Substituting Transition Faults with Path Delay Faults as a Basic Delay Fault Model

Irith Pomeranz
School of Electrical & Computer Eng.
Purdue University
W. Lafayette, IN 47907, U.S.A.

Abstract - **Comparing a single transition fault with a single path delay fault, targeting (i.e., simulating or generating a test for) a path delay fault is not more complex than targeting a transition fault. However, targeting a set of path delay faults is significantly more complex than targeting a set of transition faults when the goal is to consider the testable path delay faults that are associated with the longest paths. The reason is the large fraction of untestable path delay faults among these faults. This complication is removed if the requirement on the lengths of the paths is removed. In this case, it is possible to use path delay faults instead of transition faults as a basic delay fault model for better coverage of small delay defects. This paper studies the effects of using path delay faults as a basic delay fault model instead of transition faults.**

I. INTRODUCTION

Delay defects can be modeled by transition faults [1] or by path delay faults [2]. Transition faults model the case where the extra delay of a line is large such that propagating the effects of the defect to an output through a path of any length will result in a faulty output value. Path delay faults model the case where small extra delays, corresponding to small delay defects, result in a faulty output value when they are accumulated along a path. In addition, detecting transition faults through the longest paths [3]-[5] allows tests for transition faults to detect smaller delay defects. Small delay defects are important to detect since they are prevalent, and they are less likely to be detected accidentally than large delay defects.

Among the longest paths of a circuit there are typically many paths that are untestable [6]-[12]. This complicates test generation procedures that attempt to use the longest paths. These procedures have to consider large numbers of paths, many of which are untestable, in order to find the longest testable paths. This includes the case where path delay faults that are associated with the longest paths are targeted, as well as the case where transition faults are detected through the longest paths. Therefore, transition faults may be used as a basic delay fault model for covering delay defects, and a set of carefully selected path delay faults may be used for additional coverage of small delay defects on critical paths.

Path delay faults are considerably simpler to target without the requirement to consider the longest paths. In this case, the fraction of testable path delay faults is higher, and testable faults are easier to find. Even without considering the longest paths, path delay faults capture the effects of small delay defects better than transition faults. This is due to the use of propagation conditions for small delay defects along a path in tests for path delay faults. Therefore, for a basic test set that targets delay defects, it is advantageous to consider path delay faults instead of transition faults, and allow path delay faults that are associated with shorter paths to be targeted. An interesting result that is demonstrated in this paper is that a test set for such a fault model sometimes yields a higher fault coverage than a test set for transition faults. Thus, the circuit can be covered better with a test set for path delay faults. For this discussion, the set of target faults is defined as follows.

Let $p = g_0 - g_1 - \cdots - g_{L-1}$ denote a path of length L with lines $g_0, g_1, \cdots, g_{L-1}$. For $a_0 = 0$ or 1, let $p: a_0 \rightarrow a'_0$ denote the path delay fault that is associated with p and the $a_0 \rightarrow a'_0$ transition at its source g_0. When the $a_0 \rightarrow a'_0$ transition is propagated through p, let the transition on line g_i be $a_i \rightarrow a'_i$, for $0 \le i < L$. With an even number of inverters on the subpath between g_0 and g_i, $a_i = a_0$. With an odd number of inverters, $a_i = a'_0$. The path delay fault $p: a_0 \rightarrow a'_0$ is said to cover the $a_i \rightarrow a'_i$ transition on g_i, for $0 \le i < L$. This transition is denoted by $g_i: a_i \rightarrow a'_i$. Depending on the propagation conditions used for detecting path delay faults, a test for $p: a_0 \rightarrow a'_0$ may assign the $a_i \rightarrow a'_i$ transition to line g_i, or it may assign a pulse that includes this transition. The transition is said to be covered by the path delay fault since a defect that delays the transition will increase the delay of the path, and such a delay is the target of a test for the path delay fault.

To define a number of target faults that is equal to the number of transition faults, the fault model considered in this paper targets all the transitions in the circuit. A transition is detected when a path delay fault that covers it is detected. This is similar to the fault model considered in [13], but with the important difference that the requirement in [13] is to use one of the longest paths that covers

978-3-9815370-2-4/DATE14/©2014 EDAA

every transition. Here, the path can be of any length. This facilitates significantly the consideration of path delay faults, and allows path delay faults to be used as a basic delay fault model instead of transition faults.

