A Testability Metric for Path Delay Faults and Its Application

Huan-Chih Tsai Kwang-Ting Cheng
Dept. of ECE
University of California
Santa Barbara, CA 93106

Vishwani D. Agrawal
Bell Laboratories
Lucent Technologies
Murray Hill, NJ 07974

Abstract— In this paper, we propose a new testability metric for path delay faults. The metric is computed efficiently using a non-enumerative algorithm. It has been validated through extensive experiments and the results indicate a strong correlation between the proposed metric and the path delay fault testability of the circuit. We further apply this metric to derive a path delay fault test application scheme for scan-based BIST. The selection of the test scheme is guided by the proposed metric. The experimental results illustrate that the derived test application scheme can achieve a higher path delay fault coverage in scan-based BIST. Because of the effectiveness and efficient computation of this metric, it can be used to derive other design-for-testability techniques for path delay faults.

I. Introduction

A testability metric should reflect the testability attribute of the circuit and should be easy to derive. A reliable testability metric not only gives the early warning of the testability problems in the circuit but also provides guidance for Design for Testability (DfT). Reliable and efficient testability metrics [1][2], along with mature ATPG and fault simulation techniques, allow a fast evolution of DfT for stuck-at faults [3][4][5][6][7]. However, similar methodologies for path delay fault have not yet been established. The extremely large fault population (possibly exponential in number of gates [8]) is one of several major obstacles of developing effective DfT techniques for path delay faults. Although efficient non-enumerative path delay fault simulators exist [9][10], extending the fault-simulation-based DfT techniques from stuck-at faults to path delay faults is still an intractable problem because of the huge cost of repetitive fault simulation runs for a large number of path delay faults. Even if the fault population can be reduced by selecting only a subset of critical paths for testing, the number of paths selected can still be very large if the circuit is highly optimized for timing. As a result, using testability metrics to develop DfT techniques for path delay faults is a more viable approach if a reliable and computationally-efficient testability metric can be developed.

In this paper, we propose a new testability metric for path delay faults. We develop a non-enumerative algorithm to derive this metric. The computational complexity is linear to the number of gates in the circuit. We have conducted experiments to validate the reliability of this metric and the results show a strongly correlation between the proposed metric and the path delay fault testability of the circuit. In Section III, we present details of the metric and illustrate how to derive it in linear time. The evaluation of this testability metric will also be shown in Section III. In Section IV, we show one possible application of using this testability metric to develop a good DfT technique — a test application scheme for scan-based BIST to maximize the path delay fault coverage.

II. Background

A. Path counting algorithm

The fundamental algorithm behind all non-enumerative approaches for path delay fault simulation [9][10] and test generation [11] is a linear time path counting algorithm [12]. First, we define a *complete path* (in short *path*) of a circuit as a path from a primary input or a flip-flop to a primary output or another flip-flop; a *partial path* is defined as a path from a primary input or a flip-flop to an internal signal or from an internal signal to a primary output or a flip-flop. Unless further specified, the term *path* means the *complete path* throughout the rest of this paper. The detail of the path counting algorithm is given in Algorithm 1. In this algorithm, each line s (fanout stem and fanout branches are treated as different lines) in the circuit is associated with a number, $N(s)$. The value of N is initially set to 1 for each primary output, N's for the rest of signals in the circuit are assigned according to Steps 2(a) and 2(b) in Algorithm 1.

Algorithm 1 Path counting

1. $N(s) \leftarrow 1$, for every primary output s
2. Traverse each line s in the circuit in a breath-first fashion from the primary outputs to the primary inputs:
   (a) If s is a fanout stem which has fanout branches $f_1, f_2, \cdots, f_k$, then $N(s) \leftarrow \sum_{j=1}^{k} N(f_j)$
   (b) If s is a gate input and o is the output of the same gate, then $N(s) \leftarrow N(o)$
3. The total number of paths in the circuit is $\sum N(p)$ for every primary input p

The meaning of $N(s)$ for each line s derived in Algorithm 1 is the total number of *partial paths* starting from s to any of the primary outputs. In particular, if s is a primary input, then $N(s)$ is the number of paths starting from s. Therefore, the total number of paths in the circuit is the sum of $N(s)$'s for all primary inputs.

Fig. 1 shows an example of using Algorithm 1 to count the total number of paths in the circuit. First, the N values of m and n is set to 1. The N values of the gate output is directly copied to the gate inputs (like k, l, etc.). For fanout stem j, $N(j)$ is the sum of $N(k)$ and $N(l)$. The same rule applies to c. Finally, the total number of paths in this circuit is $N(a)+N(b)+N(c)+N(d)+N(e)$ which is 10.

