Logic Circuit Testing for Transient Faults

Smita Krishnaswamy, Igor L. Markov, and John P. Hayes
{smita,imarkov,jhayes}@eecs.umich.edu
Advanced Computer Architecture Lab, University Of Michigan, Ann Arbor 48109-2122

Abstract

Transient faults are becoming an increasingly serious concern for logic circuits. They can be caused by thermal neutrons, present at all altitudes, and by other types of ionizing radiation, especially in aerospace applications and nuclear engineering. In this paper we examine issues related to detection of transient errors. The difficulty in testing for transient errors is that they are not always present. Test vectors need to be repeated a number of times in order to detect a fault. We show how to compute a measure for the detectability of transient faults with respect to specific test vectors. This is done using a matrix-based gate-fault model known as the probabilistic transfer matrix model. Using this detectability measure we derive methods to generate multisets of tests to verify probability distributions of faults and detect abnormalities in circuit behavior. Applications of this method include detection of increased atmospheric radiation in terms of its impact on circuits, and testing for process variation that increases the susceptibility of a circuit to transient errors.

1 Introduction

In VLSI circuits, transient errors, also known as soft errors, occur due to external radiation or electrical noise. They are of increasingly serious concern due to the downscaling of device features and lowering of supply voltage. The susceptibility to external radiation can be increased by certain manufacturing defects such as abnormally large gate area, corrosion in the metal shield of a chip, thin wires, embedded dust particles etc. New manufacturing processes can also lead to new types of defects which indirectly affect susceptibility to transient errors. For instance, immersion lithography is currently being used to get finer resolutions for smaller features since water has refractive index greater than that of air [1]. However, water-induced defects due to air bubbles can cause light to refract and this can affect features size as well.

It is difficult to test for transient errors since these errors are not consistently present. Faults may not coincide with the times when test vectors are applied to detect them. Furthermore, even if we detect certain transient errors, it is not clear how to interpret the results of such a test. Detection could signal an anomalous event, or it could indicate a regular pattern. In this paper, we address these questions by formulating the problem of transient error detection and by recognizing that *fault probabilities* rather than the faults themselves should be detected during such tests. Fault probabilities can give us information about the error rate in given circuits. Then we derive test sets which increase the probability of fault detection and facilitate the estimation of the overall error rate.

Permanent stuck-at faults can be detected reliably by certain input vectors. On the other hand, transient faults can only be detected with a probability that depends on the frequency of their occurrence. Moreover, if multiple faults are possible, the fault detection probability is not the same as the occurrence probability. In this work, the probability of detection is calculated using the PTM or probabilistic transfer matrix model from [3, 7]. We also develop specialized computational machinery for circuit test, which scales better than the methods of [4], where testing was not considered.

We use previously validated formulas to estimate the probability of occurrence of a fault accounting for environmental factors such as altitude, neutron flux, and gate area [6]. Then, based on this probability we derive a small number of repetitions for particular test vectors to enable fault detection with sufficiently high probability. Repetitions of tests can be used to ensure a desired level of fault tolerance. For instance, if the fault is detected more times than expected, this can indicate an increase in the error rate.

The key contributions of this work are as follows:

- We propose a measure called the *probabilistic detectability* of a transient fault with respect to a test vector to determine the likelihood with which a fault is detected by each vector.
- Given a set of faults, we show how to derive multisets of test vectors for various objectives (we call them *multitests*). One objective is to minimize the test time, and therefore the cardinality of the test set; another is to achieve maximum fault detection with a limited budget of tests.

Our work also suggests the possibility of comparing expected detection rates to actual detection rates. A change in detection frequency can be caused by environmental conditions or a violation of assumptions in the probability analysis such as the independence of gate faults. The results produced by our techniques can be interpreted in such circumstances as well, and test vectors can be applied more often to improve the detection rate.

The remainder of this paper is organized as follows. Section 2 discusses relevant prior work in this area, Section 3 shows how to derive fault detection probabilities and discusses computational issues. Section 4 gives various methods of test set derivation. Section 5 gives conclusions.

