Novel ATPG Algorithms for Transition Faults *

Xiao Liu, Michael S. Hsiao
The Bradley Dept Electrical & Computer Engineering
Virginia Tech

Sreejit Chakravarty, Paul J Thadikaran
Intel Architecture Group
Intel Corporation

Abstract

This paper proposes novel algorithms for computing test patterns for transition faults in combinational circuits and fully scanned sequential circuits. The algorithms are based on the principle that s@ vectors can be effectively used to construct good quality transition test sets. Several algorithms are discussed. Experimental results obtained using the new algorithms show that there is a 20% reduction in test set size, test data volume and test application time compared to a state-of-the-art native transition test ATPG tool, without any reduction in fault coverage. Other benefits of our approach, viz. productivity improvement, constraint handling and design data compression are highlighted.

1 Introduction

The stuck-at (s@) fault model [1] does not model speed related failures very well. At the same time, higher clock rate, shrinking geometries, increasing metal density, etc. are resulting in defects that cause speed failures. This has prompted researchers to propose a variety of fault models for speed failures, viz.: *transition fault* [2], *path delay fault* [3], and *segment delay fault* [4].

A transition fault at node X assumes a large delay at X such that the transition at X will not reach the latch or primary output within the clock period. The path delay fault model assumes a small delay at each gate. It models cumulative effect of gate delays along a specific path, from a primary input to a primary output. If the cumulative delay exceeds the slack for the path, then the chip fails. Segment delay fault targets path segments instead of complete paths. Of these, transition fault is the most practical, and commercial tools are available for computing such tests. In this paper we focus on ATPG for transition faults.

At each line in the circuit two transitions are possible: *slow-to-rise* and *slow-to-fall* . Test pattern for a transition fault consists of a pair of vectors {V1,V2 } where: V1 (*initial vector*) is required to set the target node to an initial value and V2 (*test vector*) is required to launch the appropriate transition at the target node and also propagate the fault effect to a primary output [2, 9].

* This research was supported in part by a grant from Intel Corp.

Transition tests can be applied in three different ways: **Broadside** [6], **Skewed-Load** [7] and **Enhanced-Scan** [8]. For broadside testing (*also called functional justification*), a vector is scanned in and the functional clock pulsed to create the transition and subsequently capture the response. For each pattern in broadside testing only one vector is stored in tester scan memory. The second vector is derived from the first by pulsing the functional clock. For skewed-load transition testing, an N-bit vector is loaded by shifting in the first N-1 bits, where N is the scan chain length. The last shift clock is used to launch the transition. This is followed by a quick capture. For skewed-load testing also, only one vector is stored for each transition pattern in tester scan memory; the first vector is a shifted version of the stored vector. Finally, for enhanced-scan transition testing, two vectors (V1, V2) are stored in the tester scan memory. The first scan shift loads V1 into the scan chain. It is then applied to the circuit under test to initialize it. Next, V2 is scanned in, followed by an apply and subsequently a capture of the response. During shifting in of V2 it is assumed that the initialization of V1 is not destroyed. Therefore enhanced-scan transition testing assumes a hold-scan design [8]. In this paper we consider only enhanced-scan testing.

A number of work on generating patterns for transition faults and compacting them has been reported. A dynamic compaction technique [11] and a static compaction technique for enhanced-scan transition patterns [10] to generate small and comprehensive test sets for CMOS stuck-open and transition fault models were proposed. It reordered the test vectors in the test set to maximize the number of faults detected by adjacent patterns. In [5], two algorithms, called Redundant Vector Elimination (RVE) and Essential Fault Reduction (EFR), along with dynamic compaction were presented to reduce the size of two-pattern test set.

Our approach to generate enhanced-scan transition tests is to re-use a given set of s@ vectors. It is based on the observation that there is a close correlation between s@ tests and transition fault patterns. Furthermore, the set of all transition patterns composed from only the given set of s@ vectors can potentially achieve transition fault coverage bounded by the s@ coverage. Compared to a state-of-the-art transition fault ATPG tool, our results are 20% smaller. This implies a 20% reduction in tester memory usage and test application time.

The rest of the paper is organized as follows. Motiva-

Table 1. Storage Increase.