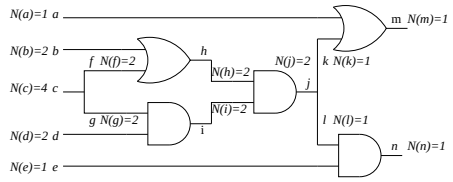


Fig. 1. Finding the total number of paths

B. A testability metric for stuck-at faults

Under the stuck-at fault model, a metric which reflects the random pattern testability of the circuit has been proposed by Lisanke *et al.* [2] as follows.

$$U = \frac{1}{|F|} \sum_{i \in F} \frac{1}{Pd_i} \tag{1}$$

where F is the fault set, $|F|$ is the total number of faults in set F, and Pd_i is the detection probability of fault i. The inverse of Pd_i, $1/Pd_i$, gives the expected test length required for detecting fault i. From this viewpoint, this metric gives the average expected test length required for detecting a single fault in F. The detection probability, Pd_i, of a stuck-at fault i can be approximated using Equation (2) or (3).

$$Pd_{s/0} = C_s \cdot O_s \ \text{, for s-a-0 at signal } s \tag{2}$$

$$Pd_{s/1} = (1 - C_s) \cdot O_s \ \text{, for s-a-1 at signal } s \tag{3}$$

where C_s and O_s are 1-controllability and observability of signal s. C_s and O_s can be derived efficiently using COP [1]. With COP and Equations (2) and (3), U can be computed in linear time. Because of the efficient computation, this metric has been successfully used in many DfT techniques targeting on increasing the stuck-at fault coverage [3][7].

III. A TESTABILITY METRIC FOR PATH DELAY FAULTS

We extend the metric U in Equation (1) to path delay faults. The fault set F in Equation (1) now contains a set of path delay faults and Pd_i becomes the detection probability of the path delay fault i. In the following, we first show how to find detection probability of a path delay fault. After that, we present a non-enumerative algorithm to compute U for path delay faults.

A. Detection probability of a path delay fault

Detecting a path delay fault requires a two-pattern test — the first pattern initializes the circuit while the second pattern causes and propagates the desired transition. To describe the temporal correlation between two consecutive patterns, switching probability, P_s^{xy}, at each primary input s is needed, where x and y can be 0 or 1. Switching probability can also be derived using signal probability (P_s^1) and transition probability (P_s^t). Table I summarizes the definitions of these probability values and their corresponding symbols used in this paper.

The relationships between signal, transition, and switching probabilities are follows [13].

TABLE I
SIGNAL, TRANSITION, AND SWITCHING PROBABILITIES

Symbol	Name	Definition
P_s^1	signal prob.	prob. of s being 1
P_s^t	transition prob.	prob. of s having transitions
P_s^{00}	switching prob.	prob. of s having stable 0
P_s^{01}	switching prob.	prob. of s having $0 \rightarrow 1$ transition
P_s^{10}	switching prob.	prob. of s having $1 \rightarrow 0$ transition
P_s^{11}	switching prob.	prob. of s having stable 1

$$P_s^1 = P_s^{11} + P_s^{01} = P_s^{11} + P_s^{10} \tag{4}$$

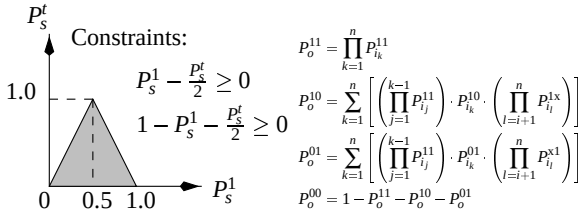
$$P_s^t = P_s^{10} + P_s^{01} \tag{5}$$

$$P_s^{01} = P_s^{10} = P_s^t/2 \tag{6}$$

$$P_s^{11} = P_s^1 - P_s^t/2 \tag{7}$$

$$P_s^{00} = 1 - P_s^1 - P_s^t/2 \tag{8}$$

As noted in [13], there are constraints on the possible values of P_s^1 and P_s^t as shown in Fig. 2(a). The values of P_s^1 and P_s^t are meaningful only in the shaded area. Given P_s^1 and P_s^t of every primary input s, the switching probability of every signal can be computed using a COP-like method. Fig. 2(b) shows the formulae for computing the switching probabilities at the output of an n-input AND gate given the switching probabilities at the inputs. Similar equations can be derived for other gate types.



(a) Constraints of P_s^1 and P_s^t (b) Computing switching prob. for an AND gate

Fig. 2. Constraints of P_s^1 and P_s^t, and equations of computing switching prob.