The procedure described in this paper associates with every transition $g:a \rightarrow a'$ a set of path delay faults $P_n(g:a \rightarrow a')$, and requires that at least one of the path delay faults in $P_n(g:a \rightarrow a')$ would be detected. Here, n is a parameter that determines the number of path delay faults that are considered for every transition. The procedure increases n gradually. This allows it to find the point where the fault coverage is sufficiently high using as few path delay faults as possible for every transition. The path delay fault coverage is computed as the percentage of transitions such that at least one path delay fault from their set is detected.

To study the relationships between using transition faults and using path delay faults as targets for test generation, the test generation procedure described in this paper is applied starting from a test set T for transition faults. In this way, the results of test generation will demonstrate the extent to which a transition fault test set already detects path delay faults as defined here, and the extent to which it needs to be augmented in order to increase the path delay fault coverage. The procedure can be applied with $T = \phi$ when the goal is to generate a path delay fault test set without first generating transition fault tests.

The procedure is separated into two parts as described next. The first part performs path delay fault simulation. As n is increased, more path delay faults are considered for every transition, and the path delay fault coverage increases even without generating additional tests. However, experimental results show that the path delay fault coverage saturates for a low value of n.

The second part of the procedure performs test generation for path delay faults. It considers every transition $g:a \rightarrow a'$ such that none of the path delay faults in $P_n(g:a \rightarrow a')$ is detected, and attempts to generate a test for one of these path delay faults.

The paper is organized as follows. Section II describes the derivation of a set of path delay faults $P_n(g:a \rightarrow a')$ for the transition $g:a \rightarrow a'$. Section III describes the procedure for fault simulation and test generation. Section IV presents experimental results.

II. SETS OF PATH DELAY FAULTS

This section describes the construction of a set of path delay faults $P_n(g:a \rightarrow a')$ for a transition $g:a \rightarrow a'$.

For the discussion in this paper, the length of a path is equal to the number of lines along the path. Let the lengths of the paths that go through a line g be $l_0, l_1, \cdots, l_{N-1}$ such that $l_0 < l_1 < \cdots < l_{N-1}$. The set $P_n(g:a \rightarrow a')$ contains all the path delay faults that cover $g:a \rightarrow a'$, and are associated with paths of lengths $l_0, l_1, \cdots$. The path length is increased until $P_n(g:a \rightarrow a')$ contains at least n path delay faults.

In Section IV, the longest paths are considered for comparison. Shortest paths were selected here since there are more testable paths among the shortest paths. This simplifies the test generation process and allows path delay faults to replace transition faults as a basic delay fault model.

To find $P_n(g:a \rightarrow a')$, the procedure first marks every line in the circuit that participates in a path, which also includes g. For this purpose, the procedure marks every line h such that there is a path from g to h. This is done by traversing the circuit forward starting from g. In addition, the procedure marks every line h such that there is a path from h to g. This is done by traversing the circuit backward starting from g. For a line h, $rch(h) = 1$ if there is a path from h to g or from g to h, and $rch(h) = 0$ if there is no such path. The set of lines with paths from or to g is denoted by C. Thus, $C = \{h : rch(h)=1\}$.

Traversing the circuit backward starting from the outputs, and using only lines from C, the procedure computes the length of the shortest path from every line to an output. The length of the shortest path from a line h to an output is denoted by $len(h)$ (when the goal is to find the longest paths, $len(h)$ is the length of the longest path from h to an output).

For illustration, Figure 1 shows ISCAS-89 benchmark $s27$. Line numbers are shown in square brackets. The transition under consideration is 19:0→1. Following the line numbers, Figure 1 shows the length of the shortest path from h to an output for every line $h \in C$.

The procedure develops paths starting from the inputs of the combinational logic. For every input h such that $rch(h) = 1$, the procedure initially includes in $P_n(g:a \rightarrow a')$ a partial path that consists only of h. The procedure extends the paths in $P_n(g:a \rightarrow a')$ iteratively by adding one line to every path in every iteration. At an arbitrary step of this process, let $g_0 - g_1 - \cdots - g_{m-1}$ be a partial path in $P_n(g:a \rightarrow a')$. The procedure extends the path as follows.

If g_{m-1} is an output of the combinational logic, the extension of the path is complete, and the path is not considered further. Otherwise, the path is replaced with the following paths.

If g_{m-1} is a fanout stem with branches $h_0, h_1, \cdots, h_{k-1}$, the procedure defines $H = \{h_0, h_1, \cdots, h_{k-1}\}$. If g_{m-1} is a gate input with output h_0, the procedure defines $H = \{h_0\}$. For every $h_i \in H$, if $rch(h_i) = 1$, the procedure adds the path $g_0 - g_1 - \cdots - g_{m-1} - h_i$ to $P_n(g:a \rightarrow a')$.

