ALAPTF: A New Transition Fault Model and the ATPG Algorithm*

Puneet Gupta[†] **and Michael S. Hsiao**[‡]
[†]Cadence Design Systems, TDA, Endicott, NY
[‡]Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA

Abstract

The paper presents a new transition fault model called As late As Possible Transition Fault (ALAPTF) Model. The model aims at detecting smaller delays, which will be missed by both the traditional transition fault model and the path delay model. The model makes sure that each transition is launched as late as possible at the fault site, accumulating the small delay defects along its way. Because some transition faults may require multiple paths to be launched, the simple path-delay model will miss such faults. Results on ISCAS'85 and ISCAS'89 benchmark circuits shows that for all the cases, the new model is capable of detecting smaller gate delays and produces better results in case of process variations. For all circuits, on an average, 30% of the time the transition reaches later than traditional models. The algorithm proposed also detects robust and non-robust paths along with the transition faults and the execution time is linear to the circuit size.

1. Introduction

Increasing performance requirements motivated testing for the correct temporal behavior, commonly known as delay testing [1]. Delay faults can be modeled in a number of different ways, among which the most common are the path delay fault (PDF) model [2] and the transition fault (TF) model [3,4]. Test patterns for both transition faults and PDFs consist of a pair of vectors $\{V_1, V_2\}$ where V_1 is required to initialize the target node and V_2 is required to launch the appropriate transition at the target node and propagate it to an observation point, such as a scanned flip-flop (FF) or a primary output (PO). In PDF, cumulative effect of gate delays along the path is considered, whereas TF models excessive delay on a single node in the circuit. A transition fault (both $1\rightarrow0$ and $0\rightarrow1$) can be modeled as 2 stuck-at faults. The test that delivers a rising (falling) transition to a node and sensitizes a path from that node to an observation point will detect a slow-to-rise (slow-to-fall) transition fault at that node. That same test may detect a path delay fault associated with the particular route passing through the node in question. Conversely, a PDF test may also detect some transition faults. Nevertheless, a complete transition test set may not detect all critical paths; likewise, a test set that exercises the longest paths in the circuit may not detect all transition faults.

Various models have been proposed in the past for testing PDFs to cover different aspects of delay tests. PDFs can be broadly classified in two ways [5-6]: robust PDF and non-robust (NR) PDF. A nine-value based ATPG for both robust and an NR test is presented in [7]. DYNAMITE [8] and RESIST [9] give an in-depth analysis of PDFs and uses deterministic approaches for ATPG. DYNAMITE uses a path tree to model the paths, whereas RESIST solves the problem of large paths by producing tests recursively. FSIMGEO [10] is a simulation-based ATPG engine for PDFs, but it may miss the delay faults on the less critical paths. Segment delay faults were proposed and studied in [11-12]. In this the authors study the technique of covering delay defects on untestable critical paths by robustly testing their longest possible segments that are not covered by any of the testable critical path. The disadvantage of this scheme is that there are a large number of untestable critical paths and generating NR tests for all can be futile. A different approach for the selection of critical paths has been presented in [13-14], in which the authors try to generate a longest path passing through each gate. But since the longest path passing through a gate may actually be one of the shorter paths in the circuit, this technique does not guarantee a proper coverage of the critical paths. A statistical based approach is presented in [15-16] where critical paths are selected based on the statistical properties of the already detected paths.

In this paper, we propose a new transition fault model called **A**s **L**ate **A**s **P**ossible **T**ransition **F**ault (ALAPTF) Model. We know that the traditional transition fault model detects gross delays on the circuit nodes. If robust tests are possible for *all* the paths in the circuit, we will not need any additional delay tests. However, since very few paths are robustly testable, there are some delays, which cannot be captured by both transition and non-robust path delay fault models. Consider the condition when there are some small delay defects distributed inside a circuit. If the nodes lie on a robustly untestable path or a less critical path, then the PDF model may miss on those faults. The segment delay fault model might also miss the fault because there might not be a path along which the effect may be propagated. Furthermore, launching of a transition may be neither robust nor non-robust. Hence, we want a model that can accumulate the effect of these small delays on a particular node, which can then be tested by the traditional transition fault model. This accumulation of small delays can be modeled by launching each transition fault as late as possible at the fault site. The notion of ALAPTF can be implemented by making sure that a transition fault is launched via one of the longest robust *segment* ending at the fault site.

