Techniques to Reduce Data Volume and Application Time for Transition Test

Xiao Liu, Michael Hsiao , Sreejit Chakravarty* and Paul J Thadikaran*

The Bradley Dept. of Electrical and Computer Engineering, Virginia Tech.; and * Intel Corporation[i]

ABSTRACT[1]

Scan based transition tests are added to improve the detection of speed failures using scan tests. Empirical data suggests that both data volume and application time, for transition test, will increase dramatically. Techniques to address the above problem, for a class of transition tests called "enhanced transition tests", are proposed.

The first technique, which combines the ATE repeat capability and the notion of transition test chains, reduces test data volume by 46.5%, when compared with transition tests computed by a commercial transition test ATPG tool. The test application time could increase or decrease. To address the test time issue Exchange Scan, a new DFT technique, is proposed. Exchange scan reduces both data volume and application time by 46.5%. These techniques rely on the use of hold scan cells and highlights the effectiveness of hold-scan design to address test time and test data volume issues.

1. INTRODUCTION

Higher clock rate, shrinking geometries, increasing metal density, etc. is resulting in defects that cause speed failures. The stuck-at fault model **[6]** does not model speed related failures very well. Researchers have proposed a variety of fault models for speed failures, viz.: transition fault**[7]** , path delay fault**[8]** and segment delay fault**[9]** . Of these, transition fault is the most practical and commercial tools are available for computing such tests. Scan based transition tests are added to the scan test suite to enhance the capability of scan based tests to detect speed failures. Work on generating scan based transition tests to improve the detection of speed failures in microprocessors **[17] [18] [19] [20] [21]** as well as ASICS**[16]** have been reported and/or proposed.

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. In the transition fault model, at each line in the circuit two faults 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 (*final vector*) is required to launch the appropriate transition at the target node and also propagate the fault effect to a primary output **[7] [10] [12]** .

Transition tests can be applied in three different ways: Broadside**[11]** , Skewed-Load**[12]** and Enhanced-Scan**[13]** . In broadside, 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 one vector is stored in tester scan memory. 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. One vector is stored per transition pattern in tester scan memory. In enhanced-scan transition testing two vectors (V1, V2) are stored in the tester scan memory. First V1 is loaded into the scan chain and applied to the circuit under test (CUT) to initialize it. Next V2 is scanned in, followed by applying it to the CUT and capturing the response of the CUT. During loading of V2 it is assumed that the initialization due to V1 is not destroyed. Therefore, enhanced-scan transition testing assumes hold-scan design **[13]** .

Among the three kinds of transition tests, broadside suffers from poor fault coverage **[11]** . Since there is no dependency between the two vectors in enhanced scans it, in general, gives better coverage than skewed-load transition test. Skewed-load transition tests also lead to larger test data volume. Compared to stuck-at tests, the increase in the number of vectors required for enhanced scan to get complete coverage is about 4X (Table 1). This data, collected using a commercial ATPG tool, shows the number of stuck-at vectors and the number of enhanced scan transition patterns. Each transition pattern consists of two vectors. On the other hand, data volume for skewed-load transition test for an ASIC shows an increase of 5.9X[16] . A drawback of enhanced-scan testing is that it requires hold-scan scan-cells. However, in microprocessors and other high performance circuits that use custom design, the circuit is not fully decoded. Hold scan cells are used to prevent contention during scan shift to prevent burnout. Furthermore, if hold-scan

[1] Research of the first two authors was funded by a grant from Intel Corporation

cells are used, failing parts in which only the scan logic failed can often be retrieved; thus addressing, to some extent, the yield loss associated with scan DFT. This paper considers only enhanced-scan transition tests.

No published work exists to address the explosion in data and application time for transition tests. We propose novel techniques to reduce the transition test data volume and test application time. Our optimization techniques consider optimization across test patterns for transition tests. The first technique, described in Section **3**, uses the ATE repeat option and requires the use of transition test chains, rather than transition test patterns. We describe the notion of transition test chains and present a novel algorithm for computing such chains. Results show an average data volume reduction of 46.5%, when compared with the conventional enhanced transition test data volume computed by COM, a commercial ATPG tool.

The above technique does not necessarily decrease test application time. To reduce test application time a new DFT technique called exchange-scan, described in Section **4**, is proposed. Combining exchange scan with transition test chains reduces both test application time and test data volume by 46.5%, when compared with a conventional transition test set computed by COM.