At the end of an iteration, the procedure associates with every partial path $p \in P_n(g:a \rightarrow a')$ its minimum expected length. For $p = g_0 - g_1 - \cdots - g_{m-1}$, the minimum expected length is $m + len(g_{m-1})$. This can be seen from

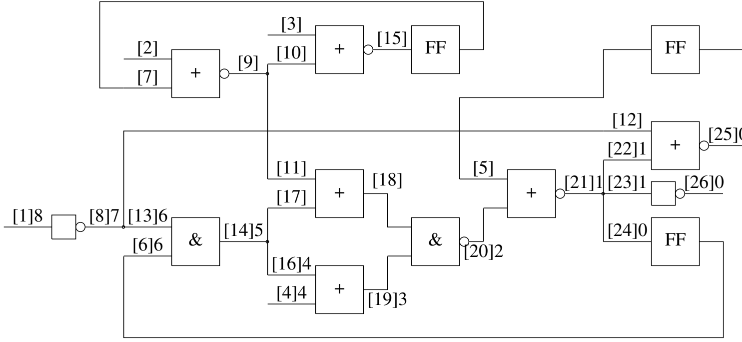

Figure 1. Path Lengths for s27

the fact that the path already includes m lines, and the length of the shortest path from g_{m-1} to an output is $len(g_{m-1})$. Let the expected lengths for the paths in $P_n(g:a \rightarrow a')$ be $l_{e,0}$, $l_{e,1}$, $\cdots$, $l_{e,N-1}$, and suppose that $l_{e,0} < l_{e,1} < \cdots < l_{e,N-1}$. The procedure keeps in $P_n(g:a \rightarrow a')$ the paths of lengths, $l_{e,0}$, $l_{e,1}$, $\cdots$, $l_{e,q-1}$, where q is the smallest integer such that the number of paths in $P_n(g:a \rightarrow a')$ is at least n. It removes all the other paths from $P_n(g:a \rightarrow a')$.

After all the paths in $P_n(g:a \rightarrow a')$ reach an output of the combinational logic, the procedure defines path delay faults by associating a transition with the first line of every path. Let $p = g_0 - g_1 - \cdots - g_{L-1}$ be a path in $P_n(g:a \rightarrow a')$, and let $g_i = g$. If the number of inverters on the subpath between g_0 and $g = g_i$ is even, the procedure associates with p the $a \rightarrow a'$ transition. If the number of inverters between g_0 and $g = g_i$ is odd, the procedure associates with p the $a' \rightarrow a$ transition.

For $s27$ with $n = 4$ and 19:0→1 as the target transition, the procedure develops partial paths as shown in Table I. The expected length of each partial path is shown under column *exp*.

Initially, in iteration 0, the procedure finds the partial paths 1, 4 and 6 as shown in the first part of Table I. In iteration 1 the procedure extends the paths into 1-8, 4-19, and 6-14 as shown in the second part of Table I. In iterations 2, 3 and 4 the procedure extends the paths as shown in Table I. The expected path lengths in iteration 4 are 5, 6, 7 and 9. There are four paths with expected lengths 5, 6 and 7. The partial path 1-8-13-14-16, with expected length 9, is removed.

In iteration 6 the procedure removes the partial paths with expected length 8 from P_4(19:0→1). In iteration 8 the procedure finds that all the paths have reached an output. The final set of paths is as shown for iteration 7. The expected path lengths are the actual path lengths.

All the paths include line 19, and the subpaths from the sources of the paths to line 19 do not have any inverters. The set of path delay faults for 19:0→1 associates the 0→1 transition with every path in the last part of Table I.

TABLE I. Example

iteration	paths	exp
0	1	9
	4	5
	6	7
1	1-8	9
	4-19	5
	6-14	7
2	1-8-13	9
	4-19-20	5
	6-14-16	7
3	1-8-13-14	9
	4-19-20-21	5
	6-14-16-19	7
4	1-8-13-14-16	9
	4-19-20-21-22	6
	4-19-20-21-23	6
	4-19-20-21-24	5
	6-14-16-19-20	7
5	4-19-20-21-22-25	6
	4-19-20-21-23-26	6
	4-19-20-21-24	5
	6-14-16-19-20-21	7
6	4-19-20-21-22-25	6
	4-19-20-21-23-26	6
	4-19-20-21-24	5
	6-14-16-19-20-21-22	8
	6-14-16-19-20-21-23	8
	6-14-16-19-20-21-24	7
7	4-19-20-21-22-25	6
	4-19-20-21-23-26	6
	4-19-20-21-24	5
	6-14-16-19-20-21-24	7