Once the switching probabilities of every signal has been calculated, the detection probability of a path delay fault can be estimated using the product of (1) the probability of creating the desired transition at the input (activation), (2) the probability of sensitizing this transition to the output (sensitization) and (3) the observability of the output (observation). The sensitization criteria for different types of path delay faults (e.g. robust, non-robust, etc.) can be found in [8]. Fig. 3 shows an example. Here, we are interested in finding the detection probability of the path delay fault of path $A \rightarrow E \rightarrow F \rightarrow G$ with a rising transition at A using the robust sensitization criterion. The probability of creating the desired transition is P_A^{01}; the probabilities of sensitizing the transition robustly through $G1$, $G2$, and $G3$ are $(P_B^{01} + P_B^{11})$, P_C^{00}, and $(P_D^{01} + P_D^{11})$, respectively. Therefore, the detection probability is $P_A^{01} \cdot (P_B^{01} + P_B^{11}) \cdot P_C^{00} \cdot (P_D^{01} + P_D^{11}) \cdot obs(G)$, assuming $obs(G)$ is the COP observability of G.

B. Computing the testability metric

The huge number of paths makes it impractical to compute the detection probability of every path delay fault explicitly.

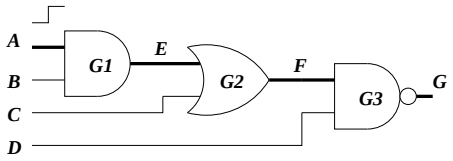


Fig. 3. Finding the detection prob. of a path delay fault

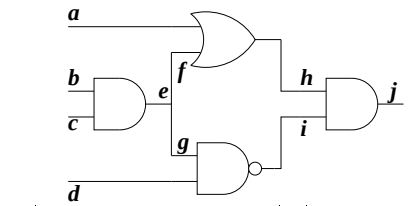
We propose a non-enumerative method to compute U by extending Algorithm 1. We first assign each line s in the circuit two values: $U_r(s)$ for rising transition and $U_f(s)$ for falling transition. It will soon become clear that $U_r(s)$ ($U_f(s)$) of an internal signal s is ${P_s}^{01} \cdot \sum_i \frac{1}{PPd_i}$ (${P_s}^{10} \cdot \sum_i \frac{1}{PPd_i}$), where i is a path delay fault for a *partial path* from s to any primary output (or flip-flop) when s has a rising (falling) transition, and PPd_i is the detection probability of i. Therefore, $U_r(s)/{P_s}^{01}$ ($U_f(s)/{P_s}^{10}$) is the sum of the inverse of detection probabilities for all faults along the partial paths from s to any primary output (or flip-flop) when s has a rising (falling) transition. If s is a primary input, the path from s to a primary output (or flip-flop) becomes a *complete* path instead of a *partial* path. Thus, the sum of $U_r(s)/{P_s}^{01}$ and $U_f(s)/{P_s}^{10}$ for every primary input s is the U value we want to derive. The $U_r(s)$ and $U_f(s)$ for every line in the circuit can be computed using Algorithm 2. In

Algorithm 2 Computing testability metric

1. $U_r(s) \leftarrow \frac{1}{obs(s)}$ and $U_f(s) \leftarrow \frac{1}{obs(s)}$, for every primary output s
2. Traverse each line s in the circuit in a breath-first fashion from the primary outputs to the primary inputs:
   (a) If s is a fanout stem which has fanout branches f_1, f_2, $\cdots$, f_k, then $U_r(s) \leftarrow \sum_{j=1}^{k} U_r(f_j)$ and $U_f(s) \leftarrow \sum_{j=1}^{k} U_f(f_j)$
   (b) If s is a gate input and o is the output of the same gate, then $U_r(s) \leftarrow \frac{U_r(o)}{S_s}$ and $U_f(s) \leftarrow \frac{U_f(o)}{S_s}$ for non-inverting gates ($U_r(s) \leftarrow \frac{U_f(o)}{S_s}$ and $U_f(s) \leftarrow \frac{U_r(o)}{S_s}$ for inverting gates), where S_s is the prob. of sensitizing the desired transition from s to o.
3. $U \leftarrow \frac{1}{|F|} \sum (\frac{U_r(s)}{{P_s}^{01}} + \frac{U_f(s)}{{P_s}^{10}})$ for every primary input p

the following, we explain how $U_r(s)$ is computed. $U_f(s)$ can be derived similarly. First, we initialize $U_r(s)$ of every primary output s to the inverse of its observability (i.e. $1/obs(s)$). Then we compute $U_r(s)$ of each line s in the circuit in a breadth-first fashion from the primary outputs toward the primary inputs. If s is a fanout stem, then $U_r(s)$ is the sum of all U_r values at the fanout branches. If s is a gate input and o is the output of the same gate, then we first compute the probability, denoted as S_s, of sensitizing the desired transition from s to o. After that, $U_r(s)$ is calculated by dividing $U_r(o)$ by S_s (i.e. $U_r(o)/S_s$) for the gate with a non-inverting output. If the gate has an inverting output (i.e. an NAND, NOR, or inverter), then $U_r(s)$ is $U_f(o)/S_s$. Once U_r and U_f of every line have been computed, U can be found by adding $U_r(s)/P_s^{01}$ and $U_f(s)/P_s^{10}$ for each primary input s and then dividing it by the total number of faults, $|F|$.