A number of ideas on reducing test data volume and test application time for single cycle scan-tests have been presented in the literature**[2] [4] [5] [3] [23] [22]** . These work assume that between 5-10% of the bits are fully specified. Unspecified bits are filled to detect the easy to detect faults. Different codes to compress the information in the tester and decompressing them on chip **[2] [5] [23]** or using partitioning and applying similar patterns to the different partitions **[3] [22]** have been proposed. The techniques proposed here complements the work on compressing individual vectors. The technique of Section **3** can be used with all the stuck-at test compression technique described in the literature, like [2] [4] [22] [23] [3] [5] , except for the technique in [2] that uses the ATE repeat option to fill unspecified bits. The technique in Section 4 does not require the use of ATE repeat and can be used in conjunction with all published techniques for compressing stuck-at vectors. It is unique in that it decreases test application time by reducing the number of scan loads. Section 5 presents a summary of all the experimental results. Data shows that the proposed technique improves both test data volume and test application time by 46.5% over COM, a commercial transition ATPG tool. Finally, Section 6 summarizes the paper.

2. ATE MODEL

Figure 1 is an abstraction of the tester model we use. ATE storage consists of scan and control memory. Scan memory is divided into several channels. Each channel consists of three bits. For each clock cycle of the scan shift operation, the scan memory contains the bit to be scanned in, the expected response bit from the circuit under test (CUT) and an indication of whether this bit of the response is to be masked or not. Figure 2(a) shows the data stored for a single scan channel for the test set {V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_9, V_{10}}, with R_j being the expected response for V_j. The control memory controls the shift and the comparison operation. The scan memory depth required is (N+1)*S bits, for a test set of size N and scan length S.

The enhanced scan transition test set of Table 2 consists of 6 patterns. Each pattern consists of a pair of vectors and the expected response to the final vector. The first pattern in our example consists of the pair (V1, V2) and the response R2 to V2. Storage of the test data in the scan memory is shown in Figure 2(b). Storage depth required is N*2*S + N*S bits. The control sequence for this test set shown in Table 3 is very repetitive and stored in the tester control memory. Row 1 states that V_1 is scanned in and applied to the CUT. Row 2 states that V_2 is scanned in, applied to the CUT and the response R_2 captured. Row 3 states that the first vector of the next vector pair, i.e. V_2, is scanned in while the response R_2 of the previous test pattern is scanned out and compared with the expected response. Once the scan operation is complete the new vector is applied to the CUT. The rest of the entries can be similarly interpreted.

3. ATE REPEAT AND TRANSITION TEST CHAINS

There is considerable redundancy in the information stored in the tester. In Figure 2(b), V_2, V_3 are used several times in the test sequence. Ideally, one copy of the information should suffice. However, storing one copy of a vector and reusing it in any random order requires the ATE to index into random locations in the scan memory, which is currently not available. Limited reuse of the information stored however is possible. In Figure 2(b), two copies of V_2 are stored in consecutive locations in the scan memory. It is possible to store just one copy of V_2 and scan in V_2 as often as possible during consecutive scan cycles. Similarly, we can replace two copies of V_3 in locations 4, 5, from the left of the scan memory in Figure 2(b), with just one copy of V_3. Further reduction of the number of copies of V_3 is not possible. Thus, we store the sequence < V_1, V_2*, V_3*, V_4, V_3, V_5, V_1, V_3> and repeatedly scan in vectors marked *. Information about vectors that needs to be scanned multiple times is stored in control memory. Thus, 8 instead of 10 vectors need to be stored.

In the above example, the scan storage requirement was reduced at a price. Since vectors that are scanned in repeatedly do not form a regular pattern the control memory requirement increases drastically. To avoid such an increase in control memory we impose a restriction that, except for the first and last vector stored in the scan memory, every vector is scanned in exactly twice. In Figure 2[c], the sequence < V_1, V_2, V_3, V_4, V_1, V_3, V_5> is stored. A vector, for example V_3, can be stored multiple times. Assume that all but the first and last vectors are scanned in twice. The set of transition test patterns applied is {(V_1, V_2,), (V_2, V_3), (V_3, V_4), (V_4, V_1), (V_1, V_3,), (V_3, V_5)}. This set includes all test patterns of Figure 2(a) as well as (V_4, V_1). Thus, by storing 7, instead of 10, vectors all transition tests can be applied. For this example, not only is control memory requirement lower but the number of vectors stored was also reduced. Sequences in which all but the first and last vectors are scanned in twice are defined to be **transition test chains**.

Computing transition test chain is different from computing a set of transition test patterns as is conventionally done. A novel ATPG algorithm, called **weighted transition graph algorithm**, to compute such chains is discussed next.

WEIGHTED TRANSITION GRAPH ALGORITHM

The algorithm constructs transition test chains from a given stuck-at test set. Instead of computing a set of vector-pairs and chaining them together, as suggested in the above example, the problem is mapped into a graph traversal problem. The algorithm uses **weighted transition-pattern graph**, a weighted directed graph. It contains a node for each vector in the stuck-at test set. A directed edge from node V_i to V_j denotes the transition test pattern (V_i, V_j) and its weight indicates the number of transition faults detected by (V_i, V_j). The **graph construction procedure** is discussed next. Assume the stuck-at test set T={T_1…T_N} is given.