$circuit$	$Stuck-at$ $vectors$	$Transition$ $Patterns$	$Expansion$
C1908	129	263	4.08
C2670	116	198	3.41
C3540	179	366	4.09
C5315	124	248	4
C6288	36	90	5
C7552	237	378	3.19
S5378	266	490	3.68
S9234	410	837	4.08
S13207	485	1002	4.14
S15850	464	924	3.98
S35932	75	137	3.65
S38417	1017	1927	3.79
S38584	727	1361	3.74

tion for enhanced scan testing is discussed in Section 2. Section 3 explains the bounding of the transition fault coverage based on s@ test set. In Section 4 heuristics to compute a much smaller set of transition patterns with the achievable coverage is discussed. Experimental result comparing these heuristics is discussed in Section 5. Section 6 compares results to a state-of-the-art transition fault ATPG tool. In addition to coverage, the fact that we are reusing the s@ patterns has a number of additional benefits pertaining to productivity improvement, constraint handling and design data compression. These are discussed in Section 7. Finally, Section 8 concludes the paper.

2 Why Enhanced-Scan Transition Tests?

Enhanced scan transition test has two primary advantages: coverage and test data volume. Among the three types of transition tests, broadside tests suffer from poor fault coverage [6]. Enhanced-scan, in general, gives better coverage than skewed-load transition test. This stems from the fact that, unlike skewed-load tests, there is no dependency between the two vectors in enhanced-scan test pattern.

Tester memory requirement is also important, and considerable attention is being paid to reduce the tester memory requirement for s@ tests. The problem is far worse for transition tests as the following data shows. In [13] it was reported that for skewed load transition tests for an ASIC, the s@ vector memory requirement was 8.51M versus 50.42M for transition test. This implies an increase of a factor of 5.9. This increase in tester memory requirement is also true if enhanced-scan transition tests are used. We generated s@ and enhanced-scan transition tests for the ISCAS85 and ISCAS89 benchmark circuits using a state-of-the-art commercial ATPG tool, and the results are shown in Table 1. Both s@ and transition test vectors were generated with compression option turned on. Note that each pattern for enhanced-scan transition test requires two vectors. The data shows an expansion of anywhere between 4X and 5X in tester memory requirement for transition tests when compared with that required for s@ tests.

The downside of using enhanced-scan transition test is that special scan-design, viz. hold-scan, is required. For designs using custom logic, such as high performance microprocessors, the circuit is often not fully decoded. In such a scenario, to avoid contention during the scan-shift phase, hold-scan design is used even for applying s@ tests. Therefore, for such designs enhanced-scan transition tests is preferred. This is our motivation for investigating good ATPG techniques for enhanced-scan transition tests.

Table 2. Results for Exhaustive Patterns.

Circuit	[12] Vec	S@ FC(%)	Trans FC(%)	Exhaust Pairwise Vec	Exhaust Pairwise TFC(%)
c880	128	100	95.51	16256	100
c1355	198	99.77	94.23	39006	99.77
c1908	143	99.67	93.03	20306	99.67
c3540	202	96.29	88.68	40602	96.27
c5315	157	99.56	96.91	24492	99.54
c6288	41	99.42	97.60	1640	99.19
S344	31	100	89.31	930	100
S382	41	100	90.50	1640	100
S526	82	99.93	90.99	6642	99.93
S832	179	99.20	82.12	31862	99.20
S1196	197	99.97	87.52	38612	99.97
S1423	97	99.11	95.12	9312	99.11
S5378	332	98.77	93.65	109892	98.40
S35932	78	90.50	89.97	6006	90.50
S38417	1207	99.67	97.59	1455642	99.66
S38584	893	95.34	91.63	796556	95.02

3 Achievable Coverage using S@ Tests

There is a close correlation between s@ vectors and enhanced-scan transition patterns. The condition to detect a transition fault is more stringent than for s@ faults in that an extra initialization vector is required. For example, a transition pattern for *slow-to-fall* fault on line L requires an *initial vector* that *initializes* L to 1, done by a test vector that excites L s@0, and a *test vector* for detecting fault L s@1. The resulting vector pair causes a falling transition and propagates the fault effect to an observable node. Transition pattern for L *slow-to-rise* can be similarly composed.