The new ALAPTF model differentiates itself from the traditional transition fault model in the way that the new

*conducted at VT and is supported in part by NSF grants 0196470, 0305881 and a grant from NJ Commission Science & Technology.

model is capable of detecting delays of much smaller sizes. At the same time, it can also outperform the PDF tests because not all paths have robust tests. For non-robust tests, only the transition at the start of the path is guaranteed, while transitions along the path are not. Moreover, traditional Transition Fault model tries to launch and propagate the transition along the easiest path, which generally are the shorter paths. The new model launches a transition such that the robust launching segment is the longest possible. Hence, the percentage of gates for which the transition reaches later by using the new ALAPTF model is much higher as compared to either the PDF or the traditional transition fault model. Since we propagate small delays to accumulate at gate nodes, the ALAPTF model is capable of detecting very small delays due to process variations along the circuit nodes. An ideal plot for this measure will look something like Figure 1. In this figure, the X-axis denotes the size of the delay defect

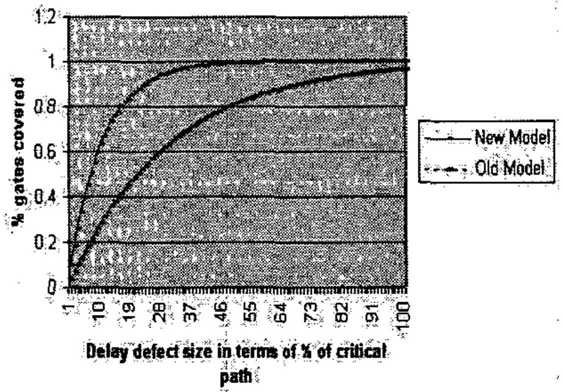

Figure 1. Ideal plot for gate coverage with varying values of delay

on a gate in terms of the percentage of the clock size. So for example if the delay on a gate is 50% the size of the clock, then its delay defect size is 50%. The Y-axis represents the percentage of gates within the circuit under test (CUT) that would be covered. Two curves are shown in the figure: the old (traditional) transition fault model and the new ALAPTF model. Because the traditional model aims at capturing gross delay defects, larger defect sizes will naturally translate to a higher coverage. This is because if a large number of gates have delays that are greater than the clock size (gross delays), then ideally all the defects should be detected by the traditional transition fault model. However, with our new model, we expect the coverage to be much higher for smaller delay sizes also than the coverage by the traditional transition fault model, if not also higher than PDF models. A side benefit for the proposed approach is that the longest robust and non-robust path delay faults through each gate will also be *simultaneously* detected. Results reported in Section 5 show that with the new model, much smaller delay defects could be captured when compared with PDF tests or traditional transition tests. The computation effort is only of the order of circuit size.

The rest of the paper is divided as follows. Section 2 gives the basic definitions and terminology used in the paper. Section 3 describes the proposed ALAPTF model in detail. It also lists the advantages of using the model. Section 4 presents the implementation issues and the methodology used. Section 5 presents the results for combinational and full-scan ISCAS'85 and ISCAS'89 benchmark circuits. Conclusions are given in Section 6.

2. Terminology

A physical path P is an interconnection of gates from a primary input (PI) to a primary output (PO). A rising (falling) path P^r (P^f) is defined as the path corresponding to a rising (falling) transition starting at the PI. The polarity of the transition for each gate on the path depends on the inversion parity along that path. *Path Length* is defined, as the number of gates in a given path P (this can be modified according to the extracted gate and interconnect delays). Segment S is a contiguous section of a path P. A segment can start and end at any point in a given path P.

Given a combinational or a full scan circuit C, a delay defect may manifest as a lumped delay defect on a gate/signal or small-distributed delay over C due to process variations. The *traditional transition fault model* is considered as a logical model and is a good candidate for modeling lumped delay defect. It considers a rising or a falling transition at the inputs and the outputs of logic gates. The vector pair $<V_1,V_2>$ detects a transition fault if V_1 sets the initial value at the fault site and V_2 detects the corresponding stuck-at fault.

A vector pair $<V_1,V_2>$ is said to be a *non-robust path delay* test for a path P, if it launches the transition at the beginning of P, and all off path inputs of P under V_2 have a non-controlling value (NCV). Note that transitions along the path (except at the PI) are not guaranteed for a NR test.

A vector pair $<V_1,V_2>$ is said to be a *robust path delay* test for a path P if (a) it is a NR test for P and (b) whenever the on-path input of a gate G on P takes a NCV under V_1, then all the side inputs of G should have a steady NCV.