1. Perform transition fault simulation using the stuck-at test set implied transition test set {(T_1, T_2), (T_2, T_3)… (T_{N-1}, T_N)} to compute *undet*, the set of undetected transition faults.
2. Deduce the subset U of stuck-at faults implied by *undet* as follows. If X *slow-to-rise* or *slow-to-fall* fault∈ *undet*, then both X stuck-at-0 and stuck-at-1 are in U.
3. Perform stuck-at fault simulation, without fault dropping, using the stuck-at test set T on the stuck-at faults in U. For each stuck-at fault f in U, record the vectors in T that excite f and the vectors that detect f. Also, for each vector, the faults excited and detected by it are recorded.
4. The weighted directed graph contains a node corresponding to each stuck-at tests in T. The directed edge, from V_i to V_j, is inserted if the corresponding test pattern (T_i, T_j) detects at least one transition fault in *undet*. The weight of (T_i,T_j) is the number of transition faults in *undet* detected by (T_i,T_j). These are deduced from the dictionaries computed in 3.

As an example, consider a circuit with 5 gates (10 stuck-at faults) and a stuck-at test set consisting of 4 vectors V1, V2, V3, and V4. The excitation and detection dictionary obtained by simulation without fault dropping are as shown in Table 5. Assuming the test set order to be [V1, V2, V3, V4], only 3 transition faults (slow-to-fall at c, e and slow-to-rise at c) are detected. Table 5 implies: (V1, V3) detects *a slow-to-fall*; (V3, V1) detects *a slow-to-rise*; (V1, V4) detects *d slow-to-fall*; (V4, V2) detects *d slow-to-rise*; (V4, V1) detects *a slow-to-rise*, *b slow-to-fall*; and (V2, V4) detects *b slow-to-rise*, *e slow-to-rise* and *d slow-to-fall*. This results in the transition-pattern graph of Figure 6(a).

Theorem 1: Faults detected by pattern (Vi, Vj) and faults detected by pattern (Vj, Vk) are mutually exclusive.

Proof: We prove this by contradiction. Without loss of generality, consider fault *f slow-to-fall* is detected by (Vi, Vj). Thus, Vi excites *f* s-a-0 (sets *f* to 1) and V_j detects *f* s-a-1. Assume (V_j,V_k) also detects *f slow-to-fall.* Then, V_j must set line *f to 1*, which is a contradiction.

An Euler trail in the transition-pattern graph traverses all edges in the graph exactly once. It is tempting to think that converting the graph to a Eulerian graph, by inserting the minimum number of edges, and computing an Euler trail would give us the minimum test. This leads to a sub-optimal solution. Traversing edge (V_i,V_j) is equivalent to selecting test (V_i,V_j). Once edge (V_i,V_j) is traversed, i.e test (V_i,V_j) is selected, it detects a number of transition faults. This alters the weights on other edges and removes some of the edges. Per Theorem 1 edges whose weights do not change are those starting from V_j. To improve the solution, edge weights are updated after traversing an edge. A preliminary version of the algorithm is outlined in Figure 4 where P is the transition test chain computed by the algorithm from the given stuck-at test set T. For the example in Figure 6(a) the updated graph, after traversing the heaviest-weight test chain (V2, V4, V3), is shown in Figure 6(b). Note that in addition to removing (V2, V4) and (V4, V3) edge (V1, V4) is also removed because the selected chain detects *d slow-to-fall*. All 7 undetected faults in Table 5 are detected with the test chain {V2, V4, V1, V3, V4, V2}.

Several optimizations were applied to the generic version of the algorithm to improve the results. Instead of considering one edge at a time we use Theorem 1 to inspect edges of length 3. This reduces the amount of simulation required and a transition test chain is generated by incrementally concatenating the vector-chains of length 3. While this solution is better than simply concatenating vector pairs for each of the remaining undetected transition faults, it still may not be optimal. When chain length increases beyond 3, the difference between the actual number of transition faults detected by the chain and the sum of edge weights in the chain determines whether it is worthwhile to use longer chains. Using longer chains reduces the number of graph updates (hence the runtime of the algorithm), but increases the size of the final solution. Experimental results in Section 5 suggest that chain lengths of 3, 4 give the best results.

Expanding on the essential test definition for stuck-at fault from [1] , we define an essential vector for transition faults to be any test vector that excites/detects at least one transition fault that is not excited/detected by any other test vector in the test set. All essential vectors must occur in the transition test chain at least once. We include essential vectors early in the chaining process. The transition test chain generation process is divided into two phases by constructing two weighted transition pattern graphs.

A. Identify all essential vectors, generate the transition-pattern graph using only essential vectors and construct the test chains only with the essential vectors in the graph. Append that to the initial transition test chain P in the generic version of the weighted transition graph algorithm.

B. Reduce the number of faults by dropping faults detected in Step A. Generate the transition-pattern graph using the remaining faults and extend the partial transition test chain from Step A as described in Step 3 of the generic weighted transition graph algorithm.