Fig.4 shows an example of using Algorithm 2 to find U. In this example, we assume the robust sensitization criterion is used. Initially we set $U_r(j)$ and $U_f(j)$ to $1/obs(j)$ (Step 1). To sensitize a rising transition from h to j, the second pattern at i needs to be 1 (i.e. the value of a pattern pair could be



	U_r	U_f		U_r	U_f
a	$\frac{U_r(h)}{P_f^{00}}$	$\frac{U_f(h)}{(P_f^{10}+P_f^{00})}$	b	$\frac{U_r(e)}{(P_c^{01}+P_c^{11})}$	$\frac{U_f(e)}{P_c^{11}}$
c	$\frac{U_r(e)}{(P_b^{01}+P_b^{11})}$	$\frac{U_f(e)}{P_b^{11}}$	d	$\frac{U_r(i)}{(P_g^{01}+P_g^{11})}$	$\frac{U_r(i)}{P_g^{11}}$
e	$U_r(f)+U_r(g)$	$U_f(f)+U_f(g)$	f	$\frac{U_r(h)}{P_a^{00}}$	$\frac{U_f(h)}{P_a^{10}+P_a^{00}}$
g	$\frac{U_f(i)}{P_d^{01}+P_d^{11}}$	$\frac{U_r(i)}{P_d^{11}}$	h	$\frac{U_r(j)}{P_i^{01}+P_i^{11}}$	$\frac{U_f(j)}{P_i^{11}}$
i	$\frac{U_r(j)}{P_h^{01}+P_h^{11}}$	$\frac{U_f(j)}{P_h^{11}}$	j	$\frac{1}{obs(j)}$	$\frac{1}{obs(j)}$

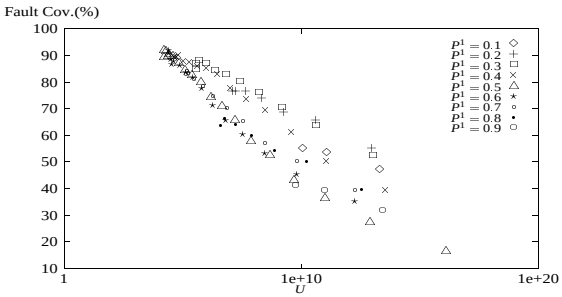
Fig. 4. Computing U for path delay faults

either 01 or 11). Therefore, $U_r(h)$ is equal to $\frac{U_r(j)}{P_i^{01}+P_i^{11}}$ (Step 2(b)). On the other hand, to sensitize a falling transition from h to j requires i to be stable 1, thus $U_f(h)$ is equal to $\frac{U_f(j)}{P_i^{11}}$ (Step 2(b)). The U_r and U_f values of all other signals can be derived similarly. For fanout stem e, its U_r and U_f are the sum of the corresponding values at the fanout branches (Step 2(a)), thus, $U_r(e) = U_r(f) + U_r(g)$ and $U_f(e) = U_f(f) + U_f(g)$. The U_r and U_f values of all other signals in this circuit can also be found in Fig. 4. It can be verified that the value of $\frac{U_f(b)}{P_b^{10}}$ is $\frac{1}{P_b^{10} \cdot P_c^{11} \cdot P_d^{11} \cdot (P_h^{01}+P_h^{11}) \cdot obs(j)} + \frac{1}{P_b^{10} \cdot P_c^{11} \cdot (P_a^{10}+P_a^{00}) \cdot P_i^{11} \cdot obs(j)}$. The denominators are indeed the detection probabilities of the path delay faults with a rising transition at b along the paths $b \rightarrow e \rightarrow g \rightarrow i \rightarrow j$ and $b \rightarrow e \rightarrow f \rightarrow h \rightarrow j$, respectively. Note that the complexity of Algorithm 2 is linear in terms of the number of gates in the circuit.