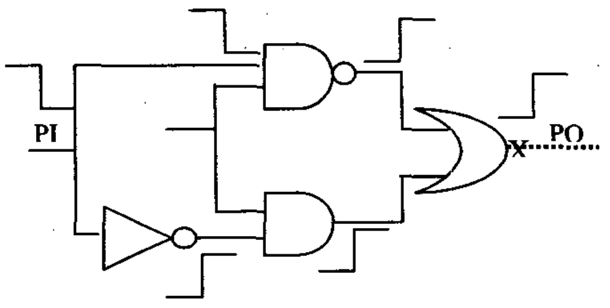

Figure 2. The slow-to-rise fault is launched by a non-robust segment

Lemma 1: A transition fault can be launched robustly, non-robustly, or neither through the segment PI-fault site.
Proof: Consider a slow-to-rise transition fault at the output of the OR gate in Figure 2. This transition can only be launched by having rising transitions at both inputs of this gate. Hence, none of the 2 paths passing through the OR gate are robustly/non-robustly tested.
Lemma 2: A detectable transition fault might not be detected by a robust/non-robust segment starting from the fault site f to a PO.
Proof: Consider the circuit shown in Figure 3. A slow-to-fall transition fault at G1 is propagated to the PO and hence detected, but neither path from G1 to G2 is robustly or non-robustly testable due to off-path inputs at gate G2.

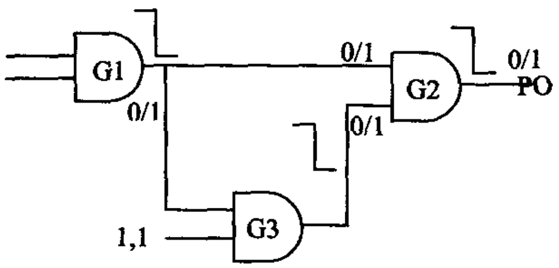

Figure 3. A transition fault need not propagate through a Robust/NR path

The two lemmas above simply conclude that both the launching and propagation of a transition fault can be done through multiple paths and none of the paths may be tested robustly or non-robustly. Hence there are some faults, which can be missed by path fault model and can only be captured by transition fault model. But for small delay defects, an enhanced transition fault model is needed to properly address the aforementioned issues.

3. ALAPTF: The New Fault Model

It was noted in Section 2 that the PDF model is used to capture small-distributed delay defects whereas the transition fault model is used to capture gross delay defects. From Lemma 1 and Lemma 2, because transition fault detection may be neither robust nor non-robust, we can formulate a methodology to combine the small delay defects present before a fault site and propagate them as one transition fault. Note that even the segment delay model may fail, because the overall path may be non-robustly untestable.
To implement this methodology, we propose a new model for transition faults in this paper. Consider the case shown in Figure 4. It shows a structural path P of length L_p (with solid lines). Let us assume that this path is robustly untestable and there are some small delay defects present at node D_1 through D_4, such that $D_1+D_2+D_3+D_4 > T_{clock}$. The robust path delay model cannot capture this delay fault because P is robustly untestable. Moreover, the traditional transition fault model may not capture this fault since each $D_{i\,(i=1,2,3,4)} < T_{clock}$ and in general each transition is launched via a short path from the PIs and its effect is propagated through the shortest propagatable paths to a PO. Hence, we need a fault model, which can group the effect of the entire D_i's and present them as a lumped delay at node D_{i+1}. The traditional transition fault model can then test this lumped delay defect.

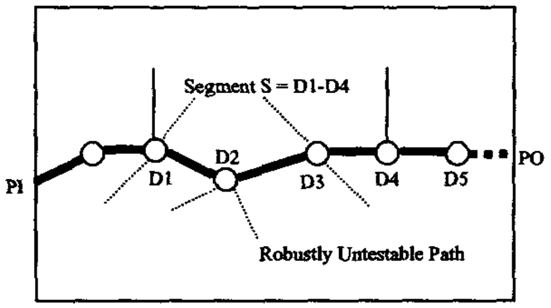

Figure 4. ALAPTF Model

The new transition fault model, which considers the above-mentioned problem, is called the **As Late As Possible Transition Fault** (ALAPTF) Model. A fault f under this model is detected if the following conditions are met:

1) The fault f is launched at the fault site as late as possible. This means that the fault should be launched by one of the longest robust *segment* ending at the fault site. Note that the segment needs not start from a PI. Hence, from Figure 4, we want the fault at gate D_5 to be propagated robustly along the segment $S=D_1$-D_4 if this is the longest testable segment present for D_5. This will ensure that the small delay defects of all the nodes in the robust segment S, gets accumulated at the final node which can then be tested traditionally.
2) The fault is propagated to a PO from one of the longest paths starting from the fault site. This will make sure that we cover long paths of propagation.
3) The traditional transition fault model detects f via this ALAPTF path.