The number of edges in the second step of the modified algorithm is significantly reduced, because most of the edges incident on the essential vectors have been traversed in the previous step and thus removed.

During the test pattern generation procedure, some faults detected by the earlier test vectors may be detected by test patterns generated later. Therefore, vectors added early in our transition test set might become redundant. To identify such redundant patterns we do reverse-order pair-wise compaction. After {(U_1, U_2, U_3), (U_4, U_5, U_6), … , (U_{n-2}, U_{n-1}, U_n)} are generated, they are appended to the original stuck-at test set in that order. The test patterns (U_{n-1}, U_n), (U_{n-2}, U_{n-1}),… (U_1, U_2) are simulated in the reverse order in which it was generated. If neither (U_{i-1}, U_i), (U_i, U_{i+1}) detects any additional fault then U_i is redundant and eliminated. Note that eliminating gives rise to new transition test pairs unlike s@ tests. We therefore follow this by doing a forward-order pair-wise compaction step to further reduce the size of the test chain length.

Additional enhancements using ideas for compacting patterns for transition faults can be used. They include a dynamic compaction technique **[15]** and a static compaction technique [14] .

4. EXCHANGE SCAN

In Section 3 we saw that using transition test chains and ATE repeat can reduce data storage. Data presented in Section 5 will show the reduction to be about 46.5%. However, test application time can either decrease or increase drastically. To reduce test application time, while still retaining the improvement in data storage, a new DFT technique is proposed. To reduce the number of scan loads it does not use the ATE repeat option. A vector can be scanned in once, and using very little overhead, reused. Reducing scan in operations reduces test application time. The net result is a reduction in both the data storage requirement and the test application time.

The block diagram of a hold-scan system is shown in Figure 5. The scan cells, which consist of two parts: System Flop and Shadow MSFF, are chained together to form two related registers: SYSTEM REGISTER and SHADOW REGISTER. During normal operation, the SYSTEM REGISTER is in use. For scan testing operations SCAN_SHIFT, SCAN_LOAD and SCAN_STORE are supported. Assume the scan cell implementation of Figure 3(a). For SCAN_SHIFT, the A_CLK and B_CLK are pulsed so that data passes from SI to SOUT. For SCAN_STORE, the STORE signal is pulsed to transfer the data from SOUT to Q. The content of SHADOW REGISTER is transferred to SYSTEM REGISTER. In SCAN_LOAD the content of SYSTEM REGISTER is transferred to SHADOW REGISTER. Pulsing LOAD transfers data from Q to AOUT. Pulsing BCLK transfers data from AL to BL..

The new operation SCAN_EXCHANGE, exchanges contents of the SHADOW and SYSTEM registers, without destroying either of them. Pulsing LOAD transfers the contents of SL to AL. Pulsing STORE transfers the contents of BL to SL. Pulsing B_CLK transfers the content of AL to BL. The corresponding timing diagram is shown in Figure 3(b). No additional hardware or signal is needed to support the exchange operation. It may require the global scan controller to

be modified slightly to realize the exchange operation. The operation requires three clock cycles.

SCAN_EXCHANGE, for the transition test chain < V1, V2, V3, V4 >, is used as follows. Test-pairs applied are: {(V1, V2), (V2, V3), (V3, V4)}. The expected response on application of V2 is R2, V3 is R3 and V4 is R4. The sequence of operations, without exchange-scan, is shown in the first column of Figure 7. *Capture R2* implies that the response of the CUT on application of V2 is latched on to the SYSTEM REGISTER. *Scan In V2, Scan Out R2* implies SCAN_SHIFT wherein V2 is scanned in and the response of the CUT, from the previous pattern, is scanned out and compared to R2.

The second column of Figure 7 shows the sequence of operations using SCAN_EXCHANGE. Once V2 is loaded into the SHADOW REGISTER, the subsequent store and capture operations do not destroy the contents of the SHADOW REGISTER. So, the SCAN_LOAD operation that destroys the contents of the SHADOW REGISTER is replaced by the SCAN_EXCHANGE operation. It exchanges the content of SHADOW REGISTER and SYSTEM REGISTER. Thus the captured response is transferred to SHADOW REGISTER and V2 is applied to the CUT as the *initial vector* for the next test pair. We can therefore skip the sequence of operations that scans V2 and stores it. In addition, the SCAN_SHIFT of the response from V2, i.e. R2, can now be merged with the SCAN_SHIFT of the final vector of the next pattern V3. The net effect of this is that we have replaced an entire scan operation with a 3-cycle SCAN_EXCHANGE operation. Considering that the SCAN_SHIFT operation may be 1000 or more clock cycles this overhead of the SCAN_EXCHANGE operation is negligible and will be neglected from our calculations. Note that transition test chains, but not ATE repeat capability of the testers, is required to realize the gains of the exchange scan operation. If the ATE Repeat capability is available, each of the vectors that are stored can be compressed, as discussed in [2] , and the benefits of transition test chains can be realized using exchange scan. Data presented in Section 5 show that both test data volume and test application time decreases by 46.5%, compared to COM.