Tests for path delay faults can be generated under the robust or non-robust propagation conditions. The non-robust propagation conditions can be classified as weak or strong non-robust. The weak non-robust propagation conditions are used in this paper. Under these conditions, a fault is detected by a two-pattern test that creates the required transition at the source of the path, and assigns non-controlling values to off-path inputs during the second pattern of the test. To increase the likelihood that tests for path delay faults will detect transition faults it is possible to use the strong non-robust propagation conditions. These conditions require, in addition, a 0→1 or 1→0 transition on every line of the path.

III. PROCEDURE FOR FAULT SIMULATION AND TEST GENERATION

This section describes the fault simulation and test generation procedure that is used for studying the possibility of replacing transition faults with path delay faults as a basic delay fault model. The overall flow of the procedure is described in Subsection III.A. The fault simulation part of the procedure is described in Subsection III.B. The test generation part is described in Subsection III.C.

A. Overall Flow

The procedure is applied to a test set T that was generated for transition faults. It is also possible to apply the procedure with $T = \phi$.

For every line g and every $a \in \{0,1\}$, the procedure initially includes the transition $g:a \rightarrow a'$ in a set denoted by F. The transition $g:a \rightarrow a'$ will be removed from F when one of the path delay faults that cover it is detected. The path delay fault coverage is the percentage of transitions such that one of the path delay faults that cover them is detected.

The procedure is applied using increasing values of the parameter n. As n is increased, the sets of path delay faults $P_n(g:a \rightarrow a')$ contain more path delay faults. Therefore, both the fault simulation and the test generation parts of the procedure have additional opportunities to increase the path delay fault coverage.

For every value of n, fault simulation is carried out first. If fault simulation increases the fault coverage by less than a percentage denoted by Δ_0, fault simulation is followed by test generation. When $T = \phi$, $\Delta_0 = 100\%$ ensures that test generation always follows fault simulation. The use of a lower value of Δ_0 when $T \neq \phi$ is based on the following considerations.

For the lowest values of n, the path delay fault coverage increases significantly simply by allowing more path delay faults to be considered for every transition. For higher values of n, fault simulation of increasing numbers of path delay faults is not sufficient for increasing the fault coverage, and test generation is carried out.

The procedure increases n until fault simulation and test generation increase the path delay fault coverage by less than a constant percentage denoted by Δ_1. This is taken as an indication that increasing n further is not likely to affect the path delay fault coverage.

During fault simulation and test generation, the sets of path delay faults $P_n(g:a \rightarrow a')$ are not stored. Instead, they are derived when they are needed. This avoids the storage requirements of multiple subsets of path delay faults. It is feasible since recomputing $P_n(g:a \rightarrow a')$ requires only structural analysis of the circuit and its contribution to the run time is negligible.

B. Fault Simulation

This subsection describes the fault simulation procedure for a given value of n.

For every transition $g:a \rightarrow a' \in F$, the fault simulation procedure finds the set of path delay faults $P_n(g:a \rightarrow a')$. It then performs path delay fault simulation of $P_n(g:a \rightarrow a')$ under every test $t \in T$. Fault simulation of a test $t \in T$ proceeds as follows.

The procedure performs logic simulation of t. It then considers the path delay faults in $P_n(g:a \rightarrow a')$ by order of increasing length. For every path delay fault, the procedure compares the propagation conditions of the fault to the values assigned by t. If all the propagation conditions are satisfied, the procedure concludes that $g:a \rightarrow a'$ is detected, and it removes $g:a \rightarrow a'$ from F without considering additional path delay faults from $P_n(g:a \rightarrow a')$.

C. Test Generation

This subsection describes the test generation procedure for a given value of n.

For every $g:a \rightarrow a' \in F$, the procedure first performs fault simulation of $P_n(g:a \rightarrow a')$ in case one of the tests that were added to T earlier already detects one of the path delay faults from $P_n(g:a \rightarrow a')$.

If none of the faults is detected, the test generation procedure considers up to N_{TG} of the path delay faults from $P_n(g:a \rightarrow a')$, for a constant N_{TG}. This limit was introduced in order to limit the computational effort of the test generation procedure. If the number of path delay faults in $P_n(g:a \rightarrow a')$ is higher than N_{TG}, the procedure selects path delay faults for test generation randomly. It stops when it detects the first path delay fault in $P_n(g:a \rightarrow a')$ or after considering N_{TG} path delay faults.