Some of the advantages of using ALAPTF are as follows:

1) Since the transition is launched through one of the longest robust segment, all the small delay defects before the fault site along with the gross delay defects on the fault site will be detected.
2) The new test set will be able to capture defects of much smaller size than traditional transition fault model.
3) The complexity is linear to the circuit size, which is much smaller than PDF or segment delay models, which could be exponential to the circuit size.
4) The algorithm will *simultaneously* detect longest robust and non-robust PDFs (if they exist) through each gate. Note that if the PDFs are not testable, we are still able to launch the transition as late as possible

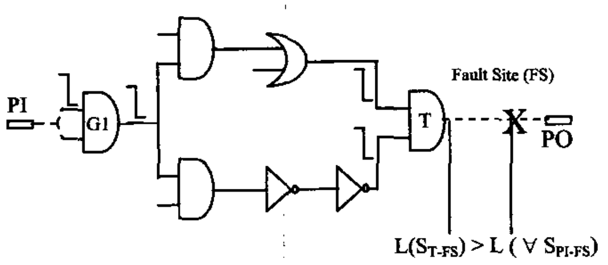

Figure 5. Special case for ALAPTF

neither robustly nor non-robustly as explained in Lemmas 1 and 2.

The special case of ALAPTF model is the condition when the robust segment does not start from a PI. This condition arises when all the robustly testable segments $S_{PI\text{-}FS}$ (from PI to transition fault site) are smaller than a robustly testable segment not starting from a PI. This condition can only occur when there are re-convergent fanouts and the inversion parity at the re-converging gate is same. This condition is explained in Figure 5.

Let as assume that the gate delays are a function of the gate size (its fanouts and fanins). Let us also assume that the path length of any given path is based on the delay sizes of the gates along that path and not simply the topological length. Now from Figure 5 We can see that both the branches of gate G1 re-converge with the same inversion parity at T. Both of the paths are robustly and non-robustly untestable. Hence, under the condition *Length ($S_{T\text{-}FS}$) > Length ($\forall S_{PI\text{-}FS}$)*, the latest transition that can arrive at the Fault Site can come from a non-PI segment, i.e. $S_{T\text{-}FS}$. Note that the two paths are each non-sensitizable, but are co-sensitizable. The difference between our technique and the segment delay model is also evident from Figure 5. Since the segment delay model requires the entire path to be at least non-robustly testable, the transition fault model does not require the segment $S_{FS\text{-}PO}$ to be tested non-robustly (lemma 2).

4. ATPG for ALAPTF Model

The test generation focuses on finding a test for each fault such that the fault is launched as late as possible and is propagated to a PO through a long path. To implement ALAPTF we use a two-step procedure. The first step is a pre-processing step based on simulation, whereas the second is the deterministic ATPG process.

4.1 Pre-processing

The first step towards ATPG is to estimate the latest time at which a fault can be launched at each fault location. This information will be used in the second stage of the ATPG. A simple topological model can also be used (LRB set in [13]) but since a large amount of long paths are robustly untestable, this approach can be very expensive if every topological path needs to be verified for robust sensitizability. To overcome this problem and to estimate a better solution of the problem, we used a Genetic Algorithm (GA) [17-18] based technique. We consider a unit gate delay model (every gate has a delay of one unit) for the pre-processing step. However, other more elaborate delay models can be easily incorporated also. The pseudo algorithm is as follows:

```
Void pre_processing() {
  For each gate g {
    Late_1[g] = GA (g,1) //a 'one' should reach g ALAP
    //fitness = time at which 'one' reaches the gate
    Late_0[g] = GA (g,0) //a 'zero' should reach g ALAP
    //fitness = time at which 'zero' reaches the gate
  }
}
```

The method finds a time that a logic 1 and logic 0 can reach each gate. Since it is a unit gate delay model, the latest time will directly correspond to the structural length of the segment. For example, if we find out that for a gate *g*, a logic '1' will reach latest at time unit 10, then there is definitely a segment S of length 10 ending at *g* such that we can propagate a robust rising transition at *g* along segment S. The primary difference between finding the LRB set [13] for each gate and this approach is that the length produced for each gate is simply not the topological maximum. Moreover, the lengths produced are indicative of the segment lengths that may or may not start from a PI. Since we are using a GA based approach, the lengths produced are not guaranteed to be the maximum. But, the length L produced here for a gate G guarantees that there exist a segment of length L ending at G, such that, we can propagate a robust transition along it.

4.2 ALAPTF ATPG

The second step towards finding the vector set is the following deterministic ATPG algorithm. To produce a test for all the transition faults so that ALAPTF criterion is satisfied, we use a reverse approach, i.e. first generate a test for a given path and then drop all the faults detected via the produced test vector. We use an approach similar to that of RESIST [9] to generate tests for all the paths starting from a given PI. Since we want the transition to launch and propagate from a long path, we generate test for all the paths and drop the faults once a vector pair is generated. The following steps explain the algorithm.