2 Previous Work

Previous research related to transient fault testing includes estimating transient error probabilities in circuits, n-detection test sets, and online tests for transient faults.

In [9] the authors discuss factors that cause transient errors in combinational logic to be latched. There are three possible masking mechanisms that can prevent a transient error from being latched: logical masking, electrical masking and latching window masking. Electrical masking occurs when the charge of the particle of external radiation is not high enough for the signal value to flip. Transient errors have to carry charge greater than Q_{crit} or the critical charge for the signal to change value. Latching window masking occurs when the transient pulse occurs outside the part of the clock cycle when flip-flops are latching. Logical masking occurs when the error does not propagate through the circuit logic to any of the primary outputs. In [6] the *soft error susceptibility* of a gate g with respect to a latch l, is calculated by $p_{error}(g,l) = R_{SEU} * P_{sense}(g,l) * P_{latched}(g,l)$. Here R_{SEU} is the probability that an incident charged particle can produce an SEU; this takes into account the flux and the gate area to determine if the total charge is greater than Q_{crit}. $P_{sense}(g,l)$ is the probability that an input sensitizes a particular gate. $P_{latched}(g,l)$ is the probability that the error occurs during the latching window of the clock cycle.

The concept of generating tests which detect a fault with a high probability is related to the notion of *n-detection test sets*, where each fault in a circuit is detected n times by the test set. In [8] the authors adapt conventional algorithms for generating compact test sets into n-detection algorithms. One of their algorithms for generating a compact n-detection test set proceeds by including every essential test and then repeatedly selecting the additional test vector t_i which has the largest value of $m(t)$. The function $m()$ counts the total number of faults that t detects which have not yet been detected n times.

While the task of detecting a transient fault with a high probability can be compared to detecting a fault n times, there are key differences. In the n-detection framework, any test can only detect a fault once; this imposes an added discreteness to the problem. However, a particular test can detect various transient faults with a spectrum of probabilities. Therefore in our framework a measure such as $m(t)$ would need to be modified to sum the total probability with which any fault, not yet detected with probability p can be detected. Also each test vector is only repeated once in an n-detection framework while a test vector can be repeated many times to detect a transient fault. One seeks to use fewer tests (including possible repetitions) while covering a given fault set.

The rate of transient errors may temporarily increase, e.g., during a solar flare or due to noise in the power supply voltage. Therefore, mechanisms to embed tests for these types of errors in chips are important, especially in mission critical applications. Work has been done in the area of online testing of transient faults for specialized circuit technologies. In [5], it was shown that Domino circuits can be tested for transient faults using self-checking circuits because they do not cause bidirectional errors, but for technologies like FPGAs and FCMOS, new strategies are necessary. Built-in tests are often generated using LFSRs that feed patterns into scan chains. To this end, the work in [10] modifies random LFSR patterns by a few bits to generate complete test sets for a set of faults. Using such a scheme one can alter an LFSR cycle so that it repeats the tests required for coverage by modifying redundant test bits.

3 Transient Fault Detectability

Transient errors can be detected by test vectors if they are applied in the same cycle in which the transient error is latched. Therefore, the probability of a test vector detecting a transient fault is proportional to the probability of the transient fault occurring at any particular clock cycle. However, when there are multiple transient errors (which may have a single physical cause), or if there are unreliable circuit components, the probability of a test vector detecting an error is not the same as the probability of occurrence. The detection probability depends on the interaction of the multiple errors and/or the observability of the errors under unreliable component behavior.

In this section we develop a way to accurately determine the detection probability of sets of independent transient errors with respect to particular test vectors. We use the matrix representation given in [7] to represent transient error probabilities at gates and signals.