Due to the selection of path delay faults, the procedure performs several iterations where it considers all the transitions in F until no new fault is detected in an iteration over all the faults in F.

For test compaction and in order to allow the constraints of test data compression to be accommodated, the test generation procedure generates incompletely-specified tests. Every time it generates a new test, it attempts to merge it with one of the tests in T. Only if merging is not possible, the procedure adds the test to T as a new test.

Transition fault simulation is carried out every time a test is updated or a new test is added to T. If T does not detect all the detectable transition faults, the addition of tests for path delay faults to T sometimes increases the transition fault coverage.

IV. EXPERIMENTAL RESULTS

The procedure described in Section III was applied to ISCAS-89, ITC-99 and IWLS-05 benchmark circuits.

The transition fault test set T includes both broadside and skewed-load tests. It is compacted by merging of incompletely-specified tests. The tests are left incompletely-specified, and new tests that are generated for path delay faults may be merged with the transition fault tests. Both broadside and skewed-load tests are generated for path delay faults.

The bound on the number of path delay faults is assigned the values $n = 1, 10, 20, 30, \cdots, 1000$. A fault coverage increase of less than $\Delta_0 = 1\%$ causes test generation to follow fault simulation. A fault coverage increase of less than $\Delta_1 = 0.1\%$ causes the procedure to terminate. The number of path delay faults for which test generation is attempted in every iteration is $N_{TG} = 10$.

TABLE II. Experimental Results (Longest Paths)

circuit	proc	tests	f.c. trf	f.c. pdf	ave paths	ave len all	ave len det	n.time
s1196	fsim.1	238	100.00	45.53	1.84	11.97	11.97	1.00
s1196	fsim.60	238	100.00	89.92	11.34	18.30	15.18	4.55
s1196	tgen.60	294	100.00	99.29	14.73	19.18	15.96	5.52
s1196	tgen.70	295	100.00	99.37	14.77	19.19	15.96	5.64
s1423	fsim.1	125	98.66	33.06	3.60	8.10	8.10	1.00
s1423	fsim.70	125	98.66	50.04	13.95	12.38	9.71	14.82
s1423	tgen.70	320	98.66	91.11	42.84	39.05	36.03	519.36
s1423	fsim.120	331	98.66	92.94	43.90	39.58	36.42	903.39
b04	fsim.1	113	99.30	52.26	4.24	9.67	9.67	1.00
b04	fsim.50	113	99.30	82.88	9.43	12.37	11.01	6.08
b04	tgen.50	190	99.34	99.47	19.85	15.71	14.26	33.23
b04	tgen.60	191	99.34	99.51	19.87	15.72	14.27	37.69
b05	fsim.1	168	93.60	24.22	4.94	13.26	13.26	1.00
b05	fsim.60	168	93.60	38.46	16.17	17.02	15.78	11.75
b05	tgen.60	293	94.24	60.58	56.52	27.21	26.06	313.38
b05	fsim.180	331	94.27	71.05	79.52	29.49	27.95	1168.35
b07	fsim.1	107	96.45	48.59	4.87	10.45	10.45	1.00
b07	fsim.40	107	96.45	78.53	10.23	12.21	11.10	4.75
b07	tgen.40	173	97.22	93.63	17.08	15.35	14.16	41.33
b07	tgen.60	176	97.22	94.13	17.31	15.39	14.19	76.42
b11	fsim.1	135	98.74	26.55	3.83	10.16	10.16	1.00
b11	fsim.60	135	98.74	45.34	21.68	15.84	14.48	15.89
b11	tgen.60	195	98.93	58.40	36.72	20.36	18.78	176.28
b11	fsim.190	215	98.93	68.06	60.18	22.83	20.84	658.11
i2c	fsim.1	55	97.13	63.71	3.56	11.59	11.59	1.00
i2c	fsim.60	55	97.13	91.89	8.55	14.81	12.57	3.96
i2c	tgen.60	97	98.25	98.67	10.99	15.55	13.25	21.91
i2c	tgen.70	98	98.32	98.74	10.98	15.54	13.25	29.91
pci_s_c	fsim.1	95	71.80	28.87	6.79	15.15	15.15	1.00
pci_s_c	fsim.80	95	71.80	66.22	21.96	20.60	16.85	9.63
pci_s_c	tgen.80	283	84.93	95.67	28.77	21.33	17.17	26.30
pci_s_c	fsim.100	286	84.96	95.92	28.91	21.34	17.17	34.78

The procedure from Section III considers the shortest paths. To demonstrate the effects of considering the longest paths, the procedure was modified to consider subsets of path delay faults that are associated with the longest paths for every transition.