5. EXPERIMENTAL RESULTS

The weighted transition graph algorithm, with all the optimizations described above, was implemented in C. Experimental data are presented for ISCAS85 and full-scan versions of ISCAS89 benchmarks, on a 1.7Ghz Pentium 4 with 512 MB of memory, running the Linux Operating System.

In Table 6 results on fullscan version of ISCAS89 circuits, with different chain lengths, are presented. For each benchmark, the ideal (ID), given by the sum of the edge weights, and actual (AC) faults detected by the chains are shown. The difference between the ideal (ID) and actual (AC) increases with the chain length. For example, for circuit s5378, when the chain length is 2, the ideal and actual numbers of detected transition faults are the same. Likewise, when the chain length is increased to 3, they are still equal as explained by Theorem 1. When we increase the chain length beyond 3, the actual number of detected transition faults start to differ, as some of the faults detected by this last chain segment may be detected by the first 2 pairs.

Table 7 presents the results of the weighted transition-graph algorithm with and without the essential vector optimization. In Table 7, column 2 gives the number of stuck-at vectors in the original STRATEGATE test set, followed by the results for our algorithm without use of essential vectors. The final four columns show the results when essential vectors are used. For each approach, the number of transition vectors produced is shown first. Next, the number of compacted test vectors and its transition fault coverage are shown. Note that the compaction step achieves considerable reduction without losing fault coverage. The transition fault coverage achieved with or without essential vectors are the same, as indicated in column 6 and column 10 of the table; only the test set sizes are different in the two approaches. In most cases, the use of essential vectors yields smaller test sets. However, because this is a greedy heuristic, optimality is not guaranteed. The execution time with essential vectors is also generally shorter due to the quick elimination of a large number of faults detected by the essential vectors.

5.1. RESULTS FOR ATE REPEAT

In Table 8 we compare our results with results using COM, a commercial ATPG tool. We first tabulate the data for the storage required (STORAGE). Both the size of the stuck-at test set and the number of transition test vectors for COM are presented. These were generated using the dynamic compaction option of COM. Thus, for C1908 we need to store a total of 526 vectors. WT_GR used the un-compacted stuck-at test set generated using COM. The next two columns show the transition test chain lengths obtained using the proposed algorithm, with chain length 3 and 4 respectively. Thus, for C1908 we need to store 353 vectors or 346 vectors depending on the chain length used. The storage improvement obtained using transition test chains and the ATE repeat option is shown in column 11. Chain length 3 was assumed. Thus, for C1908, the storage improvement is calculated as 100* (526 – 353)/(526). Note the substantial storage

reduction obtained in all cases. The average reduction in scan memory requirement is 46.5%.

Columns 6 and 7 compare the transition fault coverage obtained by weighted transition graph (WT_GR) and COM. Note that there is no loss in fault coverage using WT_GR. Columns 8, 9 and 10 compares the CPU time required by COM and the two versions of our algorithm. For most of the circuits, COM is much faster. The last column shows changes in the test application time. Recall that for a given transition test chain all but the first and last vectors are scanned in twice. Therefore, the test application time gain for C1908 is computed as 100*(526-2*353)/(526) = -34.22. This implies a 34.22% increase in test application time if transition test chains are used. However in a number of cases the test application time actually decreases by a significant amount. The average increase in test application time is 6.9%.

5.2 Experimental results for Exchange Scan

Benefits of using exchange scan, versus COM, are tabulated in Table 4. Transition test chains were computed using our heuristic with chain length set to 3. The improvement in both test application time and test data volume are the same and shown in column 4 of Table 4. Thus, for C1908, the test application time reduction is calculated as 100*(526 – 353)/(526) = 32.89%. We note that now there is a substantial reduction in both the scan memory requirement and the test application time, vis-à-vis COM. The average reduction in test application time and data storage requirement is 46.5%.

Summary

We presented techniques to reduce test data volume and test application time for transition faults. Results from Table 4 and Table 8 are shown graphically in Figure 8. For each circuit, the data storage requirement and test application time are plotted for the conventional ATE (COM), ATE repeat, and Exchange scan. The first technique combines the ATE repeat capability and the notion of transition test chains to reduce the test data volume by 46.5%, when compared with the conventional approach. The second technique that replaces the ATE repeat option with Exchange Scan improves both test data volume and test application time by 46.5%.

REFERENCES

[1] I. Hamzaoglu and J. H. Patel “Compact Two-pattern Test Generation for Combinational and Full Scan Circuits,” *Proceedings of the International Test Conference*, 1998.

[2] B. Keller, C. Barnhart, V. Brunkhorst, F. Distler, A. Ferko, O. Farnsworth and B. Koenemann, “OPMISR: The Foundation of Compressed ATPG Vectors,” *IEEE International Test Conference*, pp. 748-757, 2001.