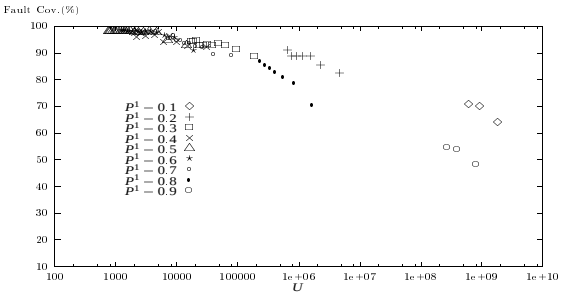
C. Metric validation

We have performed experiments on many benchmark circuits to show that the proposed testability metric is strongly correlated to the path delay fault testability of the circuit. For each circuit, we assigned the input signal probability P^1 from 0.1 to 0.9 with a step of 0.1. For each P^1, we further varied the input transition probability P^t from 0.05 to $(2P^1 - 0.05)$ with a step of 0.05 (note that with a given P^1, the valid range of P^t is only in the shaded area of Fig. 2(a)). Therefore, a total of 91 different combinations of the primary input signal and transition probability profiles were generated. Under one profile, we use the same P^1 and P^t for all inputs. For each profile, we computed the metric, U, and generated 100K random vectors which match the corresponding profile for running the path delay fault simulation. The path delay fault simulation is performed on combinational portion of the circuit using an industrial tool, **spdf**, from Lucent Technologies [10]. Note that the proposed metric can also be used for the case of having different P^1's and P^t's for different inputs. For each fault simulation run, we recorded the path delay fault coverages for all paths and for the 25% longest paths. We also recorded the line delay fault coverage [14].

The distribution of the actual path delay fault coverage (all paths) *vs.* U for circuit **s832** is shown in Fig. 5. Similar distributions were found for all other cases. In the figure, the hori-



(a) s832 (Robust)



(b) s832 (Non-robust)

Fig. 5. Distribution of fault coverage *vs. U*

zontal axis is the U value and the vertical axis is the fault coverage after simulating 100K vectors for all paths. A point in the figure corresponds to the result of one profile (out of totaled 91 profiles). Different symbols in the figure indicate the profiles of different P^1's. The distribution shows a strong correlation of a decreasing fault coverage with an increasing U value. This result, in fact, matches the implication of U — a longer average expected test length (larger U) implies that the circuit is harder to test. This trend is even clear for points with the same P^1 but different P^t (i.e. points of the same symbol). The correlation between the metric and the fault coverage can be further illustrated by calculating the correlation coefficient (ρ). Table II shows the absolute values of ρ's between the different types of fault coverage and the logarithm of U (Columns "All" for all paths, "Long" for the 25% longest paths, and "Line" for the line delay faults). The coefficients are negative for all cases. This means that the larger the U is, the smaller the fault coverage is. For most of the cases, $|\rho|$ is close to 1. This indicates that the proposed metric U and the path (line) delay fault are *strongly* correlated. Furthermore, the robust path (line) delay fault coverage and U are even more strongly correlated (points of the same P^1). As shown in Table III, $|\rho|$ is typically greater than 0.98.

TABLE II
CORRELATION COEFFICIENT, $|\rho|$, BETWEEN THE ACTUAL FAULT COVERAGE AND U

Circuit	Robust			Non-robust		
	All	Long	Line	All	Long	Line
s832	0.93	0.92	0.93	0.93	0.91	0.93
s1494	0.94	0.92	0.95	0.88	0.91	0.89
s3330	0.75	0.85	0.88	0.70	0.92	0.91
s5378	0.94	0.93	0.94	0.95	0.95	0.96

TABLE III
CORRELATION COEFFICIENT, $|\rho|$, BETWEEN THE ACTUAL FAULT COVERAGE AND U (UNDER THE SAME P^1 AND ROBUST SENSITIZATION CRITERION)

P^1		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
s832	All	0.99	0.99	0.99	0.99	0.98	0.98	0.99	0.99	0.99
	Long	0.99	0.99	0.99	0.99	0.97	0.98	0.99	0.98	0.91
	Line	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
s1494	All	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	Long	0.98	0.99	0.99	0.99	0.96	0.99	0.98	0.99	0.99
	Line	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98
s3330	All	0.94	0.99	0.99	0.98	0.98	0.99	0.99	0.98	0.99
	Long	0.99	0.96	0.98	0.98	0.93	0.98	0.98	0.99	0.99
	Line	0.96	0.98	0.98	0.98	0.97	0.97	0.99	0.99	0.99
s5378	All	0.99	0.98	0.99	0.99	0.98	0.99	0.99	0.99	0.94
	Long	0.99	0.99	0.98	0.96	0.90	0.97	0.99	0.97	0.91
	Line	0.99	0.99	0.99	0.98	0.92	0.98	0.99	0.98	0.98

IV. APPLICATION — A PATH DELAY FAULT TEST APPLICATION SCHEME FOR SCAN-BASED BIST

According to how the test is applied, the scan-based BIST architectures can be classified as either *test-per-clock* BIST or *test-per-scan* BIST [15]. In *test-per-clock* BIST, a test vector is applied and its response is compressed *every clock cycle*. The examples of test-per-clock BIST are BILBO-based design [16] and circular BIST [17]. In *test-per-scan* BIST, a test vector is applied and its response is captured into the scan chains *only after the test is scanned into the scan chains*. That is, we first set the circuit to the scan mode for l clock cycles to scan in the vector, where l is the length of the longest scan chain. After that, we change the circuit to the normal functional mode for *one* clock cycle to capture the circuit responses. Finally, we shift out the captured responses, and meanwhile we shift in a new vector. STUMPS [18] is an example of test-per-scan BIST. The test-per-clock BIST typically has shorter test time but incurs higher area and performance overhead than test-per-scan BIST. Here we focus on deriving a test application scheme which is an enhancement of the conventional test-per-scan scheme but can achieve a better path delay fault coverage.