If the s@ fault coverage of a test set T is α, then the enhanced-scan transition pattern set derived by using only vectors in T has transition fault coverage of at most α. To understand this, suppose we have an undetected s@-0 fault, then there must be at least one slow-to-rise transition fault that cannot be detected no matter how we rearrange the test vectors in the test set. Therefore, the s@ coverage of a given test set T is the upper bound for transition fault coverage, if the transition patterns are composed from vectors in T.

To determine the maximal transition coverage obtainable by exhaustively pairing all s@ vectors was computed. Results are shown in Table 2. STRATEGATE [12] stuck-at test sets were used. Columns 3, 6 of Table 2 imply that by including all possible combination of vector pairs, the transition fault coverage reach the s@ coverage in many cases. In a few cases, such as C3540, C5315, S5378 and S38584, the transition fault coverages are slightly lower. However, the differences are negligible.

Proceedings of the Seventh IEEE European Test Workshop (ETW'02)
1530-1877/02 $17.00 © 2002 IEEE

4 Composing Transition Tests from S@ Tests

The staggering number of exhaustive pairwise vectors in Table 2 clearly make it infeasible to use all such pairs as transition patterns. To alleviate this high cost, we present two heuristics to select a considerably smaller set of s@ vectors to achieve the same transition fault coverage.

4.1 Fault-list-based Extension

This technique is based on a greedy approach, illustrated by the following example. Let us consider a circuit with four gates (eight s@ faults) and a test set consisting of 5 vectors V1, V2, V3, V4, V5. The excited and detected s@ faults by the test set (simulated with fault-dropping) are shown in Table 3.

Table 3. Dictionary with fault-dropping.

$Vector$	$Excited\ Faults$	$Detected\ Faults$
V1	a-s-1,b-s-0,c-s-0,d-s-1	a-s-1,b-s-0
V2	b-s-1,c-s-1,d-s-1	c-s-1,d-s-1
V3	a-s-0,c-s-0	a-s-0,c-s-0
V4	b-s-1	b-s-1
V5	d-s-0	d-s-0

By applying the test sequence in the given order, only two transition faults (falling-transition fault and rising-transition fault on gate c) can be detected. However, we can obviously see that the following vector-pairs can detect additional transition faults:

1. (V3,V1) detects falling-transition on a;
2. (V1,V3) detects rising-transition on a;
3. (V1,V4) detects falling-transition on b;
4. (V2,V4) detects rising-transition on b;
5. (V5,V2) detects falling-transition on d;
6. (V1,V5) detects rising-transition on d;

Including these 6 additional test patterns (instead of all possible 20 pairs) is sufficient to detect all transition faults.

Next, we describe how these 6 test patterns are selected. First, we pick an undetected transition fault f. Without loss of generality, assume the fault f slow-to-rise. We look for the first vector that sets node f to logic 0, and then complete this vector-pair by searching for the first vector that detects the s@-fault f s-a-0. In the above example, for the fault a slow-to-fall, the first vector that sets a to logic 1 (excite a s@-0) is V3. The first vector in the dictionary that detects a s-a-1 is V1. Thus the pair (V3, V1) is constructed for the transition fault a slow-to-fall. The algorithm then picks the next undetected transition fault. The complete procedure for the fault-list-based approach is outlined below:

1. Assume that we have a s@ test set $T = \{T_1, \ldots, T_N\}$. Perform transition fault simulation using original test set $\{\langle T_1, T_2\rangle, \langle T_2, T_3\rangle, \ldots, \langle T_{N-1}, T_N\rangle\}$. Compute $undet\_TR$, the set of undetected transition faults.
2. Compose the useful subset U of the s@ faults implied by $undet\_TR$ as follows. If X *slow-to-rise* or *slow-to-fall* fault $\in undet\_TR$ then both X s@-0 and s@-1 are included in U.
3. Perform s@ fault simulation using T on the s@ faults in U. For each undetected s@ fault in U, record the *first vector* in T that excites it (*greedy initial vector*) and the *first vector* that detects it (*greedy test vector*).
4. Iterate through the transition faults in $undet\_TR$, and for each fault add a test pair to the transition test set. If the fault is X *slow-to-rise* then add the vector pair $\langle V_i, V_j\rangle$ where: V_i is the *greedy initial vector* for X s@ 1; and V_j is the *greedy final vector* for X s@ 0. If the fault is X *slow-to-fall* then add the vector pair $\langle V_i, V_j\rangle$ where: V_i is the *greedy initial vector* for X s@ 0; and V_j is the *greedy final vector* for X s@ 1.