The results are shown in Tables II and III as follows. In Table II the procedure considers the longest paths for several circuits. In Table III the procedure considers the shortest paths. There are up to four rows for every circuit corresponding to increasing values of n. In the first row, the results of fault simulation are reported for $n = 1$. In the next two rows, the results of fault simulation and test generation are reported for the first value of n where test generation was applied following fault simulation. The last row corresponds to the last increase in path delay fault coverage. There may be additional calls to the test generation procedure that are not reported, which increase the fault coverage and the number of tests before the last increase in path delay fault coverage is obtained. Every row shows the following information.

Column *proc* shows the procedure applied, and the value of n. Path delay fault simulation has the entry *fsim.n* under column *type*. Path delay fault test generation has the entry *tgen.n* under column *type*.

Column *tests* shows the number of tests in the test set. Column *f.c.* shows the transition fault coverage and the path delay fault coverage.

Column *ave paths* shows the average number of path delay faults that are included in a set $P_n(g:a \rightarrow a')$. The last set computed for every transition is considered. After a transition is removed from F, no additional sets of path delay faults are computed for it, and the value of n for the transition does not increase.

Column *ave len* shows the average length of a path delay fault considering all the path delay faults in the last computed sets $P_n(g:a \rightarrow a')$, and considering the path delay faults whose detection causes a transition to be removed from F.

Column *n.time* shows information about the run time of the procedure. Considering the initial transition fault test set, let the run time for transition fault simulation and path delay fault simulation using $n = 1$ be rt_0. The run time of the procedure is divided by rt_0 to provide an indication of the increase in computational effort as n is increased and due to test generation.

Comparing the results in Table II with the results in Table III it can be seen that the path delay fault coverage is significantly lower for $n = 1$ when the longest paths are considered. This is because large percentages of the longest paths are untestable, while large percentages of the shortest paths are testable. As a result, the procedure for the longest paths considers higher values of n and larger subsets of path delay faults. Even with these values, it achieves lower path delay fault coverage for the same termination condition. For a basic delay fault model that can replace transition faults, it is important to avoid the requirement to consider the longest paths.

In Table III, test generation for path delay faults increases the path delay fault coverage significantly. For all the circuits considered there are transitions for which the corresponding transition fault cannot be detected, but it is possible to detect a path delay fault that covers the transition. For several circuits the path delay fault coverage is higher than the transition fault coverage. These results are important since they imply an increase in the coverage of delay defects when path delay faults are considered.

High path delay fault coverage is obtained in Table III even with low values of n. For these values, a low number of path delay faults is considered on the average for every transition. For most of the circuits, this average is lower than five.

Higher values of n are needed only for a small percentage of the transitions, and they do not increase the average number of path delay faults per transition significantly. Thus, by increasing n gradually, the procedure is able to find an appropriate value of n for every transition such that $P_n(g:a \rightarrow a')$ contains as few path delay faults as possible.

Considering the number of tests that need to be added to a transition fault test set in order to detect path delay faults, the increase is typically manageable. It is higher in cases where the path delay fault coverage

exceeds the transition fault coverage. For example, in the case of $b14$, the transition fault test set with 536 tests achieves 83.55% transition fault coverage, and 1800 tests are required for the path delay fault coverage to reach 92.33%, and exceed the transition fault coverage.

V. CONCLUDING REMARKS

This paper studied the possibility of using path delay faults as a basic delay fault model instead of transition faults in order to improve the ability of a basic test set for delay defects to detect small delay defects. For this purpose, the paper associated a set of path delay faults with every transition. The set included the path delay faults that are associated with the n shortest paths. The value of n was increased gradually in order to consider the smallest possible number of path delay faults for every transition. Experimental results indicated that the path delay fault coverage can exceed the transition fault coverage. The possibility of covering more transitions provides further motivation for targeting path delay faults instead of transition faults.

REFERENCES

[1] Z. Barzilai and B. Rosen, "Comparison of AC Self-Testing Procedures", in Proc. Intl. Test Conf., 1983, pp. 89-94.

[2] G. L. Smith, "Model for Delay Faults Based Upon Paths", in Proc. Intl. Test Conf., 1985, pp. 342-349.

[3] Y. Shao, I. Pomeranz and S. M. Reddy, "On Generating High Quality Tests for Transition Faults", in Proc. Asian Test Symp., 2002, pp. 1-8.