A. Path delay faults testing for scan-based BIST

In the conventional test-per-scan BIST, path delay faults can be tested using the clocking scheme shown in Fig. 6. Here

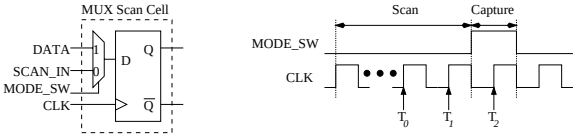


Fig. 6. The conventional clocking scheme for detecting path delay faults

we use a MUX scan cell as an example. The scan operation depends on the value of the mode switch (MODE_SW input of the MUX scan cell). The scan cell is performing the scan operation if MODE_SW is 1, otherwise, it is performing the capture operation. To test path delay faults, the first and second vectors are applied at time T_0 and T_1, respectively. The circuit response is then captured into the scan cell at T_2.

It has been shown that applying different numbers (k) of capture cycles after each scan sequence (k captures per scan) can

produce patterns with different signal probability profiles at the scan flip-flops [19]. Patterns which matches a specific signal probability profile can improve the detection for some stuck-at faults but deteriorate the detection for others comparing to the pseudo random patterns. In order to increase the overall circuit random testability, in [19] the entire test process is divided into multiple test sessions. In each test session, patterns with a specific signal probability profile are generated by applying a unique number of capture cycles after each scan sequence. This is similar to the multiple weight set weighted random testing. However, instead of using a special weighting logic, the circuit under test itself is used as the weighting logic to generate the weighted random patterns at the scan flip-flops and thus does not incur additional hardware. In summary, unlike the conventional test-per-scan BIST which always performs 1 capture per scan, they proposed a general test scheme which tests the circuit using multiple test sessions with k captures per scan in each test session, where k can be other than 1 and is different in different test sessions.

This generalized test application scheme can also help to catch delay faults. Because the circuit is running at-speed and exercising the functional paths during the capture cycles, the path delay faults are more likely to be activated. Meanwhile, applying multiple capture cycles increasing the chance of latching fault effects into the scan flip-flops. This is desirable especially when we also observe the responses at the primary outputs every clock cycle. In other words, the circuit response captured in a scan flip-flop can be observed not only at the output of the scan chain but also at the primary outputs (via functional logic). The clocking scheme for path delay faults testing by applying multiple capture cycles after each scan sequence is shown in Fig. 7. The initial vector pair is applied at time T_0 and T_1. After that, we capture the circuit responses k times (from T_2 to T_{k+1}). The captured responses at T_2 to T_k are also used as the test vectors for cycles of T_3 to T_{k+1}. The response captured at time T_{k+1} will eventually be scanned out and observed.

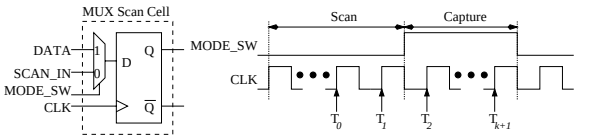


Fig. 7. A general clocking scheme for detecting path delay faults

B. Deriving test application scheme

As noted in [19], applying multiple capture cycles can improve the detection for only a subset of faults. Thus, they found the best number of capture cycles for each fault explicitly by computing the fault detection probability and then identify the number of test sessions and the corresponding number of capture cycles used in each test session. However, finding the detection probability explicitly for each path delay fault is impractical. To overcome this problem, we divide all faults into several *fault groups*, and then compute U for each group separately. The U value for a group only reflects the testabilities of faults in that group. In this application, we assign all paths between each I/O pair as one group. The U values of all groups can be computed by applying Algorithm 2 m times with a modified Step 1, where m is the number of outputs. Each time, we only assign $1/obs$ to the U_r and U_f for *one* primary output s. For all other primary outputs, their U_r and U_f values are assigned 0. At the end of this run, we obtain the U values for *all groups between any primary input and the primary output s.* The overall complexity of computing U values for all groups is $O(mN)$, where N is gate count. For k captures per scan, we use a procedure similar to that in [19] to compute the switching probabilities. The procedure can be illustrated in Fig. 8, where PSI is the output of a scan flip-flop, and PSO is the input of a scan flip-flop. We first perform time frame expansion on the circuit (one time frame for each capture cycle). We then assign the appropriate switching probabilities at the primary inputs (PIs) and PSIs of the circuit in each time frame so that we can derive the switching probabilities of every signal in the circuit in different time frame using the formulae shown in Fig. 2(b). Observabilities can be computed similarly by setting appropriate observabilities at the primary outputs (POs) and PSOs and then backward propagated toward the inputs. After computing the switching probabilities and observabilities, the U value of every fault group is calculated for each time frame. We assume no path delay faults can be detected during the *scan cycle*, therefore, to derive the test application scheme, we only consider U values in those k time frames which correspond to k capture cycles. We use the average of these k U's for each fault group i, $U_i^{ave,k}$, to determine the best number (k) of capture cycles for this fault group. The best k for fault group i (or all faults in group i) is the one resulting in the smallest $U_i^{ave,k}$.