3.1 The PTM Model

The PTM framework was first introduced in [7], but the underlying concept can be traced back to [3]. The probabilistic behavior of a gate is described by a matrix M, where the (j,k)th entry represents the probability of output signals $O = o_0, o_1, \ldots o_n$ having value k given that input signals $I = i_0, i_1 \ldots i_m$ have value j. This is denoted $p(O = k|I = j)$. Here the row and column indices j and k are thought of as bit vectors whose entries represent the values of the signals that form the input and output. For convenience we omit the signal names and write this as $p(k|j)$. For instance $p(1,1|1,0)$ represents the probability that the two output variables (O_0, O_1) have value $(1,1)$ given that the two input variables (I_o, I_1) have value $(1,0)$. Probability $p(1,1|1,0)$ can be alternately written as $p(3|2)$.

Definition 1 *The matrix M where the (j,k)th entry represents the probability of output value k given input value j, i.e., $p(k|j)$, is called a* probabilistic transfer matrix (PTM)*. A fault-free circuit has an* ideal transfer matrix (ITM)*, i.e., the correct value of the output occurs with probability* 1.

$$\begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \begin{bmatrix} p & (1-p) \\ p & (1-p) \\ p & (1-p) \\ (1-p) & p \end{bmatrix}$$

Figure 1: The ITM and a PTM for a NAND gate.

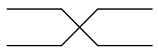
Similarly, an input vector v is a row vector representing the joint probability distribution of the input signals. The ith entry of v denoted $v(i)$ gives the probability that the input signals have values represented by the bit vector i. The output probability distribution after input vector v is evaluated on gate g with PTM P_g is given by vP_g. (See Figure 2)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p & (1-p) \\ p & (1-p) \\ p & (1-p) \\ (1-p) & p \end{bmatrix} = \begin{bmatrix} p & (1-p) \end{bmatrix}$$

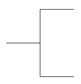
Figure 2: Test vector evaluation at a faulty NAND gate.

$$\begin{bmatrix} .05 & .95 \\ .05 & .95 \\ .05 & .95 \\ 1 & 0 \end{bmatrix}$$

PTMs can be used to model transient errors by inserting error probabilities into their entries. For instance, if a NAND gate has output stuck-at-0 with probability 0.05 then the



$$\begin{bmatrix} 1&0&0&0 \\ 0&0&1&0 \\ 0&1&0&0 \\ 0&0&0&1 \end{bmatrix}$$



$$\begin{bmatrix} 1&0&0&0 \\ 0&0&0&1 \end{bmatrix}$$

Figure 3: ITMs of a wire swap and a fanout network.

corresponding PTM is given by the matrix on the right. In this matrix, entries where the correct output value is 0, are left unchanged while entries with correct output value 1 have an error of 0.95. In this way we can introduce independent transient errors simultaneously into several gates.

Starting with gate PTMs, we can calculate the PTM of an entire circuit, using a few construction rules in terms of matrix multiplications and tensor products. Since the latter has not often been used in this context, we define it formally.

Definition 2 *Given two matrices M_1 and M_2 with dimensions $m \times n$ and $o \times p$ respectively, the* tensor product *of M_1 and M_2, denoted $M_1 \otimes M_2$ is an $mo \times np$ matrix whose entries are given by $p(k|j) = p(k_1|j_1)p(k_2|j_2)$.*

The combination rules for PTMs are as follows:

- If two gates g_1 and g_2 with PTMs P_1 and P_2 are connected in series then their combined PTM is by P_1P_2.
- If two gates g_1 and g_2 with PTMs P_1 and P_2 are connected in parallel then their combined PTM is $P_1 \otimes P_2$.

Wiring subnetworks can be treated as a special class of gates and represented by ITMs just like other gates; wire permutations form a special class of permutations which we call *variable permutations*. Variable permutation matrices are induced by permuting bits of row indices and column indices (these bits are known as row variables and column variables) of the identity matrix. The wire swap matrix for two adjacent wires is shown in Figure 3. Similarly, a fanout can be represented by an ITM of the kind also shown in Figure 3.

