Journal of Science and Technology, Vol. 29, No. 3 (2009), pp 126-135 126
© 2009 Kwame Nkrumah University of Science and Technology (KNUST)

STATIC TEST COMPACTION AS A MINIMUM COVERING PROBLEM

K. O.Boateng
Department of Computer Engineering,
Kwame Nkrumah University of Science and technology,
Kumasi, Ghana

ABSTRACT
Large numbers of test stimuli impact on the time and cost of test application. Hence there is the need to keep numbers of test stimuli low while maintaining as high fault coverage as possible. In this paper, static compaction of test stimuli is seen as a minimization problem. The task of static compaction of a set of test stimuli has been formulated as a minimum covering problem. Based on the concept of minimization, a method of static compaction has been developed. Results of experiments conducted to evaluate the method are also presented. The method achieved a significant compaction of sets of test stimuli that had previously been compacted by means of a test generation algorithm that features dynamic compaction.

Keywords: *test compaction, test stimuli, minimum covering problem, essential test selection, redundant test elimination*

INTRODUCTION

Test is indispensable in the production and maintenance of reliable integrated circuits (IC). Testing is achieved by the application of a set of test stimuli. For a digital circuit, test stimulus may be a single input pattern or a sequence of input patterns depending on the type of circuit and/or kind of fault. In combinational circuits, for example, a test stimulus for a delay fault consists of a pair of input patterns. In this paper, a general test stimulus (a single pattern or a sequence of patterns forming a single unit of a test set) is referred to as a test matrix.

Memory capacity is one of the main factors that determine the cost of test equipment. Large memory means high cost. More test patterns than can be contained in the memory of the available test equipment require memory update (reloading of a subset of the test patterns) during test application. Even a single memory update leads to a big leap in test application time (TAT) that affects the time-to-market of the manufactured IC products. It is evident, from the foregoing, that small sets of test matrices are desirable. However, making test sequences shorter should not reduce the fault coverage. This means that test compaction algorithms should find ways of achieving small test sets without diminishing fault coverage.

Two types of compaction techniques exist, namely dynamic and static compaction. Dynamic techniques attempt to reduce the number

of test matrices while they are being generated. Often dynamic compaction requires the modification of the test generator. Static techniques, on the other hand, seek to reduce the number of the already generated test matrices. Thus, static compaction is a post-processing step to test generation. Static techniques are therefore independent of the test generation algorithms and do not require any modifications of the algorithms. Moreover, even if dynamic compaction is used during test generation, static compaction can further reduce the size of the generated test set. This suggests that static compaction is more effective in reducing the sizes of test sets and hence the subsequent cost of testing.

Hsiao and Chakradhar (1998a, 1998b) and Hsiao, *et al.,* (1997) have reported works on methods of test pattern compaction for sequential circuits. Pomeranz and Reddy (1998) proposed a method of efficient storage of test responses but his method does not involve compaction of test stimuli. Kajihara and Saluja (1998) reported on their work on test pattern compaction (for combinational circuits) based on random pattern fault simulation. Also, Hamzaoglu and Patel (1997), have reported their work on a static compaction algorithm and the computation of minimum sets of test patterns for combinational circuits. Each of the two methods of static compaction are tailor-made to work together with a particular existing test generation algorithm.

Since a static compaction is effective by itself and can also act as a supplement to dynamic compaction, there is the need to develop a versatile test-generation-independent static compaction algorithm for a general digital circuit (which may be combinational or sequential).

By formulating static compaction of test matrices as a minimum covering problem (Boateng *et al.,* 2001), the proposed method performs static compaction of test matrices generated by any generation algorithm. The method finds the minimum subset of the original test set that can detect all the faults detectable in the circuit under test (CUT) by the original test set without modifying any test matrix. Characteristics of the proposed method include the following:

- The method does not assume any knowledge of the test generation algorithm.
- The method finds a minimum-sized subset of a given test set that covers all the faults detectable by the original set without modifying any of the test matrices.
- The method is applicable to both combinational and sequential circuits.
- The method applies a method of simulation (Boateng *et al.,*1998*,* 1999), that facilitates the extraction of faults (Boateng *et al.,* 1998, 1999), (Yanagida *et al.,* 1995) covered by the original test set without conducting repetitive fault simulation.
- The method uses a data structure that ensures efficient run-time data storage.

