be used to compute the probability of error of individual outputs.
- *eliminate_redundant_variables*. This operation eliminates redundant input variables that result from tensoring matrices of gates that are in different fan-out branches of the same signal

We now formally define and describe these operations in more detail. First, we define the *element-wise* product used in computing *fidelity*.

Definition 2.4 The element-wise product of two matrices A and B , both of dimension $n \times m$, is denoted $A * B = M$ and defined by $M(i, j) = A(i, j) \times B(i, j)$.

To obtain the *fidelity*, the element-wise product of the ITM and the PTM is multiplied on the left by the input vector, and the norm of the resulting matrix is computed. In the definition below, $\|\mathbf{v}\| = \sum_i \|v_i\|$ denotes the l_1 norm of vector \mathbf{v} .

Definition 2.5 Given a circuit C with PTM M , ITM J , and input vector \mathbf{v} , the fidelity of M is defined as

$$fidelity(v, M, J) = ||v(M * J)||$$

The fidelity of a circuit is a measure of its *reliability* against transient errors. Fig. 2.6 illustrates the *fidelity* computation on the circuit of Fig. 2.1.

Example 2.2 Consider the circuit C from Fig. 2.1, with inputs $\{w, x, y\}$ and output $\{z\}$. The ITM is denoted J , and the PTM, shown in Fig. 2.2b, is denoted M . The circuit PTM is calculated using the PTMs from Example 2.1, with probability of error $p = 0.1$ at each gate, on all inputs. Fig. 2.6 shows intermediate matrices needed for this computation. The quantity $fidelity(v, M, J)$ is found by first element-wise multiplying J and M , then left-multiplying by an input vector v . The l_1 norm of the resulting matrix is $fidelity(v, M, J) = (0.3716 + 0.3716) = 0.7432$. The probability of error is $1 - 0.7432 = 0.2560$.

The *eliminate_variables* operation is used to compute the “sub-PTM” of a smaller set of input and output variables. We formally define it for 1-variable elimination as follows.

Definition 2.6 Given a PTM matrix M that represents a circuit C with inputs in_0, \dots, in_{n-1} , $eliminate_variables(M, in_k)$ is the matrix M' with $n - 1$ input variables $in_0, \dots, in_{k-1}, in_{k+1}, \dots, in_{n-1}$ whose rows are

$$\begin{aligned} M'(i_0 \dots i_{k-1} i_{k+1} \dots i_{n-2}, \mathbf{j}) &= M(i_0 \dots i_{k-1} 0 i_{k+1} \dots i_{n-2}, \mathbf{j}) \\ &\quad + M(i_0 \dots i_{k-1} 1 i_{k+1} \dots i_{n-2}, \mathbf{j}) \end{aligned}$$

The *eliminate_variables* operation is similarly defined for output variables.¹ The elimination of two variables can be achieved by eliminating each of the variables individually in arbitrary order. Fig. 2.7 demonstrates the elimination of column variables from a subcircuit C' of the circuit in Fig. 2.5, formed by the logic between inputs w, x and outputs g, h . The PTM for C' with probability of error $p = 0.1$ on all its gates is given by:

$$(F_2 \otimes F_2)(swap \otimes NOT_p)(NAND2_p \otimes NAND2_p)$$

If we eliminate output h , then we can isolate the conditional probability distribution of output g , and vice versa. Output h corresponds to the second column variable of the PTM in Fig. 2.7b. To eliminate this variable, columns with indices 00 and 01 of Fig. 2.7b are added, and the result is stored in the column 0 of the resultant matrix (Fig. 2.7c). Columns 10 and 11 of M are also added, and the result is stored in column 1 of the resultant matrix. The final PTM gives the probability distribution of output variable g in terms of the inputs w and x . A similar process is undertaken for

¹ The *eliminate_variables* operation is analogous to the existential abstraction of a set of variables x in a Boolean function f [8], given by the sum of the positive and negative cofactors of f , with respect to x : $\exists x f = f_x + f_{\bar{x}}$. The *eliminate_variables* operation for PTMs relies on arithmetic addition of matrix entries instead of the Boolean disjunction of cofactors.