4.2 Priority-based Extension

In the previous *fault-list-based* algorithm, only one excitation vector and one detection vector is recorded per fault in the dictionary. Thus, only one vector pair can be formed for a transition fault. Table 4 shows the excitation and detection dictionary obtained by simulation *without* fault-dropping on the same circuit as in Table 3. If we consider all the combinations of test vectors for transition faults in Table 4, we can see that:

Table 4. Dictionary without fault-dropping.

$Vector$	$Excited\ Faults$	$Detected\ Faults$
V1	a-s-1,b-s-0,c-s-0,d-s-1	a-s-1,b-s-0
V2	b-s-1,c-s-1,d-s-1	c-s-1,d-s-1
V3	a-s-0,c-s-0	a-s-0,c-s-0
V4	a-s-0,b-s-1	a-s-0,b-s-1
V5	a-s-0,d-s-0	a-s-0,d-s-0

1. (V3,V1), (V4,V1), (V5,V1) detect falling-trans. on a;
2. (V1,V3), (V1,V4), (V1,V5) detect rising-trans. on a;
3. (V1,V4) detects falling-transition on b;
4. (V2,V1), (V4,V1) detect rising-transition on b;
5. (V5,V2) detects falling-transition on d;
6. (V1,V5), (V2,V5) detect rising-transition on d;

If we select the test patterns intelligently, 4 test patterns (V1,V4), (V1,V5), (V5,V2) and (V4,V1) suffice to detect all transition faults.

The assumption behind this heuristic is that a pattern which detects a hard-to-detect fault may detect many other faults as well (some could be easy-to-detect). The heuristic, named *Priority-based extension algorithm*, processes the hard-to-detect faults first. The **priority of a transition fault** at node N, with respect to the s@ test set T, is defined to be the number of times node N s@-0 and N s@-1 faults are detected by the test set T. The transition fault with the lowest priority is one that is most difficult to detect and is therefore at the top of the list.

Table 5. Results for Fault-List-Based Extension.

$Circuit$	$Stuck-at$ $test\ vectors$	$Trans$ $Patterns$	$Exten$ $time(s)$	$Comp$ $Patterns$	$Comp$ $time(s)$	$Trans$ $FC(\%)$	$Total$ $Time$
c880	128	333	0.23	256	0.24	100	0.47
c1355	198	587	0.43	420	0.47	99.77	0.90
c1908	143	580	0.41	437	0.74	99.67	1.15
c3540	202	1057	1.52	796	3.91	96.27	5.43
c5315	157	628	1.98	556	3.00	99.54	4.98
c6288	41	302	1.30	229	1.50	99.19	2.80
S344	31	132	0.04	102	0.03	100	0.07
S382	41	166	0.04	114	0.07	100	0.11
S526	82	285	0.10	224	0.13	99.93	0.23
S832	179	662	0.35	475	0.41	99.20	0.76
S1196	197	686	0.56	521	0.71	99.97	1.27
S1423	97	382	0.34	288	0.48	99.11	0.82
S5378	332	1397	2.98	1109	5.08	98.40	8.06
S35932	78	507	11.34	417	22.70	90.50	34.04
S38417	1207	4654	52.36	3805	100.76	99.66	153.12
S38584	893	4386	68.33	3475	175.62	95.02	243.95

Because fault-simulation without fault-dropping could be expensive, we restrict simulation of only those undetected transition faults missed by the original s@ test set. This more complete dictionary also enables us to identify essential vectors that must be included in the final test set. The priority-based extension algorithm is described next.