[3] I. Hamzaoglu and J. H. Patel, “Reducing test application time for full scan embedded cores,” *29th International Symposium on Fault-Tolerant Computing*, June 1999, pp. 260-267.

[4] B. Koeneman, “LFSR-Coded Test Patterns for Scan Designs,” IEEE European Test Conference, 1991, pp. 237-242.

[5] A. Chandra and K. Chakrabarty, “Frequency directed run length (FDR) codes with application to system on a chip data compression,” *IEEE VLSI Test Symposium*, 2001, pp. 42-47.

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

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

[8] G.L. Smith ``Model for Delay Faults based upon Paths,” IEEE *International Test Conference*, pp. 342-349, 1985.

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

[10] M. L.Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*, Kluwer Academic Publishers, Boston, 2000.

[11] J. Savir and S. Patil ``On Broad-Side Delay Test," *IEEE VLSI Test Symposium*, pp.284-290, 1994.

[12] J. Savir and S. Patil ``Scan-Based Transition Test,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No.8, 1993.

[13] B. Dervisoglu and G. Stong `` Design for Testability: Using Scanpath Techniques for Path-Delay Test and Measurement',' *Proceedings of International Test Conference*, pp.365-374, 1991.

[14] I. Pomeranz and S. M. Reddy ``Static Compaction for Two-Pattern Test Sets,” *IEEE Asian Test Symposium*, pp.222-228, 1995.

[15] L. N. Reddy, I. Pomeranz and S. M. Reddy ``COMPACTEST-II: a method to generate compact two-pattern test sets for combinational logic circuits," *IEEE/ACM International Conference on Computer-Aided Design*, pp.568-574, 1992.

[16] F. F. Hsu, K. M. Butler and J. H. Patel, ``A Case Study of the Illinois Scan Architecture,” *IEEE International Test Conference*, 2001, pp. 538-547.

[17] N. N. Tendulkar, R. F. Molyneaux, C. Pyron and R. Raina, “At-speed Scan Based Testing on MPC7400 Microprocessor,” *IEEE VLSI Test Symposium*, pp. 3-8, 2000.

[18] K. McCauley, “Scan is Good Enough for Stuck Faults, Why Not AC Scan for Delay Faults?” Panel position paper, *IEEE International Test Conference*, 2001, pp. 1172.

[19] R. Raina, “AC-Scan: Microprocessors are ready But where is the Infrastructure?” Panel position paper, *IEEE International Test Conference*, 2001, pp. 1173.

[20] S. Patil, “AC-Scan: Microprocessors are ready But where is the Infrastructure?” Panel position paper, *IEEE International Test Conference*, 2001, pp. 1174.

[21] N. Tendulkar, R. Raina, R. Woltenburg, X. Lin, B. Swanson and G. Aldrich, ``Novel Techniques for Achieving High At-Speed Transition Fault Coverage for Motorola's Microprocessors Based on PowerPC Instruction Set Architecture,” *IEEE VLSI Test Symposium*, 2002, to appear.

[22] K-J. Lee, J-J. Chen and C-H Huang, “Using a single input to support multiple scan chains,” *IEEE/ACM International Conference on Computer-Aided Design*, 1998, pp. 74-78.

[23] D. Das and N. A. Touba, “Reducing test data volume using external/LBIST hybrid test patterns,” *IEEE International Test Conference*, 2000, pp. 115-122.

Table 1

Name	s@ Test	Trans Test	Expansion
C1908	129	263	4.07
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	4.13
S15850	464	924	3.98
S35932	75	137	3.65
S38417	1017	1927	3.79
S38584	727	1361	3.74

Table 2

Vector1	Vector2	Response
V1	V2	R2
V2	V3	R3
V3	V4	R4
V3	V5	R5
V1	V3	R3
V4	V3	R3

Table 3

(i) Shift In V1**; (ii) Apply;
(i) Shift In V2**; (ii) Apply; (iii) Capture;
(i) Shift In V2, Shift Out and Compare R2; (ii) Apply;
(i) Shift In V3; (ii) Apply; (iii) Capture;
(i) Shift In V3, (ii) Shift Out and Compare R3; (iii) Apply;
(i) Shift In V4; (ii) Apply; (iii) Capture;
(i) Shift In V3, Shift Out and Compare R4; (ii) Apply;
(i) Shift In V5; (ii) Apply; (iii) Capture;
(i) Shift In V1, Shift Out and Compare R5; (ii)Apply;
(i) Shift In V3; (ii) Apply; (iii)Capture;
(i) Shift In V4, Shift Out and Compare R3; (ii) Apply;
(i) Shift In V3; (ii) Apply; (iii) Capture;
(i) Shift Out and Compare R3;