1) Since the vector V2 is same for both robust and non-robust test for a given path, we generate V2 first via a recursive search.
2) Many paths starting from the same PI are structurally similar; hence we use a RESIST-based approach to produce a test for a path, by reducing re-computation of side-inputs for the common segments along the paths. In our algorithm, we restrict ourselves to finding V2 only. Our approach for generating V2's for a given PI and transition, for all paths starting from PI, can be understood by the flowchart of *Figure 6*.
3) After generating V2, compute the values required by the gates on P by V1 so that P becomes robustly testable and store them in Set_V1.

4) Since we want to launch every transition ALAP, we want to satisfy as many contiguous values as possible in the set Set_V1.
5) If all the values of Set_V1 are satisfied, then the test becomes a robust test for P. If the first value is satisfied (which is simply an assignment to PI), then it is a non-robust test for P.
6) Drop all the transition faults detected by the vector pair <V1, V2> if the faults satisfy all the conditions required by the ALAPTF model, i.e.
    a) The fault should be detected in the traditional sense.
    b) The fault should be launched by at least a segment of length ≥ the length found in the pre-processing step.
    c) The fault should be propagated through one of the longest paths through the fault site.

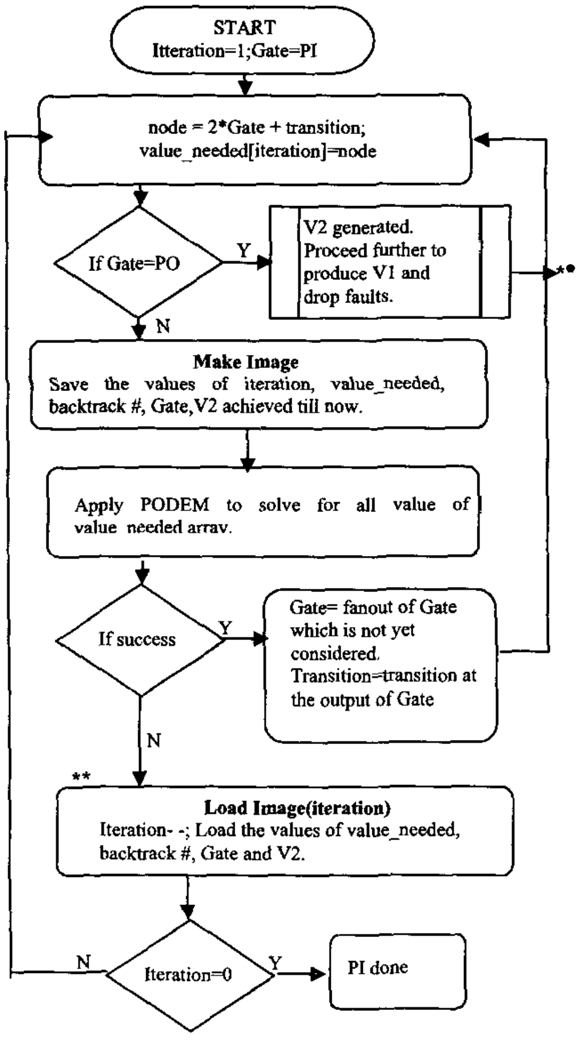
START
Itteration=1;Gate=PI
node = 2*Gate + transition;
value_needed[iteration]=node
If Gate=PO
Y
N
V2 generated. Proceed further to produce V1 and drop faults.
*
Make Image
Save the values of iteration, value_needed, backtrack #, Gate,V2 achieved till now.
Apply PODEM to solve for all value of value needed array.
If success
Y
N
Gate= fanout of Gate which is not yet considered.
Transition=transition at the output of Gate
**
Load Image(iteration)
Iteration- -; Load the values of value_needed, backtrack #, Gate and V2.
Iteration=0
N
Y
PI done

Figure 6. Flowchart for finding V2 for all paths starting from a given PI.

7) If no test is found for a fault *f* such that it satisfies all 3 criterions then we add a vector that at least detects the fault traditionally.
8) For circuits such as c6288 where there are about 2^{17} potentially testable paths, we abort on a PI as soon as we have tested at least 5000 paths starting from it.

At the end of the process we have a test set that has a maximum robust and non-robust coverage (if no paths are aborted) along with high transition fault coverage (TFC). If we want to have high TFC alone, we can simply keep the vectors that detect at least one fault in step 7 of the algorithm.