[4] W. Qiu, J. Wang, D. M. H. Walker, D. Reddy, X. Lu, Z. Li, W. Shi and H. Balachandran, "K Longest Paths Per Gate (KLPG) Test Generation for Scan-Based Sequential Circuits", in Proc. Intl. Test Conf., 2004, pp. 223-231.

[5] J. Jiang, M. Sauer, A. Czutro, B. Becker and I. Polian, "On the Optimality of K Longest Path Generation Algorithm under Memory Constraints", in Proc. Design, Autom. & Test in Europe Conf., 2012, pp. 418-423.

[6] K. Fuchs, F. Fink and M. H. Schulz, "DYNAMITE: An Efficient Automatic Test Pattern Generation for Path Delay Faults", IEEE Trans. on Computer-Aided Design, Oct. 1991, pp. 1323-1335.

[7] U. Sparmann, D. Luxenburger, K.-T. Cheng and S. M. Reddy, "Fast Identification of Robust Dependent Path Delay Faults", in Proc. Design Autom. Conf., 1995, pp. 119-125.

[8] S. Kajihara, K. Kinoshita, I. Pomeranz and S. M. Reddy, "A Method for Identifying Robust Dependent and Functionally Unsensitizable Paths", in Proc. VLSI Design Conf., 1997, pp. 82-87.

[9] K. Heragu, J. H. Patel and V. D. Agrawal, "Fast Identification of Untestable Delay Faults Using Implications", in Proc. Intl. Conf. on Computer-Aided Design, 1997, pp. 642-647.

[10] A. Murakami, S. Kajihara, T. Sasao, I. Pomeranz and S. M. Reddy, "Selection of Potentially Testable Path Delay Faults for Test Generation", in Proc. Intl. Test Conf., 2000, pp. 376-384.

[11] Y. Shao, S. M. Reddy, I. Pomeranz and S. Kajihara, "On Selecting Testable Paths in Scan Designs", Journal of Electronic Testing - Theory and Application, Aug. 2003, pp. 447-456.

[12] S. Padmanaban and S. Tragoudas, "A Critical Path Selection Method for Delay Testing", in Proc. Intl. Test Conf., 2004, pp. 232-241.

[13] W.-N. Li, S. M. Reddy and S. K. Sahni, "On Path Selection in Combinational Logic Circuits", IEEE Trans. on Computer-Aided Design, Jan. 1989, pp. 56-63.

TABLE III. Experimental Results (Shortest Paths)