$$\begin{array}{ccc}
 v^T = \begin{bmatrix} 0.125 \\ 0.125 \\ 0.25 \\ 0.25 \\ 0 \\ 0 \\ 0.125 \\ 0.125 \end{bmatrix} & J.*M = \begin{bmatrix} 0.756 & 0 \\ 0 & 0.756 \\ 0.756 & 0 \\ 0 & .756 \\ 0 & 0.705 \\ 0.705 & 0 \\ 0 & 0.705 \\ 0.705 & 0 \end{bmatrix} & v(J.*M) = [0.3716 \ 0.3716] \\
 \text{(a)} & \text{(b)} & \text{(c)}
 \end{array}$$

Fig. 2.6 Matrices used to compute *fidelity* for the circuit in Fig. 2.1: **a** input vector; **b** result of element-wise product of its ITM and PTM; **c** result of left-multiplication by the input vector

$$\begin{array}{ccc}
 \begin{array}{c} \text{00 01 10 11} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array} & \begin{array}{c} \text{00 01 10 11} \\ \begin{bmatrix} 0.010 & 0.090 & 0.090 & 0.810 \\ 0.010 & 0.090 & 0.090 & 0.810 \\ 0.082 & 0.018 & 0.738 & 0.162 \\ 0.162 & 0.738 & 0.018 & 0.082 \end{bmatrix} \end{array} \\
 \text{(a)} & \text{(b)} & \\
 \begin{array}{c} \text{0 1} \\ \begin{bmatrix} 0.010 + 0.090 & 0.090 + 0.810 \\ 0.010 + 0.090 & 0.090 + 0.810 \\ 0.082 + 0.018 & 0.738 + 0.162 \\ 0.162 + 0.738 & 0.018 + 0.082 \end{bmatrix} \end{array} & \begin{array}{c} \text{0 1} \\ \begin{bmatrix} 0.010 + 0.090 & 0.090 + 0.810 \\ 0.010 + 0.090 & 0.090 + 0.810 \\ 0.082 + 0.738 & 0.018 + 0.162 \\ 0.162 + 0.018 & 0.738 + 0.082 \end{bmatrix} \end{array} \\
 \text{(c)} & \text{(d)} &
 \end{array}$$

Fig. 2.7 Example of the *eliminate_variables* operation: **a** ITM of subcircuit C' from Fig. 2.5; **b** PTM of C' ; **c** output variable h eliminated; **d** output variable g eliminated

elimination of g in the PTM of Fig. 2.7d. However, this time the first column variable is eliminated.

Often, parallel gates have common inputs, due to fan-out at an earlier level of logic. An example of this appears in level L_3 of Fig. 2.5 due to fan-out at level L_1 . The fan-out gate was introduced to handle such situations; therefore, the PTM for level L_1 in Example 2.1 is composed of two copies of the fan-out PTM F_2 tensored with an identity PTM I . However, this method of handling fan-out can be computationally inefficient because it requires numerous matrix multiplications. Therefore, in either inputs or outputs we introduce a new operation called *eliminate_redundant_variables* to remove redundant signals that are due to fan-out or other causes. This operation is more efficient than matrix multiplication because it is linear in PTM size, whereas matrix multiplication is cubic.

Definition 2.7 Given a circuit C with n inputs in_0, \dots, in_{n-1} and PTM M , let in_k and in_l be two inputs that are identified with (connected to) each other. Then $eliminate_redundant_variables(M, in_k, in_l) = M'$, where M' is a matrix with $n - 1$ input variables whose rows are

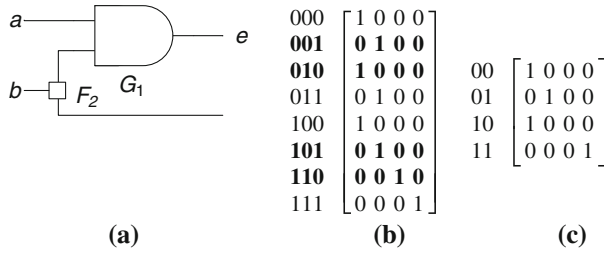


Fig. 2.8 Signal forwarding using *eliminate_redundant_variables*: **a** circuit with signal b fanning out to two different levels; **b** $AND \otimes I$, adding b as an input and output; **c** final ITM for circuit computed by removing rows in boldface

$$M'(i_1 \dots i_k \dots i_{l-1} \ i_{l+1} \dots i_{n-1}, \mathbf{j}) = M(i_1 \dots i_k \dots i_{l-1} \ i_k \ i_{l+1} \dots i_{n-1}, \mathbf{j})$$

The definition of *eliminate_redundant_variables* can be extended to a set of input variables that are redundant. Fig. 2.8 shows an example of this operation.