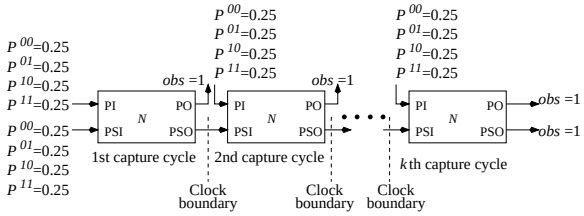


Fig. 8. Computing switching probabilities

Fig. 9 shows the flow of procedure to determine the test application scheme. The procedure first find all fault groups and the total number of path delay faults (F). The number of capture cycles, k, is initially set to 0. At each iteration, k is first incremented and then the $U_i^{ave,k}$ for every fault group is computed and is used to determine the best k so far for each fault group. We then record the number of *faults* which change their best k values, i.e. n in Fig. 9. If the n is larger than C_{th}% of the total number of faults, F, the iteration continues. Otherwise, the iteration stops, and we select a set of k's which cover F_{th}% of F.

C. Experimental Results

We conducted experiments to demonstrate the effectiveness of using the metric U to guide the selection of the test application scheme. In the experiments, the P^1 and P^t for all inputs (primary inputs and inputs of scan chains) were set to

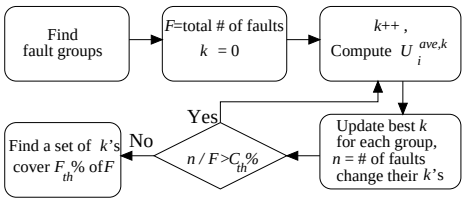


Fig. 9. Flow of test scheme selection

0.5. We first identified the test scheme and then fault simulated the scanned circuits (using both single test session with one capture cycle per scan and multiple test sessions with multiple capture cycles per scan) for one million clock cycles. The numbers under Column "*k*" show the test application scheme determined using the flow in Fig. 9. For instance, for circuit **s953**, the proposed procedure found that we should use 3 test sessions to test the circuit and in each test session, we should use 1, 2, or 3 capture cycles after each scan sequence, respectively. The operation of capturing the responses multiple times after each scan sequence can be achieved with a minor modification of the BIST controller. Note that using the multiple test sessions scheme incurs a slightly bigger BIST controller. However, the size of the BIST controller does not depend on the circuit size and the number of flip-flops in the circuit. Furthermore, the BIST controller typically occupies a small fraction of the total area. Therefore, using the multiple test sessions scheme practically incurs no extra overhead for large designs. Table IV shows the fault simulation results. The path delay fault coverages for all paths (Columns "All") and the 25% longest paths (Columns "Long"), and the line delay fault coverage (Columns "Line") are presented. The fault coverages are lower than the pure random cases as shown in Fig 5. This is because only a limited number of two-pattern tests can be applied in the scan environment. Moreover, the observability of the scan flip-flop is also limited. The last column ("CPU") is the CPU time (measured on Sparc 20) used to determine the multiple test sessions test application scheme. Table IV illustrates that using the multiple test sessions scheme can achieve higher path (line) delay fault coverages. These results clearly show that the use of the new metric successfully guides the selection of test application scheme to increase the path (line) delay fault coverage.

TABLE IV
RESULTS OF MULTIPLE TEST SESSION SCHEME (NON-ROBUST)

Circuit	Single			Multiple				
	All (%)	Long (%)	Line (%)	*k*	All (%)	Long (%)	Line (%)	CPU (s.)
s832	58.76	36.11	50.23	1, 4	67.23	51.39	61.89	4.8
s953	46.15	41.01	83.46	1, 2, 3	49.88	44.06	87.91	4.4
s1494	67.24	36.05	64.60	1, 4	73.41	48.07	73.93	9.5
s3271	22.93	14.91	88.10	1, 2	26.00	19.00	88.58	43.1
s3330	65.82	41.78	68.35	1, 4	69.67	50.12	76.80	170.6

V. CONCLUSION

In this paper, we propose a new testability metric for path delay faults. The proposed metric can be computed efficiently using a non-enumerative algorithm. The metric has been validated through extensive experiments. The results show a strong correlation between the metric and the path delay fault testability. We also propose a test application scheme for scan-based BIST to maximize path delay fault coverage. The test scheme selection is guided by the proposed metric. The experimental results indicate that the selected test scheme can achieve higher path delay fault coverages without adding extra hardware to the circuit under test. Other possible applications (such as designing test pattern generators, etc.) of the proposed metric are currently under investigations.