Definition 3 *For a matrix A consider a permutation $\sigma(\ )$ on the row variables. A row of A with index $i_0i_1 \dots i_n$ in binary is mapped to the row with index $\sigma(i_0)\sigma(i_1)\dots\sigma(i_n)$ in binary. This defines a* row variable permutation *of A with respect to σ, which we denote as $rperm(A,\sigma)$.*

3.2 Detectability

Combining gate PTMs into circuit PTMs is useful because the latter can provide information about the detectability of various faults in a circuit. A detectable fault in a gate causes an erroneous gate output. Therefore, the detectability of a fault can be thought of as a measure of the effect of the fault on the primary output.

Definition 4 *The* detectability *of a fault set $F = \{f_1, f_2, \dots f_n\}$ in circuit C with PTM M_F and ITM M is defined as the total probability that the output is erroneous given that faults f_i exist with probability p_i. Let $M_f \times_e M$ be the entry-wise product of M with M_f. The total probability of output error is given by the sum of the entries of $\frac{1}{p}M_f \times_e M$.*

The detectability of a fault can be calculated by obtaining the circuit PTM when the gate g has error probability p, finding the total probability of error, and then multiplying by $1/p$. The detectability of a fault with respect to a test vector is the probability with which the fault is exposed in the output for a particular test vector. For a test vector t, this is given by

$$p_{det}(f,t) = tM_f \times_e M$$

Table 1 illustrates the detectability of multiple faults in the circuit C17 of Figure 5 where there is an error probability of 0.05 associated with every gate input. Note that if there is only one probabilistic stuck-at fault in a circuit, and this fault occurs with probability p, then any vector that detects the fault also detects it with probability p. Therefore the PTM formulation in this case provides information about whether or not the fault is detectable. In the case where there are multiple errors, the detection probability is generally different from the occurrence probability of an individual fault. Undetectable faults have detectability zero for each test vector.

3.3 Computational Method

Earlier we showed how to calculate circuit PTMs and then extract detectability information for particular vectors. A computational method for this process is given in [4]. However, the entire circuit PTMs do not need to be computed in order to evaluate particular test vectors. Therefore we propose an alternate, more scalable method for evaluating the test vectors. As before, we represent gate faults by a PTM. Then, for each input vector t we compute the output vector by evaluating in topological order the output of each gate and then feeding these outputs into new inputs in turn. The final output vector $o_{t_{err}}$ is then be compared to the ideal output vector $o_{t_{ideal}}$ in order to calculate $p_{exp}(f,t)$.

$$p_{exp}(f,t) = o_{t_{ideal}} \times_e o_{t_{err}}(i)$$

Calculating in this way avoids some unnecessary computation. This is because circuit PTMs include output information for all possible input combinations, but each test vector corresponds to a single row in the circuit PTM. The proposed technique scales test vector evaluation to much larger circuits than the complete PTM evaluation method described in [4]. The topological ordering ensures that each gate is only evaluated after its inputs are ready. Before a gate is evaluated, its inputs are tensored together so that one input vector is multiplied with the gate. A complication occurs when the input distributions of two signals are coupled. As shown in Example 1 a fanout gate has two or more outputs but the output signals are highly correlated with each other. Therefore, when we have 2-output gates, it is possible that their input distributions cannot be separated since there is a possibility that the joint probabilities of the signals are correlated. While individual signal probabilities can always be combined to form a joint probability vector, the converse is not always true. There are cases where the joint probability vector cannot be split into two or more individual probability vectors because the distribution of signals can be coupled with a high correlation as shown in the example below.

Example 1 *When input vector $i = [0.5\ 0.5]$ is multiplied by the fanout PTM shown in Figure 3, the output vector is $o = [0.5\ 0\ 0\ 0.5]$. However, if we abstract the two output variables and express their states separately we get*

```
store input signals
topologically sort circuit
for each gate
{
  current_inputs = NULL
  for each gate input
   if entangled
    {
     for each entangled signal
      {
      add signal to current_inputs vector
      add signal to gate_outputs
      $gate\_PTM = gate\_PTM \otimes Identity$
      }
    }
permute gate to match current_inputs order
tensor signals in current_inputs order for gate_input
gate_outputs = gate_input × gate_PTM
Store gate_outputs as entangled signals
Existentially abstract any output signals without sinks
}
```

Figure 4: Procedure for evaluating detectability of a fault with respect to a test vector.