PTMs yield correct output probabilities despite reconvergent fan-out because the joint probabilities of signals on different fan-out branches are computed correctly using the tensor product and *eliminate_redundant_variables* operations. Suppose two signals on different fan-out branches reconverge at the same gate in a subsequent circuit level. Since the joint probability distribution of these two signals is computed correctly, the serial composition of the fan-out branches with the subsequent gate is also correct, by the properties of matrix multiplication. On the other hand, if the individual signal probabilities are computed separately, then these probabilities cannot be recombined into the joint probability without some loss of accuracy.

The *eliminate_redundant_variables* operation can efficiently handle fan-out to different levels by “signal forwarding,” as seen in Fig. 2.8. Signal b is required at a later level in the circuit; therefore, b is added to the ITM as an output variable by tensoring the AND ITM with an identity matrix. However, tensoring with the identity ITM adds both an input and output to the level. Hence, the additional input is redundant with respect to the second input of the AND gate and is removed using *eliminate_redundant_variables*. Note that the removed rows correspond to assigning contradictory values on identical signals.

2.1.3 Handling Correlations

There are many cases of errors where input and output values cannot be separated and combinations of these values must be taken into account. For example, using the *eliminate_variables* operation, the conditional probabilities of the inputs or outputs cannot always be stored separately in different matrices. While such storage can alleviate the input-space explosion inherent in storing all possible combinations of inputs and outputs, it may not capture correlations within the circuit.

$$\begin{array}{ccc}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.75 & 0.25 & 0 \\ 0 & 0.25 & 0.75 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0.75 & 0.25 \\ 0.25 & 0.75 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.75^2 & (0.75)(0.25) & (0.25)(0.75) & 0.25^2 \\ 0.25^2 & (0.75)(0.25) & (0.25)(0.75) & 0.75^2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
\text{(a)} & \text{(b)} & \text{(c)}
\end{array}$$

Fig. 2.9 Example of output inseparability: **a** PTM for a probabilistic wire-swap; **b** PTM for each individual output after applying *eliminate_variables*; **c** incorrect result from tensoring two copies of the PTM from part **b** and applying *eliminate_redundant_variables*

Example 2.3 Suppose two wires have a 0.25 probability of swapping. The matrix corresponding to this error is given in Fig. 2.9a. If we try to separate the probability of each output, using *eliminate_variables*, the output probabilities both have the PTM of Fig. 2.9b. If these outputs are tensored (with redundant inputs eliminated), they result in the erroneous combined matrix of Fig. 2.9c. This demonstrates that these two outputs cannot be correctly separated; their joint conditional distributions are, in fact, inseparable.

Just as some errors cannot be separated, some faults affect multiple gates simultaneously. In this case, the combined PTM cannot be built from individual PTMs, and the joint probabilities must be obtained (or the exact correlation determined). This same effect can occur with input vectors that cannot always be separated into probabilities of individual inputs. An example is given below.

$$\begin{array}{c}
00 \ 01 \ 10 \ 11 \\
[0.5 \ 0 \ 0 \ 0.5]^T
\end{array}$$

PTMs have the advantage that, at every level, they can represent and manipulate joint probabilities from the inputs to the outputs. If necessary, individual output distributions can be obtained using the *eliminate_variables* operation.

So far, we have introduced the PTM representations of gate and wire constructs, and the operations needed to combine them into circuit PTMs. In the next section, we give examples of the various kinds of faults that PTMs can capture, as well as the application of PTMs to soft-error analysis and error-threshold computation.

2.2 Applications

In this section, we discuss applications of PTMs to various fault types as well as in determining the error-transfer behavior of logic circuits.