REFERENCES

[1] F. Brglez, "On Testability of Combinational Networks," *Proc. of ISCAS*, pp. 221–225, May 1984.

[2] R. Lisanke, F. Brglez, A.J. Degeus, and D. Gregory, "Testability-Driven Random Test Pattern Generation," *IEEE Trans. on CAD*, vol. CAD-6, no. 6, pp. 1082–1087, Nov. 1987.

[3] B. Seiss, P. Trouborst, and M. Schalz, "Test Point Insertion for Scan-Based BIST," *Proc. of European Test Conf.*, pp. 253–262, Apr. 1991.

[4] N. Tamarapalli and J. Rajski, "Constructive Multi-Phases Test Point Insertion for Scan-Based BIST," *Proc. of ITC*, pp. 649–658, Oct. 1996.

[5] N.A. Touba and E.J. McCluskey, "Altering a Pseudo-Random Bit Sequence for Scan-Based BIST," *Proc. of ITC*, pp. 167–175, Oct. 1996.

[6] K.-H. Tsai, S. Hellebrand, J. Rajski, and M. Marek-Sadowska, "STAR-BIST: Scan Autocorrelated Random Pattern Generation," *Proc. of DAC*, pp. 472–477, June 1997.

[7] H.-C. Tsai, S. Bhawmik, and K.-T. Cheng, "An Almost Full-scan BIST Solution — Higher Fault Coverage and Shorter Test Application Time," *Proc. of ITC*, pp. 1065–1073, Oct. 1998.

[8] A. Kristić and K.-T. Cheng, *Delay Testing for VLSI Circuits*, Kluwer Academic Publishers, Boston, 1998.

[9] M.A. Charaybeh, M.L. Bushnell, and V.D. Agrawal, "An Exact Non-Enumerative Fault Simulator for Path Delay Faults," *Proc. of ITC*, pp. 276–285, Oct. 1996.

[10] C.G. Parodi, V.D. Agrawal, M.L. Bushnell, and S. Wu, "A Non-Enumerative Path Delay Fault Simulator for Sequential Circuits," *Proc. of ITC*, pp. 934–943, Oct. 1998.

[11] I. Pomeranz, S.M. Reddy, and P. Uppaluri, "NEST: A Nonenumerative Test Generation Method for Path Delay Faults in Combinational Circuits," *IEEE Trans. on CAD*, vol. 14, no. 12, pp. 1505–1515, Dec. 1995.

[12] I. Pomeranz, L.N. Reddy, and S.M. Reddy, "SPADES: A Simulator for Path Delay Faults in Sequential Circuits," *Proc. of European Design Automation Conf.*, pp. 428–435, Sept. 1992.

[13] D.I. Cheng, *Power Estimation of Digital CMOS Circuits and the Application to Logic Synthesis for Low Power*, Ph.D. thesis, Dept. of ECE, University of California at Santa Barbara, 1995.

[14] A.K. Majhi, J. Jacob, L.M. Patnaik, and V.D. Agrawal, "On Test Coverage of Path Delay Faults," *Proc. of Int'l Conf. on VLSI Design*, pp. 418–421, Jan. 1996.

[15] V.D. Agrawal, C.R. Kime, and K.K. Saluja, "A Tutorial on Built-In Self-Test, Part 2: Applications," *IEEE Design and Test of Computers*, vol. 10, no. 22, pp. 69–77, June 1993.

[16] B. Konemann, J. Mucha, and C. Zwiehoff, "Built-In Logic Block Observation Technique," *Digest of Papers 1979 Test Conf.*, pp. 37–41, Oct. 1979.

[17] A. Krasniewski and S. Pilarski, "Circular Self-Test Path: A Low-Cost BIST Technique for VLSI Circuits," *IEEE Trans. on CAD*, vol. 8, no. 1, pp. 46–55, Jan. 1989.

[18] P.H. Bardell and W.H. McAnney, "Self-Testing of Multichip Logic Modules," *Digest of Papers 1982 Int'l Test Conf.*, pp. 200–204, Nov. 1982.

[19] H.-C. Tsai, K.-T. Cheng, and S. Bhawmik, "Improving The Test Quality for Scan-based BIST Using A General Test Application Scheme," *Proc. of DAC*, pp. 748–753, June 1999.