1. Assume that we have a s@ test set $T = \{T_1, \ldots, T_N\}$. Perform Transition Fault Simulation on the original test set $\{\langle T_1, T_2\rangle, \langle T_2, T_3\rangle, \ldots, \langle T_{N-1}, T_N\rangle\}$. Record all undetected transition faults $undet\_TR$.
2. Compose the useful subset U of the s@ faults implied by $undet\_TR$ as follows. If X *slow-to-rise* or *slow-to-fall* fault $\in undet\_TR$, then both X s@-0 and s@-1 are included in U.
3. Perform s@ fault simulation, *without fault dropping*, using T on the s@ faults in U. For each transition fault in $undet\_TR$ calculate its priority index.
4. Sort the transition faults in $undet\_TR$ in increasing order of their priority.
5. For each target transition fault:
   (a) If its priority number is 0, then mark the fault as undetectable (by the given s@ test).
   (b) Else, select a test pair that can detect the fault. If there exists more than one vector in either the detection list or excitation list, select the test pair that detects most faults and has not been included in the test set.

Using the example shown in Table 4 again, vector $V5$ is the only test for d s@ 0. Therefore, $V5$ must be the second vector in the transition test pattern for the fault d slow-to-rise. Although each of $V1, V2, V3$ can set d to 0, we select $V1$ as the initial vector since it detects more s@ fault than the other two. Thus, the pair selected is $(V1, V5)$.

4.3 Compaction

After using any of the two heuristics we have a set of test-pairs $\{(U_1, U_2), (U_3, U_4), \ldots, (U_{n-1}, U_n)\}$. They were inserted into the test set in the order given above. We next simulate the test-pairs in the reverse order starting with $(U_{n-1}, U_n), (U_{n-3}, U_{n-2}), \ldots, (U_1, U_2)$. If a pair (U_{i-1}, U_i) does not detect any new faults then it is dropped.

5 Experimental Result

The algorithms described in Section 4 have been implemented in C. Experimental data are presented for ISCAS85 and fullscan versions of ISCAS89, on a 1.7GHz Pentium 4 PC with 512 MB of memory, running the Linux Operating System. Tables 5 and 6 report the results for the fault-list-based and priority-based heuristics, respectively. All times given are in CPU seconds. In these two tables, for each circuit, the number of s@ vectors is given first, followed by the initial number of transition patterns computed. Next, the runtime for the pre-compression phase is shown. This time includes the "dictionary computation time". The next two columns report the number of transition patterns after compression and the compression time needed. The total transition coverage and total time are reported in the final two columns. One additional column (rightmost in Table 6) shows the percentage improvement the priority-based algorithm has over the fault-list-based algorithm in terms of test data volume. For example, in Table 6, the initial STRATEGATE s@ test set has 197 vectors for fully scanned S1196 circuit, the pre-compressed transition pattern count is 532. These set of patterns were computed in 0.71 seconds from the 197 patterns using the priority-based heuristic. This initial set of patterns was compressed down to 428 in 0.69s. The total transition fault coverage was 99.97%.

For most circuits, the results by the priority-based algorithm, prior to compaction, are *already* better than those obtained by the fault-list-based followed by compaction. As

Table 6. Results for Priority-based Extension.

$Circuit$	$Stuck-at$ $test\ vectors$	$Trans$ $Patterns$	$Exten$ $time(s)$	$Comp$ $Patterns$	$Comp$ $time(s)$	$Trans$ $FC(\%)$	$Total$ $Time$	Improve (%)
c880	128	257	0.25	203	0.23	100	0.48	20.70
c1355	198	461	0.53	336	0.42	99.77	0.97	20.00
c1908	143	440	0.51	375	0.57	99.67	1.08	14.19
c3540	202	755	2.06	629	3.06	96.27	5.12	22.91
c5315	157	421	2.14	396	2.11	99.54	4.25	28.78
c6288	41	268	1.36	190	1.19	99.19	2.55	20.53
S344	31	96	0.05	84	0.04	100	0.09	17.65
S382	41	120	0.05	101	0.08	100	0.13	11.40
S526	82	209	0.09	176	0.12	99.93	0.21	21.43
S832	179	450	0.42	386	0.38	99.20	0.80	18.74
S1196	197	532	0.71	428	0.69	99.97	1.40	17.85
S1423	97	278	0.40	237	0.43	99.11	0.83	17.71
S5378	332	1069	3.40	924	4.29	98.40	7.69	16.68
S35932	78	399	12.53	347	17.15	90.50	29.68	16.79
S38417	1207	3562	58.31	3152	77.86	99.66	136.17	17.16
S38584	893	3288	76.73	2785	132.80	95.02	209.53	19.86

indicated in the right-most column of Table 6, the average improvement over all benchmark circuits was 18.90%.