000	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	000	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	000	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	000	$\begin{bmatrix} 0.95 & 0.05 \end{bmatrix}$	000	$\begin{bmatrix} 1 & 0 \end{bmatrix}$
001	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	001	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	001	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	001	$\begin{bmatrix} 0.95 & 0.05 \end{bmatrix}$	001	$\begin{bmatrix} 0 & 1 \end{bmatrix}$
010	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	010	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	010	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	010	$\begin{bmatrix} 0.95 & 0.05 \end{bmatrix}$	010	$\begin{bmatrix} 0 & 1 \end{bmatrix}$
011	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	011	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	011	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	011	$\begin{bmatrix} 0.05 & 0.95 \end{bmatrix}$	011	$\begin{bmatrix} 1 & 0 \end{bmatrix}$
100	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	100	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	100	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	100	$\begin{bmatrix} 0.05 & 0.95 \end{bmatrix}$	100	$\begin{bmatrix} 0 & 1 \end{bmatrix}$
101	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	101	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	101	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	101	$\begin{bmatrix} 0.95 & 0.05 \end{bmatrix}$	101	$\begin{bmatrix} 1 & 0 \end{bmatrix}$
110	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	110	$\begin{bmatrix} 1 & 0 \end{bmatrix}$	110	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	110	$\begin{bmatrix} 0.05 & 0.95 \end{bmatrix}$	110	$\begin{bmatrix} 1 & 0 \end{bmatrix}$
111	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	111	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	111	$\begin{bmatrix} 0 & 1 \end{bmatrix}$	111	$\begin{bmatrix} 0.05 & 0.95 \end{bmatrix}$	111	$\begin{bmatrix} 0 & 1 \end{bmatrix}$
(a)		(b)		(c)		(d)		(e)	

Fig. 2.10 PTMs for various types of gate faults: **a** a fault-free ideal 2-1 MUX gate (select line is the 3rd input); **b** first input signal stuck-at 1; **c** first two input signals swapped; **d** probabilistic output bit-flip with $p = 0.05$; **e** wrong gate: MUX replaced by 3-input XOR gate.

2.2.1 Fault Modeling

The PTM model can represent a wide variety of faulty circuit behaviors, including both hard and soft physical faults, and design errors. The fact that there are separate probabilities for each input and output, and the fact that they are propagated simultaneously make this possible. Fig. 2.10 lists some fault/error types that can be represented by PTMs.

Figure 2.10a shows the ITM for a fault-free ideal 2-1 multiplexer (MUX). Fig. 2.10b shows the first data input signal of the MUX stuck-at 1, i.e., row 000 is replaced with row 100 of the ITM, row 010 with row 111, and so forth. Fig. 2.10c shows an example where the first two wires have been swapped; this is captured by permuting the rows of the ITM, accordingly. Fig. 2.10d shows the first example of a probabilistic error, an output bit-flip where the wrong value occurs with probability $p = 0.05$ in each row. Fig. 2.10e shows a design error where a MUX has been replaced by a 3-input XOR gate. As these examples indicate, PTMs can capture both gate and wiring errors.

PTMs can also represent faults that are likely to occur in nanoscale circuits. For instance, in QCA, the wires themselves are made of “quantum dots,” and so, like gates, wires can experience bit-flips. Such bit-flips on wires can be represented by the 1-input identity gate I , with probabilities as shown below.

$$\begin{bmatrix} 1-p & p \\ 1 & 1-q \end{bmatrix}$$

2.2.2 Modeling Glitch Attenuation

Thus far, signals have been described by their logic value, with each signal represented by a 1×2 vector that indicates the probability of it being 0 or 1. While retaining

the discreteness of our model, we now expand signal representation to incorporate some necessary electrical characteristics.

For instance, we can differentiate between signals of long and short duration, just as we differentiate between signals with high and low amplitude by their logic value. We can represent a signal by a vector w which has four entries instead of two, $w = [p_{0s} \ p_{0l} \ p_{1s} \ p_{1l}]$. The second bit of the row index represents short (“s”) or long (“l”) duration, so p_{0s} is the probability of a logic 0 with short duration. Extraneous glitches, such as those induced by SEUs, are likely to have short duration, while driven logic signals are likely to have relatively long duration.

Each gate in a circuit has a probability of an SEU strike that depends upon various environmental factors, such as neutron flux and temperature. We call this the *probability of occurrence* for a gate (or node) g , and denote it by $p_{\text{occur}}(g)$. However, SEU strikes create glitches which can be differentiated by a combination of shape and amplitude. These distinctions are important for the propagation of a glitch through circuit gates. Therefore, we utilize a modified identity matrix denoted $I_{1,n}(p_{\text{occur}})$ to represent a probability distribution on a glitch induced by an SEU strike.

We use the glitch-propagation model from [9] to determine which signal characteristics to capture; a different model might require other characteristics to be represented. In [9], glitches are classified into three types depending on their duration D and amplitude A , relative to the gate propagation delay T_p , as well as the threshold voltage V_t . Glitches are assumed to change only logic 0 to logic 1 when they occur, but they can be inverted later.