$o_0 = [0.5\ 0.5]$ and $o_1 = [0.5\ 0.5]$. This corresponds to a joint distribution of $[0.25\ 0.25\ 0.25\ 0.25]$ which is not the original joint distribution.

In order to avoid the foregoing situation, we keep track of potentially correlated signals from the same source gate, in tensored form. If only one of these signals is an input for a certain gate g, then g is modified to take in both signals as input. The gate g also outputs the extraneous signal, and its PTM can be computed by tensoring the PTM of g with an identity matrix, as shown below.

Example 2 *In the context of Example 1, suppose gate g with PTM M receives input o_1, and another input o_2, and outputs z. Since o_1 and o_0 have correlated probability distributions, the PTM for g needs to be modified to take the joint distribution of o_0 an o_1 as input. This is done by tensoring M with a 2×2 identity matrix.*

$$M' = I \otimes M$$

The modified gate has o_0 and o_1 as inputs and o_0 and z as outputs. Essentially, it leaves o_0 unchanged and causes the signals o_0 and z to be correlated. If o_0 has no sinks left, then it can be abstracted out.

A test vector evaluation method must take correlated signals into account by storing the appropriate signals jointly and making the appropriate modification when evaluating sink-gates of correlated signals. It is necessary to permute the inputs of gate g to change the order of the input wires, before tensoring with the identity so that the correlated signals form adjacent variables in cases where the original correlated input occurs in the middle of the gate. Figure 4 gives the pseudo-code for the algorithm used to evaluate test vectors.

Table 2 shows the average runtime and memory requirements for evaluating the output probabilities for various input vectors on standard benchmark circuits from the ISCAS-85 suite. These simulations were conducted on a Linux workstation with Intel Xeon CPU 2.0GHz processor and cache size 512 KB. Table 2 shows two sets of results. The first includes runtime and memory for computations with ideal circuits, i.e., all gates having error probability zero. The second set shows the corresponding results for circuits with all gates having error probability .05 at every input. Faults at the input pins demonstrate the worst case time complexity for evaluating a test vector. This is because faults at the input cause multiple paths to be evaluated starting at the beginning of the evaluation rather than after an erroneous gate at a later level of the circuit.

The first set of run times are generally smaller because no errors are present and the computation involves only 0s and 1s. Similarly, the PTM representing any gate has a single non-0 entry per row. Our results show that the runtime and memory usage for circuits with ideal gates vary with the number of gates in the circuit. In the case of faulty gates, these parameters can vary further depending on the input vector under evaluation. Some input vectors cause probabilistic outputs that represent the occurrence and non-occurrence of gate faults. This leads to multiple paths in the circuit that contribute to the probability of the same output combination. Our techniques perform such summations implicitly, but runtime generally depends on the total number of paths present.

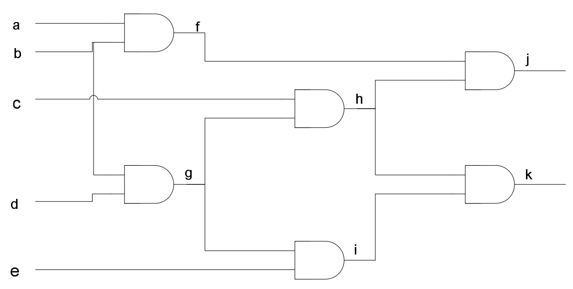


Figure 5: The small ISCAS-85 circuit C17.

Test vector	Detectability
00000	.225
11111	.185
11110	.186
11101	.205
11011	.215
10111	.215
01111	.186

Table 1: Detectability of a multiple fault in the circuit in Figure 5, with respect to various test vectors.