The efficiency of the priority-based heuristic is competitive with the fault-list based heuristic, even though fault simulation without fault-dropping on selected faults is needed. We observed that the pre-compaction phase of the priority-based algorithm takes longer than the fault-list-based algorithm. However, since the compaction phase takes about 60% of the time, the total time required by the priority-based algorithm was often shorter due to a smaller starting test set.

6 Comparison With Commercial ATPG

We have shown so far that reusing s@ vectors to compose transition patterns is effective in composing transition patterns. An important question arises is "Can this approach be a substitute for a native-mode transition fault ATPG?" In this section we compare the priority based heuristic with a state-of-the-art commercial transition fault ATPG.

The experiment consisted of generating two test sets using the commercial tool. A compressed s@ test set, T_1, and a compressed transition test set, T_2. Then, we applied the priority-based algorithm (and compression) technique to the s@ test set T_1 to obtain another transition test set, T_3. The results are tabulated in Table 7.

In Table 7, for each circuit, the number of s@ vectors produced by the commercial ATPG is first given. Next, the number of transition patterns generated, transition fault coverage, and ATPG time are listed for the commercial ATPG and our priority-based heuristic alternately. Note that the data storage and the test application time will be proportional to twice this pattern count. The final column reports the improvement in test data volume, as well as test time, is calculated as follows:

$$Data\ Volume\ Improve = 100 * \frac{|T_2| - |T_3|}{|T_2|}$$

For three cases the commercial ATPG performs better than our heuristic. For C5315 the difference is marginal. In the other two cases, C6288 and S35932, the highly compressed s@ vectors T_1 leaves very little room for the heuristic to pick from a variety of vectors. These scenarios are due to the fact that these circuits are very deep and narrow circuits. If we consider the trend that circuits today are becoming flatter and broader to meet cycle time goals, s@ tests for newer high performance circuits will not have these characteristics. From that perspective these two examples can be ignored. In the other cases, there is a substantial improvement in the test data volume. For S9234, the compression is 50%. The average compression over all circuits, including those with negative improvement, is 19.59%.

Columns 5 and 6 give the coverage numbers. In all cases, except for C6288, our priority heuristic gives a higher fault coverage than the commercial ATPG. For C3540 the difference is about 5%. Columns 7 and 8 give the execution times for the commercial ATPG and our technique. The commercial ATPG was run on a different machine, and we cannot conclude which is faster. The data presented simply shows that the run times are comparable.

7 Additional Benefits of Reusing s@ Vectors

The approach of constructing enhanced-scan transition patterns using only s@ vectors has several additional benefits. Firstly, prior to using the test-patterns in manufacturing, the transition patterns and the vectors of which it is comprised must be validated. Since the s@ vectors have already gone through that process, the patterns in the transition tests computed by our algorithm can skip this validation process. Note, however, that transition patterns still have to be validated on silicon against yield loss. Thus, although we have not eliminated pattern validation from the flow, the overhead has been considerably reduced. From a project execution point of view this is a tremendous productivity boost.

Secondly, none of our test cases contain any contention since they are fully synthesized logic. For designs where

Table 7. Comparison with Native Transition ATPG Tool.

Circuit	T_1 (S@ Vectors)	Pattern Count T_2	Pattern Count T_3	Coverage T_2	Coverage T_3	Time T_2	Time T_3	Data Volume Improve
C1908	129	263	249	99.72	99.72	4.2	2.04	5.32
C2670	116	198	145	78.61	79.26	4.3	4.37	26.77
C3540	179	366	317	82.94	87.62	6.8	10.71	13.39
C5315	124	248	249	96.63	97.05	4.8	9.06	-0.40
C6288	36	90	111	98.95	98.54	3.6	4.28	-23.33
C7552	237	378	322	90.95	91.61	11.0	15.64	14.82
S5378	266	490	316	86.57	87.51	5.3	13.43	35.51
S9234	410	837	412	68.62	70.58	14.7	82.87	50.78
S13207	485	1002	444	80.53	82.29	27.4	70.95	55.69
S15850	464	924	522	84.98	85.76	28.0	101.30	43.51
S35932	75	137	183	90.02	90.33	94.3	61.86	-33.57
S38417	1017	1927	1027	89.92	91.19	115.6	293.30	46.71