5. Experimental Results

This section presents the results for combinational and full scan sequential ISCAS'85 and ISCAS'89 benchmark circuits. The programs were written using C++ and were experiments were conducted on a 1.8GHz, Pentium 4, 512MB RAM machine, running the Linux operating system.

Table 1. Results of the proposed ATPG Algorithm

Circuit	TFC(%)	# of Robust Paths Detected*	# of NR Paths Detected*	Time(s)
C880	100.0	16489	16652	19.4
C1355	99.7	200462	841613	4533.2
C2670	85.06	33156	130638	4287.0
C5315	99.54	186635	341634	10564.2
C6288†	99.18	45853	305894	2223.1
C7552	95.74	192021	274920	3531.2
S641	99.05	2092	2266	8.41
S713	94.2	2066	4922	31.4
S1196	100.00	3710	3759	9.73
S1423	99.17	33981	45182	210.7
S5378	97.87	19398	21919	306.2
S9234	89.78	38593	59830	6615.9
S35932	90.5	38372	58657	4891.7

* If all the vectors are considered. † Only 5000 paths per PI are considered.

Table 1 presents the results obtained by our approach for ISCAS'85 and full-scan ISCAS'89 benchmark circuits. Second column reports the transition fault coverage for the corresponding circuits. This number represents the maximum coverage obtained for the corresponding circuits. Most of the detected faults satisfy the ALAPTF criterion. But some of them were detected by relaxing the detection criterion since those faults do not satisfy the first two ALAPTF detection criteria. We can see that for all the circuits, the TFC reached is the maximum TFC that can be achieved. The third and the fourth columns give the robust and the NR coverage. Since the ATPG is a deterministic algorithm, the path coverage is the maximum for most of the cases. For c6288, we aborted on each PI node after every 5000 paths. The last column reports the execution time in seconds. The time is under limits even in the case of circuits such as c1355 and c5315, which have a large number of paths. This is because of the recursive nature of the algorithm due to which we need not generate test for each path from the start every time. Hence, one of the advantages of using this algorithm is that in only one pass, we can find tests for all three models of delay tests.

To validate that the new test set T_{new} launches the transitions as late as possible, we compared T_{new} with the test set (T_{old}) produced by the traditional transition fault ATPG. Figure 7 plots the result for various circuits. The y-axis represents the percentage of gates on which a transition reaches later by T_{new} as compared to T_{old}. This shows that T_{new} is indeed better in performance that T_{old}. There are on average 30% gates for which transition reaches later by T_{new}. Each test set was then simulated for the same parameter against a path delay fault test set T_{path}. This set was produced by the method presented in [13]. Figure 8 shows the corresponding result. Here we can see that for roughly 20% of the gates, the transitions were launched later. Moreover, the technique presented in [13] was not able to generate tests for circuits having a large number of long untestable paths (e.g c6288). But the

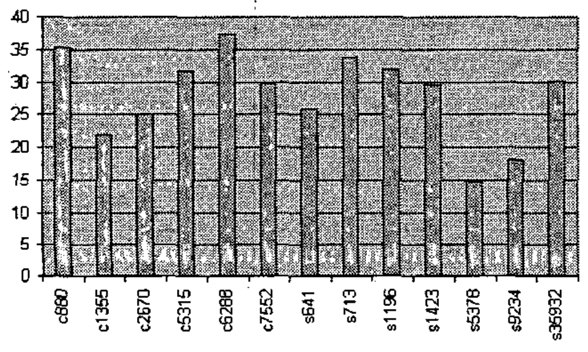

Figure 7. Percentage of gates for which transition reaches later by using T_{new} compared with T_{old}.

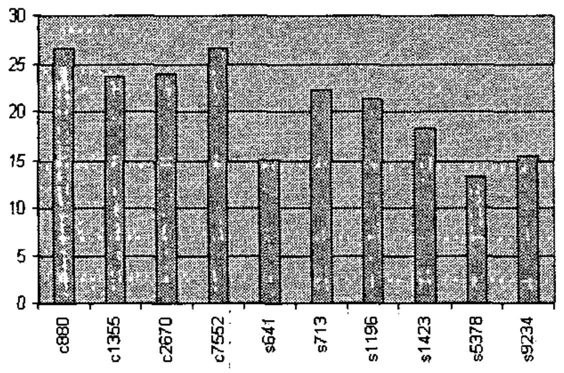

Figure 8. Percentage of gates for which transition reaches later by using T_{new} compared with T_{path}.

technique presented here is able to generate test for these circuits as well.