4 Test Set Generation

In the previous section, we showed how to compute the detectability of transient errors for particular test vectors based on transient fault probabilities. Using this information, we can derive *multitests*, i.e. test sets with repetition, to increase the likelihood of detecting or confirming a fault probability. Suppose, a transient error f occurs with probability p according to previously validated estimates. If t detects the error with probability $q = p_{det}(f,t)$, then q is a binomial random variable. If we repeat the test t twice, the probability of obtaining one detection is $q^2 + 2q(1-q) = 1-(1-q)^2$. Therefore if we want to confirm the estimated probability for fault f we can repeat the test n times such that

Circuit	Characteristics			Ideal circuits		Erroneous Circuits	
	inputs	outputs	gates	time, s	memory, MB	time, s	memory, MB
C432	36	7	160	0.28	0.7	0.73	0.8
C499	41	32	202	0.30	0.2	0.36	1.2
C880	60	26	383	0.47	0.4	52.50	124.0
C1355	41	32	546	1.44	0.1	0.22	0.6
C1908	33	25	880	0.76	1.1	11.70	42.2
C3540	50	22	1669	1.48	2.2	131.50	547.1
C6288	32	32	2416	2.12	3.3	50.90	44.8

Table 2: Runtime and memory usage for the detectability evaluation for benchmark circuits with all gates having error probability 0 in the ideal case and .05 in the erroneous case.

$1-(1-q)^n > p_{th}$ to increase the probability of detection to some threshold probability. If there are no detections after repeating the test n times, then we can be fairly certain that the probability of transient error does not exceed the estimate. For m detections, we can calculate the probability with which n repetitions yield m detections as $\binom{n}{m} q^m (1-q)^{n-m}$. Depending on this probability we can decide if the circuit is experiencing more transient errors than normal

Similarly, given a set of test vectors for a set of multiple faults, we can derive a multitest to compare the fault probabilities to a given upper bound. Different test vector sets can be developed for different practical objectives. We first discuss a method for generating a small multitest to detect a set of faults with a desired detection probability. This *test vector repetition algorithm* first calculates $p_{det}(f_i,t)$, as described earlier. Given a desired detection probability $p_{th}(f_i)$ we calculate $d(f_i)$ for each fault f_i, which can be thought of as the probability with which fault f_i is already detected. We then rank each test vector t in descending order by the value $r(t)$, where

$$r(t) = \sum_{all\ i:d(f_i)>p_{th}} max(p_{det}(f_i,t) * d(f_i), 1-p_{th}(f_i))$$

Initially we set $d(f_i) = 1$ for all i. Then we calculate a "repetition list" which gives the number of times to repeat each vector t .

1. Initialize the repetition list to 0 for all vectors t in the test set.
2. Calculate $r(t)$ for all t.
3. If for all f_i, $d(f_i) \leq p_{th}(f_i)$ then the procedure terminates, and the repetition list is the desired multitest.
4. Select item t_i with minimum value of $r(t_i)$ and increment its value in the repetition list.
5. For all f_i, set $d(f_i) = d(f_i) * p_{det}(f_i,t)$.
6. Recalculate $r(t) = \sum_i d(f_i) * p_{det}(f_i,t)$.
7. Return to step 3.

Example 3 *Consider again the circuit C17 in Figure 5. For the set of faults consisting of all primary inputs stuck at* 1 *with probability* .05 *each, the multitest is* $\{00111(28), 01011(28),\ 01110(28), 01101(28)\}$, *with repetitions given in parenthesis. For the set of faults consisting of primary outputs stuck at* 0 *with probability* .05 *each, the multiset is* $\{11111(28)\}$. *For the set of faults consisting of primary inputs stuck at* 1 *with probabilities* 0.05, 0.1, 0.15, 0.25, *from top to bottom (see Figure 5), the multitest is* $\{00111(14), 01011(9),\ 01110(7), 01101(28)\}$.