- Glitches of type 1 have amplitude $A > V_t$ and duration $D > 2T_p$. Glitches of this type are propagated without attenuation.
- Glitches of type 2 have amplitude $A > V_t$ and duration $2T_p > D > T_p$. Glitches of this type are propagated with an attenuated amplitude of $A' < A$.
- Glitches of type 3 have $A < V_t$. Glitches of this type are not propagated, i.e., they are electrically masked.

Since amplitude is already indicated by the logic value, an additional bit is used to indicate whether the duration is larger or smaller than the propagation delay of the gate (when the amplitude is higher than the threshold voltage). The duration is irrelevant for glitches with amplitude lower than the threshold voltage, since these are likely to be attenuated. Fig. 2.11a shows the probability distribution of an SEU strike when the correct logic value is 0. Glitches of type 1 are indicated by row labels 11, glitches of types 2 are indicated by labels 10, and glitches of type 3 are indicated by 01. In particular, Fig. 2.11a assumes uniform distribution with respect to glitches.

Once an SEU strikes a gate g and induces a glitch, the electrical characteristics of the circuit gates determine whether the glitch is propagated. Glitches with long duration and high energy relative to the gate propagation delay and threshold voltage are generally propagated; other glitches are normally quickly attenuated. We call the probability that a glitch is propagated $p_{\text{prop}}(g)$. The glitch-transfer characteristics of a logic gate are described by a modified gate PTM that represents relevant characteristics of the glitch. For instance, Fig. 2.11b shows a modified *AND* PTM, denoted $AND_{2,2}(p_{\text{prop}})$.

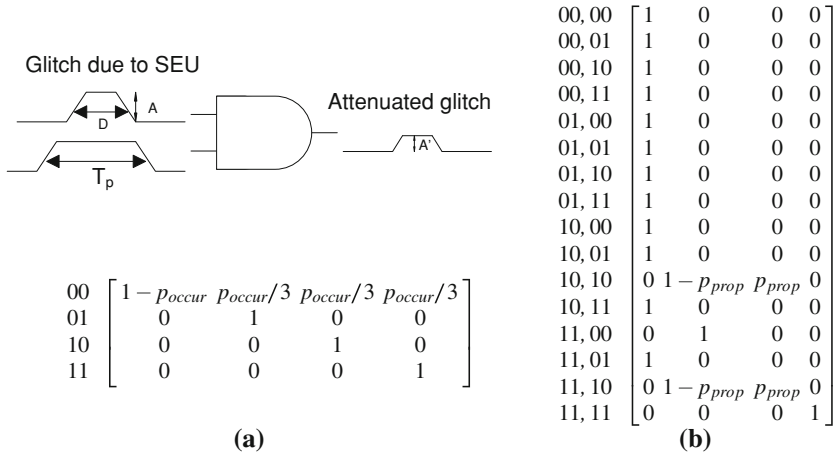


Fig. 2.11 PTMs for SEU modeling where the row labels indicate input signal type: **a** $I_{2,2}(p_{\text{occur}})$ describes a probability distribution on the energy of an SEU strike at a gate output, **b** $AND_{2,2}(p_{\text{prop}})$ describes SEU-induced glitch propagation for a 2-input AND gate. The type-2 glitches become attenuated to type 3 with a probability $1 - p_{\text{prop}}$

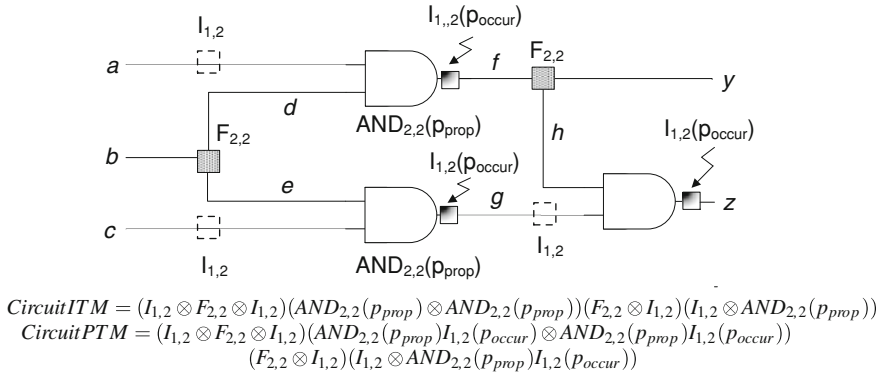


Fig. 2.12 Circuit with ITM and PTMs describing an SEU strike and the resultant glitch propagation with multi-bit signal representations

In the selected glitch model [9], attenuation acts by transforming sensitized glitches of type 2 with a certain probability, into glitches of type 3. All other signals retain their original output values given by the logic function of the gate. This transfer function can be described by the PTM of Fig. 2.11b. This PTM shows an AND gate which propagates an input glitch (only if the other input has a non-controlling value), with certainty if the glitch is of type 1 (in which case it is indistinguishable from a driven logic value) or with probability p_{prop} if the glitch is of type 2.