In order to show that the test set T_{new} can detect small delays as compared to T_{old}, we inserted random delays on each gate. We then simulate the good and the faulty circuit by T_{new} and T_{old} to see if the defects could be captured by either of the test sets. The sizes of the defects were varied for this experiment. We considered a delay model in which the nominal delay of each gate g was kept equal to $D_g = 5+$ # of fanin(g)+ # of fanout(g). This model can be changed and we can also use real values of delay of each gate. A Gaussian distribution with standard deviation (SD) of 10.0 was used to generate random delay sizes. Wichman-Hill transform [18] was used to generate a uniform random variable, which was then converted to a Gaussian random variable by using the Box-Muller [19] method. Table 2 reports the results for some of the benchmark circuits. Let the nominal delay of the critical path in the circuit be L_d. Then, for the second column (1%) of Table 2, we shift the Gaussian probably distribution curve by an amount equal to 1% of L_d. This means that the amount of delay added on each gate is about 1% of L_d and can vary up to ± 10 units (because of SD). We can see that as the size of the delay increases, the number of gates for which we get a faulty response increases with both T_{new} and T_{old}. But T_{new} is capable of detecting delay faults on more number of gates even for smaller values of added delays. As the delay approaches 100%, both the test sets will detect the same number of gates, since both have the same transition fault coverage.

Some of the results of Table 2 are also plotted in a under Figure 9. We can see that the curves of T_{old} and T_{new} are apart at the beginning and become closer as the delay size reaches a large value. The curves should be compared with the predicted curves of Figure1. It is evident that the plots of Figure 9 closely resembles to Figure 1.

Table 2. # of faults detected by varying amounts of added delay

Ckt	1%		5%		10%		50%		80%	
	T_o	T_N	T_o	T_N	T_o	T_N	T_o	T_N	T_o	T_N
C880	0	1	0	68	54	80	249	258	350	350
S641	0	0	0	0	0	86	155	243	338	344
S713	0	0	0	0	0	0	123	231	347	356
S1196	0	23	29	71	57	86	322	323	533	534
S1423	0	60	0	82	0	100	67	345	403	496
S5378	21	27	43	87	149	180	2328	2346	2849	2855

6. Conclusion

A new transition fault model called ALAPTF is presented. The new model has been shown to perform better than the existing transition fault model and the path delay fault model. At the same TFC as produced by the older models,