T_1: s@ test set generated by commercial ATPG T_2: transition test set generated by commercial ATPG
T_3: transition test set generated by our priority-based algorithm using T_1

contention can occur, a tool usually extracts such contention-causing constraints which are fed to the ATPG tool. The ATPG tool then generates tests that satisfies these constraints. It is a well-known fact that ATPG tools work very hard to satisfy such constraints and that adversely affects the run time, coverage and test set size. Our approach scores over native mode transition fault ATPG in that it leverages off the "hard work" of the s@ ATPG phase. Since all the s@ vectors are contention free the transition test set our algorithm generates is also contention free.

Finally, increasing numbers of IP cores are now shipped with their test sets as part of their design data. If transition tests are added to the test data, then the test data volume problem is compounded. Since we are using the s@ test set we need to store only one copy of the vectors that are used several times. The transition test set can, then, be expressed as a sequence of pointers to these set of vectors. Thus, with a modest increase over the storage requirement over the s@ test set we can also ship the transition test set.

8 Conclusion

We presented two algorithms, Fault-List-based extension and Priority-based extension, for composing transition patterns from vectors in a s@ test set. The priority-based algorithm was shown to be superior to the fault-list based algorithm. It was demonstrated that a high quality transition pattern sets can be obtained, bypassing the need for a native mode transition fault ATPG. Experimental comparison with a native mode transition fault ATPG tool showed the proposed heuristics resulted in 20% smaller pattern set while achieving the same or higher transition fault coverage. We discussed the additional advantages of reusing the s@ vectors in pattern validation, constraint handling and reducing design data in the context of IP cores.

References

[1] R.D. Eldred "Test Routing Based on Symbolic Logical Statement" *Journal ACM, Vol.6 pp.33-36 Jan. 1959.*

[2] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar. "Transition Fault Simulation" IEEE Design & Test of Computers, 4:32-38, April 1987.

[3] G.L. Smith "Model for Delay Faults based upon Paths," *Intl Test Conf.*, pp. 342-349, Sept. 1985.

[4] K. Heragu, J. H. Patel, and V. D. Agrawal, "Segment delay faults: a new fault model," *VLSI Test Symp.,* pp. 32-39, April 1996.

[5] I. Hamzaoglu and J. H. Patel "Compact Two-pattern Test Set Generation for Combinational and Full Scan Circuits" *Intl Test Conf.*, Oct. 1998.

[6] J. Savir and S. Patil "On Broad-Side Delay Test" *VLSI Test Symp.*, pp.284-290 Sept. 1994.

[7] J. Savir and S. Patil "Scan-Based Transition Test" *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* Vol. 12, No.8 Aug. 1993.

[8] B. Dervisoglu and G. Stong " Design for Testability: Using Scanpath Techniques for Path-Delay Test and Measurement" *Intl Test Conf.*, pp.365-374, 1991.

[9] M. H. Schulz and F. Brglez "Accelerated Transition Fault Simulation," *Design Automation Conf.* pp. 237-243, June 1987.

[10] I. Pomeranz and S. M. Reddy "Static Compaction for Two-Pattern Test Sets," *Asian Test Symp.*, pp.222-228, 1995.

[11] L. N. Reddy, I. Pomeranz and S. M. Reddy "COMPACTEST-II: a method to generate compact two-pattern test sets for combinational logic circuits," *Intl. Conf. Computer-Aided Design*, pp.568-574, 1992.

[12] M. S. Hsiao E. M. Rudnick and J. H. Patel, "Sequential circuit test generation using dynamic state traversal," *European Design & Test Conf.*, pp. 22-28, 1997.

[13] F. F. Hsu, K. M. Butler and J. H. Patel, "A Case Study of the Illinois Scan Architecture," *Intl Test Conf.*, 2001, pp. 538-547.

1530-1877/02 $17.00 © 2002 **IEEE**