When using 2-bit signal representations, the probability of a logic 1 value for a signal is computed by *marginalizing*, or summing-out, over the second bit. For

Fig. 2.13 Circuit used in Example 2.4 to illustrate the incorporation of electrical masking into PTMs

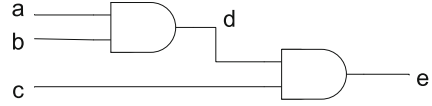


Fig. 2.14 PTM incorporating electrical properties of gates for the circuit in Example 2.4

$$fidelity = .99994791$$

$$P_{error} = 1 - fidelity = .000052083$$

1	0	0	0
⋮			
1	0	0	0
0.9996	0.0001	0.0003	0
0.9996	0.0002	0.0002	0
1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0
0.2500	0.1875	0.5625	0
0.2500	0.3750	0.3750	0
1	0	0	0
1	0	0	0
0.5000	0.1250	0.3750	0
0.5000	0.2500	0.2500	0
1	0	0	0
1	0	0	0
0.9995	0.0002	0.0003	0
0.9995	0.0001	0.0001	0.0003
1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0
0	0.1250	0.3750	0
0	0.2500	0.2500	0
1	0	0	0
1	0	0	0
0	0.5000	0.5000	0
0	0	0	1

instance, if a signal has 2-bit distribution $[0.2 \ 0.1 \ 0.3 \ 0.4]$, since the second bit indicates duration, the probability of a logic 0 is $0.2 + 0.1$ and the probability of a logic 1 is $0.3 + 0.4$. Fig. 2.12 shows a circuit with the corresponding ITM and PTMs with multi-bit signal representations.

Example 2.4 For the circuit in Fig. 2.13, suppose an *SEU* strike produces a glitch at input b . By inspection, we see that this glitch will only propagate to primary output e for the primary input combination 101. In other words, the glitch propagates if the input sensitizes the appropriate path to d and then e . If we let $p_{occur} = 0.001$ and $p_{prop} = 0.5$, and $AND_{2,2}(p_{prop})$ is as shown in Fig. 2.11, then the circuit PTM is given by:

Table 2.1 Polynomial approximations of circuit and residual errors. The fitted polynomials are of the form $e(x) \approx a_0 + a_1x + a_2x^2 + a_3x^3 \dots$

Circuit	Error	Polynomial coefficients						
		a_0	a_1	a_2	a_3	a_4	a_5	a_6
Majority	2.5 E−7	0.2080	0.1589	0	0	0	0	0
MUX	6.6 E−6	0.0019	1.9608	−2.8934	1.9278	0	0	0
Parity	0.0040	0.0452	5.4892	−21.4938	31.9141	−4.2115	−30.3778	19.5795
tcon	0.0019	0.0152	6.2227	−13.5288	7.1523	9.2174	−9.0851	0
9symml	0.0010	0.0250	2.4599	−3.7485	1.5843	0	0	0
XOR5	0.0043	0.0716	5.9433	−26.4666	51.1168	−44.6143	14.4246	0

$$(I_2 \otimes I_{2,2}(p_{\text{occur}}) \otimes I_2)(AND_{2,2}(p_{\text{prop}}) \otimes I_2)(AND_{2,2}(p_{\text{prop}}))$$

The corresponding PTM and fidelity are given in Fig. 2.14.

2.2.3 Error Transfer Functions

In this section, we analyze circuit reliability as a function of gate reliability. Using data points for various gate error values, we derive low-degree polynomial approximations for the error transfer functions of some benchmark circuits. Such functions can be used to derive upper bounds for tolerable levels of gate error.

Definition 2.8 The *error transfer function* $e(x)$ on $0 \leq x \leq 1$ of a circuit C is the *fidelity* of C with output-error probability x on all gates.