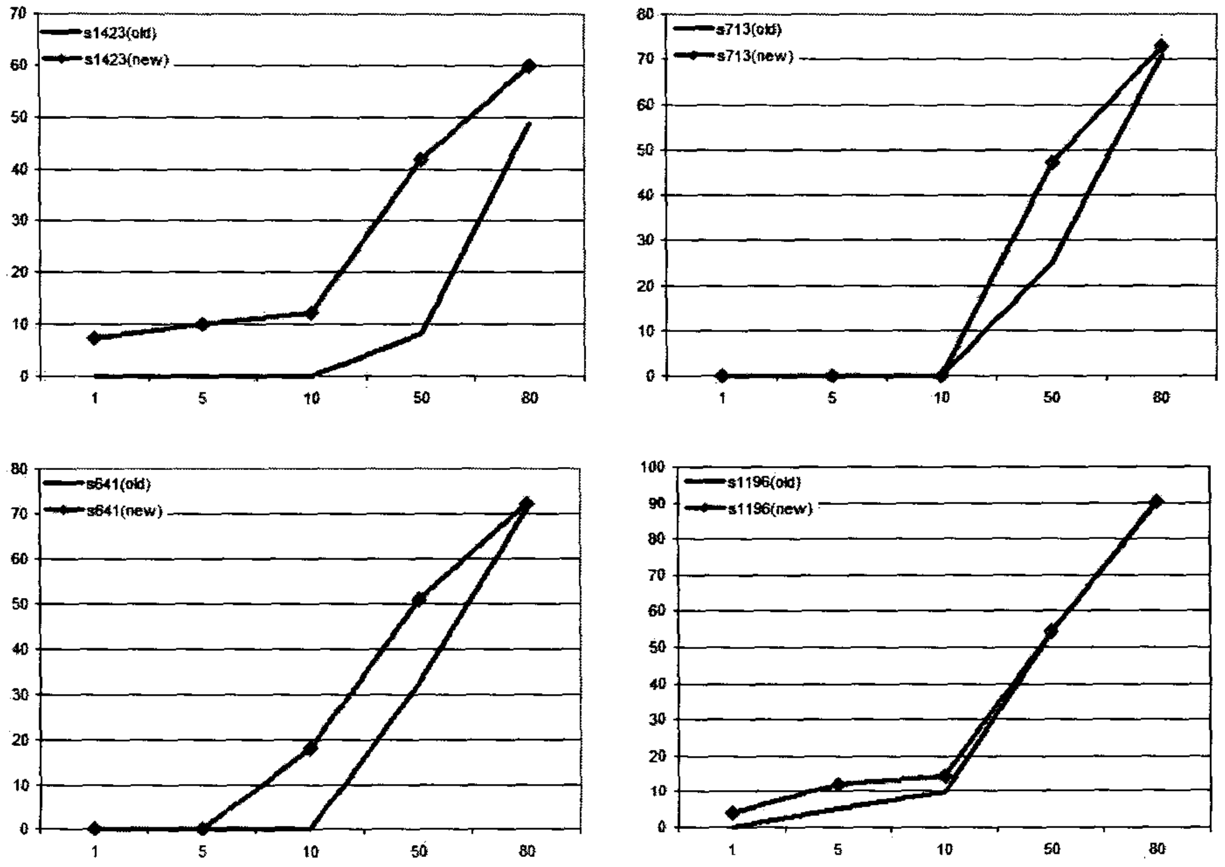

Figure 9. Percent gate coverage with varying amounts of delays

the ALAPTF model, finds a test set which launched the transitions on each gate as late as possible. For about 30% times the transition reaches later by using the new fault model. It is also capable of detecting much smaller delays (delays much smaller than the critical path delays). The execution times are of the order of the circuit size and the final test set also has a high robust and non-robust coverage.

7. References

1. A. K. Majhi and V.D. Agrawal, "Delay Fault Model and Coverage", Proc. VLSI Design Conference, 1998, pp 688-699.

2. G. L. Smith "Model for delay faults based upon paths," ITC 1985

3. J. Savir and S. Patil, "On Broad Side Delay Test", Proc. VLSI Test symposium, 1994, pp 368-372.

4. J. Savir and S. Patil, "Scan-Based Transition Test," IEEE Trans. on CAD of Integrated Circuits and Systems, 1993, pp 1232 - 1241

5. K. T. Cheng and H. C. Chen, "Classification and identification of non-robust untestable path delay faults", IEEE Trans. CAD of Integrated Circuits and Systems, 1996, pp 845 - 853

6. E. S. Park and M. R. Mercer, "Robust and Nonrobust Tests for Path Delay Faults in a Combinational Circuit", ITC, 1987.

7. A. K. Majhi, J. Jacob, L.M. Patnaik, and V.D. Agrawal, "An efficient automatic generation system for path delay faults in combinational Circuits", VLSI Design Conference, 1995, pp 161-165.

8. K. Fuchs, F. Fink, and M. Schulz, "DYNAMITE: An efficient automatic test pattern generation system for path delay faults ", IEEE Trans. CAD

of Integrated Circuits and Systems, 1991, pp 1323 – 1335.

9. K. Fuchs, M. Pabst, and T. Rossel, "RESIST: A recursive test pattern generation algorithm for path delay faults considering various test classes", IEEE trans. Computer-Aided Design of Integrated Circuits and Systems, 1991, pp 1550-1562.

10. S. Yihe and W. Qifa, "FSIMGEO: A Test Generation Method for Path Delay Fault Test Using Fault Simulation and Genetic Optimization", Proc. ASIC/SOC Conference, 2001, pp 225-229.

11. M. Sharma and J.H. Patel, "Testing of Critical Paths for Delay Faults", Proc. ITC, 2001, pp 634 - 641

12. K. Heragu, J. Patel, and V. D. Agarwal, "A Test Generator for Segment delay Faults", VLDI design 1999, pp 484-491.

13. M. Sharma and J.H. Patel, "Finding a small set of longest testable paths that cover every gate", Proc. ITC, 2002, pp 974-982 .

14. W. Qiu and D.M.H. Walker, "An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit", Proc. ITC, 2003, pp 592 – 601.

15. J.J. Liou, L.C. Wang, and K.T. Cheng, "On theoretical and practical considerations of path selection for delay fault testing", ICCAD 2002, pp 94-100.

16. J.J. Liou, A. Krstic, L.C. Wang, and K.T. Cheng, "False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation", Proc DAC 2002, pp 566-569.

17. M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Application of genetically-engineered finite-state-machine sequences to sequential circuit ATPG", IEEE Trans. on CAD of Integrated Circuits and Systems, 1998, pp 239-254.

18. E. M. Rudnick, J. H. Patel, G.S. Greenstein, and T. M. Niermann, "A Genetic Algorithm Framework for Test Generation," IEEE Trans. on CAD of Integrated Circuits and Systems, 1997, pp 1034 – 1044.

19. W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, Numerical Recipes in C: The Art of Scientific Computing, (Cambridge University Press, Cambridge, 1988).