			f.c.		ave	ave len		
circuit	proc	tests	trf	pdf	paths	all	det	n.time
s1196	fsim.1	238	100.00	80.48	1.53	8.21	8.21	1.00
s1196	fsim.20	238	100.00	95.40	3.12	9.51	9.17	1.44
s1196	tgen.20	280	100.00	99.50	3.56	9.89	9.38	2.12
s1423	fsim.1	125	98.66	83.59	1.96	7.20	7.20	1.00
s1423	fsim.30	125	98.66	95.29	3.60	8.43	8.10	1.92
s1423	tgen.30	152	98.66	97.89	4.40	8.88	8.31	142.42
s5378	fsim.1	346	97.86	78.90	3.81	13.26	13.26	1.00
s5378	fsim.30	346	97.86	96.16	5.63	14.50	14.17	2.07
s5378	tgen.30	403	97.87	98.23	6.12	14.65	14.28	10.37
s5378	fsim.40	403	97.87	98.27	6.13	14.65	14.28	10.48
s9234	fsim.1	556	93.22	68.84	2.25	16.06	16.06	1.00
s9234	fsim.30	556	93.22	85.99	5.23	17.88	17.59	2.84
s9234	tgen.30	649	93.23	91.63	7.21	18.90	18.39	616.56
s9234	tgen.40	650	93.23	91.73	7.28	18.92	18.39	853.06
s13207	fsim.1	543	96.80	82.23	2.15	13.78	13.78	1.00
s13207	fsim.30	543	96.80	94.09	4.15	15.67	15.25	3.42
s13207	tgen.30	628	96.81	97.10	4.81	16.15	15.64	10.93
s13207	tgen.60	632	96.81	97.55	5.13	16.24	15.73	18.59
s15850	fsim.1	364	95.11	75.52	2.36	14.37	14.37	1.00
s15850	fsim.30	364	95.11	91.65	4.93	16.71	16.30	2.26
s15850	tgen.30	440	95.12	95.04	6.18	17.23	16.73	294.01
s15850	tgen.70	445	95.12	95.62	6.83	17.33	16.83	1897.23
s35932	fsim.1	115	89.78	78.78	2.41	13.71	13.71	1.00
s35932	fsim.20	115	89.78	89.20	4.02	14.98	14.83	1.62
s35932	tgen.20	119	89.78	89.49	4.10	15.04	14.88	34.53
s35932	fsim.30	119	89.78	89.67	4.22	15.07	14.91	34.87
s38417	fsim.1	453	99.56	79.48	3.13	13.18	13.18	1.00
s38417	fsim.40	453	99.56	92.88	6.53	14.72	14.59	3.35
s38417	tgen.40	662	99.59	98.95	9.48	15.54	15.32	141.84
s38417	fsim.50	662	99.59	99.02	9.53	15.55	15.33	141.95
b04	fsim.1	113	99.30	93.54	5.07	8.43	8.43	1.00
b04	fsim.20	113	99.30	97.98	5.29	8.64	8.59	1.27
b04	tgen.20	119	99.34	99.96	5.39	8.71	8.65	3.93
b05	fsim.1	168	93.60	68.87	3.70	9.30	9.30	1.00
b05	fsim.40	168	93.60	87.38	8.64	10.61	10.50	4.09
b05	tgen.40	223	94.27	91.16	10.63	10.96	10.79	167.74
b05	fsim.90	227	94.30	92.70	12.57	11.11	10.95	329.39
b07	fsim.1	107	96.45	86.72	4.38	7.97	7.97	1.00
b07	fsim.30	107	96.45	95.00	5.60	8.44	8.37	1.83
b07	tgen.30	129	97.13	97.73	6.23	8.60	8.47	20.00
b11	fsim.1	135	98.74	74.81	9.40	10.16	10.16	1.00
b11	fsim.40	135	98.74	94.03	13.39	11.44	11.28	2.67
b11	tgen.40	155	98.93	97.62	14.88	11.72	11.49	15.40
b11	fsim.70	156	98.93	98.11	15.24	11.76	11.52	25.33
b14	fsim.1	536	83.55	56.87	21.34	10.34	10.34	1.00
b14	fsim.60	536	83.55	73.18	28.48	12.47	12.32	7.18
b14	tgen.60	1800	85.58	92.33	38.46	16.38	15.71	298.33
b14	fsim.80	1845	85.58	92.72	38.83	16.44	15.77	421.68
aes_core	fsim.1	269	99.99	61.46	2.46	13.32	13.32	1.00
aes_core	fsim.30	269	99.99	98.15	7.51	15.89	15.23	1.57
aes_core	tgen.30	845	99.99	100.00	8.07	16.05	15.36	47.00
des_ar	fsim.1	123	100.00	39.15	75.07	17.00	17.00	1.00
des_ar	fsim.70	3422	100.00	99.92	101.49	20.36	20.16	640.38
i2c	fsim.1	55	97.13	82.54	2.15	7.03	7.03	1.00
i2c	fsim.20	55	97.13	95.62	3.50	8.05	7.83	1.60
i2c	tgen.20	92	98.39	98.95	3.75	8.26	7.98	14.85
i2c	fsim.30	92	98.39	99.00	3.76	8.27	7.98	14.90
pci_s_c	fsim.1	95	71.80	51.12	2.25	7.67	7.67	1.00
pci_s_c	fsim.40	95	71.80	69.38	6.32	9.45	9.09	3.30
pci_s_c	tgen.40	280	84.70	95.38	12.05	12.00	9.97	14.87
pci_s_c	tgen.70	287	84.86	95.86	12.24	12.03	10.00	26.08
spi	fsim.1	501	99.57	63.17	5.00	8.91	8.91	1.00
spi	fsim.50	501	99.57	88.42	16.08	11.22	11.09	4.30
spi	tgen.50	963	99.82	96.73	19.57	11.80	11.53	1145.20
spi	tgen.60	968	99.82	96.82	19.67	11.81	11.54	1600.89
syscdes	fsim.1	79	99.98	47.30	3.02	9.36	9.36	1.00
syscdes	fsim.100	79	99.98	91.90	34.78	14.93	14.66	40.94
syscdes	tgen.100	384	99.98	99.98	42.30	15.65	15.20	1123.39
wb_dma	fsim.1	97	99.44	86.19	2.91	6.08	6.08	1.00
wb_dma	fsim.20	97	99.44	94.97	3.96	6.93	6.72	1.75
wb_dma	tgen.20	229	99.69	99.78	4.79	7.41	7.07	31.82
wb_dma	fsim.30	229	99.69	99.80	4.80	7.41	7.07	31.85