The above method yields a compact multitest because at each step we select the vector that probabilistically covers the largest percent of uncovered faults. Since we never remove vectors under consideration, repetition occurs automatically. Adding a test vector (or an extra repetition of an existing vector) never causes the probability of detection to decrease. In our experiments, we used test vectors generated by the ATALANTA program [2]. Table 3 shows the total number of test vectors required by various circuits for stuck-at-0 faults at input signals with probability 0.05. The repetition algorithm shows significant improvement over random test vector selection from a complete set of test vectors. The desired probabilities of detection are shown in the table. We can further decrease the runtime by identifying test vectors t which are essential for certain faults, i.e., such that $p_{det}(f_i, v) = 0$ for some fault f_i, for all $v \neq t$. Such test vectors t will need to be repeated

$$log_{p_{det}(f_i,t)}(1-p_{th}(f_i))$$

times. Similarly, an upper bound on the number of repetitions of any test vector can be derived in terms of the smallest value of $p_{det}(f_i, v)$.

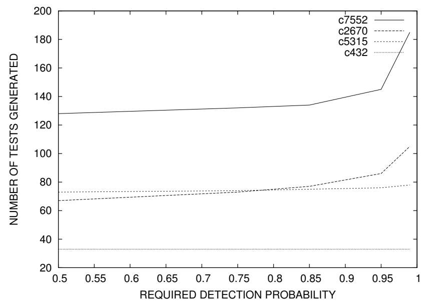


Figure 6: The number of distinct test vectors generated by our techniques as a function of the required detection probability p_{th}.

Table 3 also shows that the number of vectors required increases rapidly from a threshold of .95 to .99. The general trend in the number of vectors required mirrors the growth in case of a fault with one test vector detecting it. Figure 6 shows how many distinct vectors are generated by our techniques for various required detection thresholds p_{th}. The increased variety of test vectors with growing p_{th} indicates that the our technique is not equivalent to simple repetition of a compact set of test vectors.

Alternatively, if we have a limited budget of n test vectors we might like to choose the vectors such that the target faults are detected with as high a probability as possible.

		$p_{th} = .05$			$p_{th} = .75$			$p_{th} = .85$			$p_{th} = .95$			$p_{th} = .99$		
Circuit	# faults	time	alg	rand	time	alg	rand	time	alg	rand	time	alg	rand	time	alg	rand
c6288	32	0.01	56	377	0.01	112	782	0.01	148	1034	0.01	236	1266	0.02	360	1998
c1908	33	0.04	974	T/O	0.04	1337	T/O	0.04	1585	T/O	0.10	2161	T/O	0.12	2991	T/O
c432	36	0.03	462	731	0.67	924	1415	0.90	1221	1771	1.43	1947	2696	2.17	2970	3797
c499	41	0.05	518	1643	0.05	1036	2723	0.06	1369	3085	0.10	2183	4448	0.13	3330	8157
c3540	50	0.05	411	907	0.06	817	1665	0.06	1078	2256	0.14	1716	3589	0.17	2615	4975
c5315	178	0.64	854	2669	14.90	1708	4691	19.96	2557	6531	32.20	3599	8961	50.39	5490	13359
c7552	207	1.68	1680	3729	51.33	3364	6824	68.32	4445	8352	110.60	7082	12210	170.75	10805	18314
c2670	233	0.97	884	3650	22.95	1770	5699	30.21	2339	7755	49.45	3729	11104	76.36	5682	15961

Table 3: Number of test vectors (with repetition) required to detect input signal faults with various threshold probabilities with our method denoted *alg*. Columns marked *rand* give the average number of vectors needed with randomly chosen test vectors. The maximum runtime was one hour.

A naive approach would be to initially evaluate $r(t)$ as before and simply repeat the vector with the lowest $r(t)$ value n times. However, this approach would only detect certain faults and not others. The following method gives roughly equal chances of detecting any of the faults regardless of their occurrence probabilities.

1. Execute the test vector repetition algorithm with $p_{th} = .5$ to obtain the multitest S. Set the variable $inc = .25$.
2. If $|S| > 1.05n$, then $p_{th} = p_{th} - inc$.
3. If $S < n$, then $p_{th} = p_{th} + inc$.
4. Set $inc = inc * .5$.
5. If $n < S < 1.05n$, then remove 5% of the vectors and return S and p_{th}
6. Rerun the test vector repetition algorithm.
7. Return to step 2.

This algorithm uses binary search to detect the desired threshold level that can be met for all the faults without exceeding a test budget.