Figure 2.15 illustrates the error-transfer functions for several standard benchmark circuits, determined by introducing varying amounts of error into gates and then calculating the circuit fidelity according to Definition 2.5. Generally, such error transfer curves can be described by polynomials. If two gates experience errors with probability $p > 0$, then their serial and parallel compositions experience errors with probability $O(p^2)$. If a circuit has n gates, each with error p , then its fidelity is a polynomial in p of degree n . Realistically, only gate error values under 0.5 are useful since the gate can simply be viewed as its negated version for higher error values. However, Fig. 2.15 uses probabilities of gate error up to 1 to emphasize the polynomial nature of the curves.

Table 2.1 gives low-degree polynomials that estimate error transfer functions with high accuracy. Such functional approximations are useful in determining the upper bounds on gate error probability necessary to achieve acceptable levels of circuit error. For instance, it has been shown that replication techniques such as TMR or NAND-multiplexing only decrease circuit error if the gate error is strictly less than 0.5 [10]. However, Fig. 2.15 suggests that for most circuits, replicating the entire circuit at gate errors of 0.20 or more will only increase circuit error.

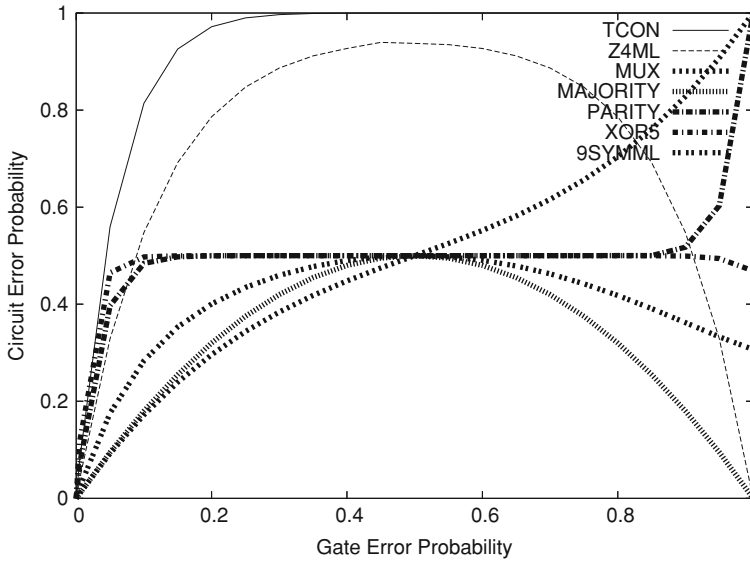


Fig. 2.15 Circuit error probability under various gate error probabilities

References

1. Kumamuru RK et al (1993) Operation of quantum-dot cellular automata (QCA), shift registers and analysis of errors. *IEEE Trans Electron Devices* 50–59:1906–1913
2. Rejimon T, Bhanja S, (2006) Probabilistic error model for unreliable nano-logic gates. In: *Proceedings of NANO*, pp 47–50
3. Parker KP, McCluskey EJ (1975) Probabilistic treatment of general combinational networks. *IEEE Trans Comput C-24*(6):668–670
4. Ercolani S et al. (1989) Estimate of signal probability in combinational logic networks. In: *Proceedings of European test conference*, pp 132–138
5. Savir J, Ditlow G, Bardell PH (1983) Random pattern testability. In: *Proceedings of FTCS*, pp 80–89
6. Krishnaswamy S, Viamontes GF, Markov IL, Hayes JP (2005) Accurate reliability evaluation and enhancement via probabilistic transfer matrices. In: *Proceedings of DATE*, pp 282–287
7. Krishnaswamy S, Viamontes GF, Markov IL, Hayes JP (2008) Probabilistic transfer matrices in symbolic reliability analysis of logic circuits. *ACM Trans Des Autom Electron Syst* 13(1):1–35 article 8
8. Hachtel G, Somenzi F (1996) *Logic synthesis and verification algorithms*. Kluwer Academic Publishers, Boston
9. Omana M et al. (2003) A model for transient fault propagation in combinatorial logic. In: *Proceedings of IOLTS*, pp 11–115
10. Pippenger N (1998) Reliable computation by formulas in the presence of noise. *IEEE Trans Inf Theory* 34(2):194–197

Design, Analysis and Test of Logic Circuits Under
Uncertainty

Krishnaswamy, S.; Markov, I.L.; Hayes, J.P.

2013, XII, 124 p., Hardcover

ISBN: 978-90-481-9643-2