Table 4

CIRCUIT	STORAGE		IMPROVEMENT
	COM	WT_GR	DATA, APP TIME
C1908	526	353	32.89
C2670	396	185	53.28
C3540	732	410	43.99
C5315	496	310	37.50
C6288	180	130	27.77
C7552	756	428	43.38
S5378	980	455	53.57
S9234	1674	644	61.53
S13207	2004	681	66.02
S15850	1848	729	60.55
S35932	274	224	18.29
S38417	3854	1555	59.65

Table 5

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

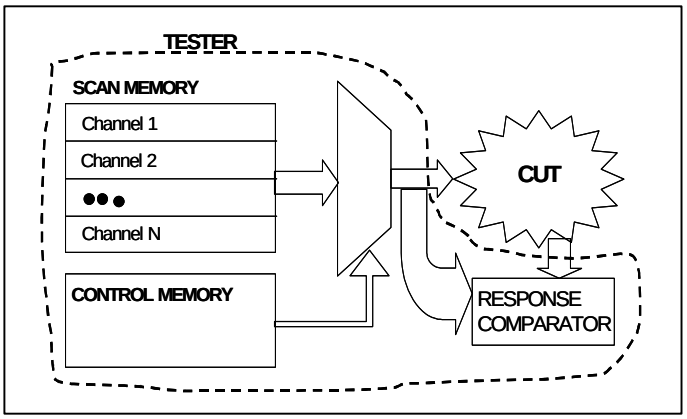

Figure 1. Tester Memory Model

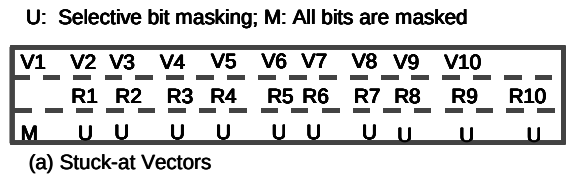

Figure 2. ATE Storage Model

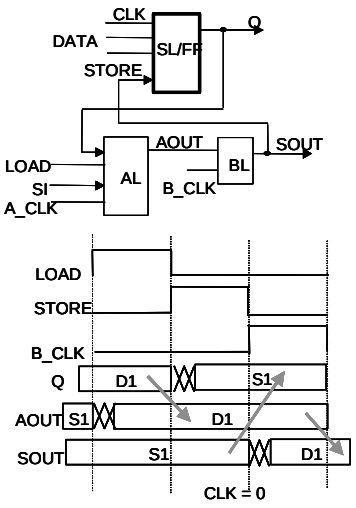

Figure 3. Hold Scan cell and Exchange scan timing

GENERIC WEIGHTED TRANSITON GRAPH ALGORITHM

Compute the transition-pattern graph;
Initialize P to T = {$T_1 \ldots T_N$};
WHILE ((transition fault coverage < 100%) &&
(iteration number < MAX))
BEGIN
Identify an edge (V_i, V_j) with the largest weight;
Append vectors V_i, V_j to P;
For all edges starting from V_j
BEGIN
Look for edge (V_j, V_k) having the largest weight;
Append vectors V_j and V_k to the original stuck-at test set;
END
Update the weighted transition graph and the faultlist;
END

Figure 4. Generic Weighted Transition Graph Algorithm

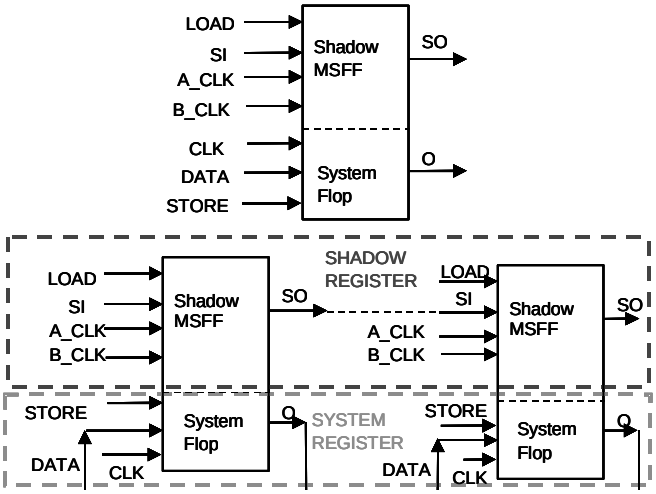

Figure 5. Block diagram of Hold-Scan System

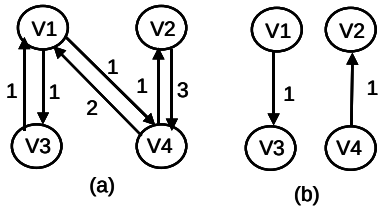

Figure 6. Example Weighted Transition-Pattern Graph

WITHOUT EXCHANGE	WITH EXCHANGE
Scan In V1	Scan In V1
Store V1	Store V1
Scan In V2	Scan In V2
Store V2	Store V2
Capture R2	Capture R2
Load R2	Exchange (R2, V2)
Scan In V2, Scan Out R2	Scan In V3, Scan Out R2
Store V2	Store V3
Scan In V3	Capture R3
Store V3	Exchange (R3, V3)
Capture R3	Scan In V4, Scan Out R3
Load R3	Store V4
Scan In V3, Scan Out R3	Capture R4
Store V3	Exchange (R4, V4)
Scan In V4	Scan In V5, Scan Out R4
Store V4	
Capture R4	
Scan In V4, Scan Out R4	