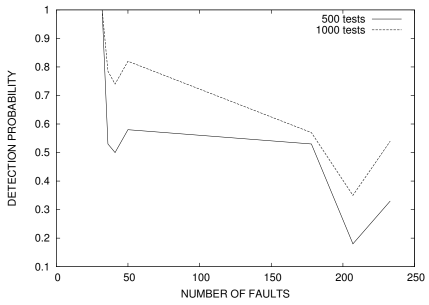


Figure 7: Detection probabilities achievable using 500 and 1000 test vectors. Dips in the graph correspond to faults with low detectability.

Figure 7 shows the optimal threshold of detection for budgets of 500 and 1000 test vectors. As in the case of the circuit with 200 faults, we observe that certain faults with extremely low detectabilities dominate the number of vectors required. Therefore, the elimination of extremely rare faults can considerably increase the optimal detection threshold.

Consider an n-vector subset $T = \{t_1, t_2 \dots t_n\}$ of a multitest S. The probability of vectors in T detecting errors can be estimated using the individual detection probabilities of the test vectors. This can be done by calculating $p_{det}(F, t_i)$ where F consists of all faults under consideration, as in Section 3. Then the probability of n detections will be

$$P_{ndet} = \Pi_{t_i \in T}[p_{det}(F, t_i)] * \Pi_{t_j \in S-T}[1 - p_{det}(F, t_j)]$$

This measure can tell whether the circuit is experiencing a higher error rate than originally expected.

5 Conclusion

Probabilities of transient faults in logic gates can be estimated using gate and environmental parameters. Based on such estimates, we designed test sets to compare the error rate of a combinational circuit to a given threshold. The set of test vectors starts from a complete set of deterministic test vectors intended to detect single stuck-at faults, and is extended to a multitest by a greedy algorithm that repeats some of the test vectors.

We modeled simultaneous multiple faults using a matrix formalism known as probabilistic transfer matrices (PTM). As our results show, the probability of detecting multiple faults is not the same as the probability of the faults occurring, due to logic masking. However, this effect is automatically considered in the process of computing output vectors with PTMs. Our results show that the PTM test vector evaluation method scales well to realistic circuits.

Ongoing work seeks to model correlated faults, such as bridging faults, using the PTM technique. We are also interested in potential applications of transient-fault detection to online testing.

References

[1] L. Geppert "Chip Making's Wet New World," *IEEE Spectrum* May, 2004.

[2] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," Technical Report No. 12-93, Dept. of Electrical Eng., Virginia Polytechnic Institute and State University.

[3] V. L. Levin,"Probability Analysis of Combination Systems and their Reliability,"*Engin. Cybernetics*, no 6. Nov-Dec. 1964, pp. 78-84.

[4] S. Krishnaswamy, G. F. Viamontes, I. L. Markov and J. P. Hayes, "Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices", *Proc. DATE*, March 2005, pp. 282-287.

[5] C. Metra, S. D. Francescantonio, G. Maralle. "On-Line Testing of Transient Faults Affecting Functional Blocks of FCMOS, Domino and FPGA-Implemented Self-Checking Circuits" *DFT* , 2002, pp. 205-217.

[6] K. Mohanram and N. A. Touba, "Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits," *Proc. ITC*, 2003, pp. 893-901.

[7] K.N. Patel, J.P. Hayes, and I.L. Markov, "Evaluating Circuit Reliability Under Probabilistic Gate-Level Fault Models," *Proc. IWLS,* May 2003, pp. 59-64.

[8] S. Reddy, I. Pomeranz, S. Kajihara "Compact Test Sets for High Defect Coverage" *IEEE Trans. on CAD of ICs and Systems* vol.16 no. 8, 1997, pp. 923-930.

[9] P. Shivakumar, et al., "Modeling the Effect of Technology Trends on Soft Error Rate of Combinational Logic" *Proc. DSN*, 2002, pp. 389-398.

[10] H.-J. Wunderlich and G. Kiefer "Bit-Flipping BIST," *Proc. ICCAD*, 1996, 337-343.