Figure 7

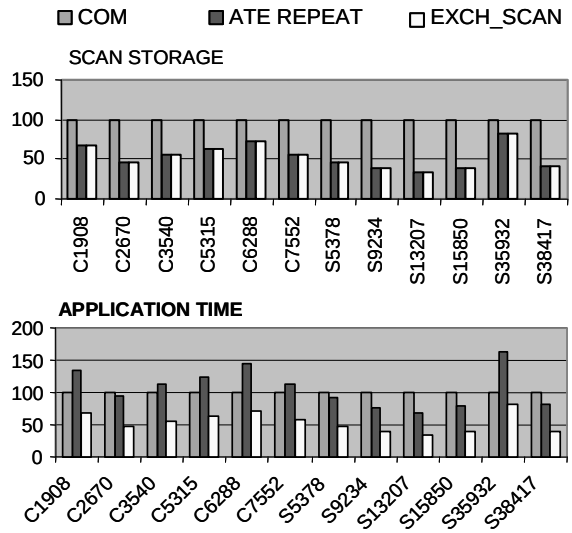

Figure 8

Table 6

CIRCUIT	2		3		4		5		6		7	
	ID	AC	ID	AC	ID	AC	ID	AC	ID	AC	ID	AC
S344	38	38	64	64	90	72	111	89	143	89	164	103
S382	24	24	43	43	63	44	75	56	92	61	111	61
S832	60	60	87	87	114	89	136	95	158	95	180	103
S1196	47	47	78	78	116	93	146	105	177	105	208	119
S1423	36	36	60	60	79	79	106	95	129	118	153	124
S5378	59	59	103	103	149	142	190	144	235	172	284	176
S35932	1072	1072	1175	1175	1270	1247	1382	1261	1475	1312	1564	1334
S38417	132	132	190	190	274	249	380	285	439	296	500	319

Table 7

CIRCUIT	S@ Set	Without essential vectors				With essential vectors			
		Tran. Tests	Comp. Tests	Time (s)	TFC (%)	Tran. Tests	Comp. Tests	Time (s)	TFC (%)
C1355	198	928	285	3.51	99.77	915	270	2.82	99.77
C1908	143	918	318	4.57	99.67	966	298	3.32	99.67
C3540	202	1222	515	25.43	96.27	1181	514	22.06	96.27
C5315	157	816	342	11.80	99.54	762	313	9.79	99.54
C6288	141	310	122	5.60	99.19	334	120	5.70	99.19
S344	31	135	63	0.37	100	207	64	0.37	100
S832	179	988	310	4.36	99.20	937	292	2.78	99.20
S1196	197	1004	362	5.24	99.97	1022	358	4.38	99.97
S1423	97	566	186	2.25	99.11	528	177	2.09	99.11
S5378	332	1672	722	35.73	98.40	1685	722	29.76	98.40
S35932	78	542	196	133.13	90.50	633	197	133.01	90.50
S38417	1207	5142	2682	1073.03	99.66	5208	2686	858.85	99.66

Table 8

CIRCUIT	STORAGE				TF COV		CPU TIME			IMPROVEMENTS	
	COM		WT_GR		COM	WT_GR	COM	WT_GR		STORAGE	APP TIME
	s@	TRAN	3	4				3	4		
C1908	177	526	353	346	99.7	99.72	4.2	3.33	3.47	32.89	-34.22
C2670	167	396	185	184	78.6	79.26	4.3	9.25	9.99	53.28	6.57
C3540	247	732	410	419	82.9	87.62	6.8	19.45	20.11	43.99	-12.02
C5315	213	496	310	323	96.6	97.05	4.8	10.20	10.77	37.50	-25.00
C6288	47	180	130	118	99	98.54	3.6	3.72	3.62	27.77	-44.44
C7552	348	756	428	431	91	91.61	11	28.34	28.21	43.38	-13.23
S5378	391	980	455	464	86.6	87.51	5.3	37.66	37.76	53.57	7.14
S9234	630	1674	644	646	68.6	70.58	14.7	319.18	324.20	61.53	23.06
S13207	662	2004	681	679	80.5	82.29	27.4	292.89	295.30	66.02	32.06
S15850	641	1848	729	749	85	85.76	28	350.00	358.58	60.55	21.10
S35932	81	274	224	209	90	90.33	94.3	87.56	86.30	18.29	-63.50
S38417	1449	3854	1555	1547	89.9	91.19	116	1416.82	1406.83	59.65	19.30