Even though only the stuck-at fault model has been used in some examples in this paper, the proposed method can be applied to other fault models.

Table 1: The *cover table* for T with respect to circuit C

Fault / Test	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
t_1	x	x		x			x					
t_2	x		x	x				x				
t_3		x			x		x			x		
t_4			x			x						
t_5					x				x			
t_6				x			x	x			x	
t_7						x	x					x

The rest of the paper is organized as follows. Section 2 presents the basic concept of the minimum covering problem as applied to the problem of the compaction of test matrices. The compaction strategy is presented in Section 3, the formal statement of the compaction method in Section 4, and experimental results in Section 5. Section 6 concludes the paper.

THE BASIC CONCEPT

Definition 1: The set of all faults, F_{set}, detectable in the circuit, C, by a set, T_{set}, of test matrices forms the cover of T_{set}.

Definition 2: Let the fault f in circuit C be detectable by a test matrix t_{ep} of a test set T_{set}. If no other test matrix of T_{set} detects f, then t_{ep} is an *essential matrix* of T_{set}, and the fault f is a *critical fault* of C with respect to T_{set}.

Consider that the set $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ of test matrices covers all the stuck-at faults on lines $L_1, L_2, L_3, L_4, L_5, L_6$ which form part of a circuit, C. Let the corresponding fault set be $F = \{f_1, f_2, f_3, f_4, \ldots, f_{11}, f_{12}\}$. In Table 1, a × in the intersection of row t_i and column f_j indicates that matrix t_i detects fault f_j under the single fault assumption. From the table, it can be seen that $t_3; t_5; t_6; t_7$ are essential matrices of T corresponding to the critical faults $f_9; f_{10}; f_{11}; f_{12}$. When $t_3; t_5; t_6; t_7$ are removed from T and all the faults they cover are removed from the original cover F, we have the situation depicted in Table 2. None of the faults (f_1 and f_3) remaining in F is critical with respect to the updated test set $T = \{t_1, t_2, t_4\}$.

When the remaining test matrices t_1, t_2, t_4 (in T) are checked (in that order) for their coverage of the remaining faults, t_4 is found to be redundant. When the identified redundant test matrix t_4 is discarded, f_3 becomes a critical fault with respect to $T = \{t_1, t_2\}$ (the corresponding essential matrix being t_2). Selecting the essential matrix t_2 from T and removing the faults detectable by it (t_2) from the set $F = \{f_1, f_3\}$, leave no more faults (undetected) in the original fault set F. So, the remaining test matrix t_1 in T is also discarded. Thus the compacted test set, $CT = \{t_2, t_3, t_5, t_6, t_7\}$, is now without the matrices t_1, t_4.

Table 2: The *cover table* after first iteration

Fault Test	f_1	f_3
t_1	x	
t_2	x	x
t_4		x

The Static Compaction Strategy

Redundant Test Matrix Removal

The strategy is to effect static compaction by a few repetitions of the selection of essential matrices and discarding of identified redundant matrices.

Lemma 1: With reference to a given set of faults in a circuit C, any test set T without essential matrices has at least one test matrix t that is redundant with respect to the set $T = T\backslash\{t\}$.

Proof: Suppose that at a cycle of the repetition, the test matrices $rt_1, rt_2, \ldots, rt_i, \ldots, rt_N$ are remaining. Assume there are no essential matrices in the remaining subset of test matrices. Thus, any fault in the cover of the subset of the remaining test matrices that is detectable by the matrix rt_j is also detectable by another matrix rt_k (where $k \neq j$) in the same subset. Thus, at least, rt_j is redundant. Q.E.D.

Lemma 1 shows that in the process of repeated application of essential matrix removal and the discarding of redundant matrices, a deadlock situation where there are neither essential matrices nor redundant matrices, does not occur. This means algorithm of repeated selection of essential matrices and elimination of redundant matrices will always converge.

Processing the remaining data as explained in the last paragraph of Section 2 may lead to a non-minimal solution. For example, if t_2 happened to detect only f_3 and t_4 happened to detect f_1 in addition to f_3, then t_1 and t_2 would be selected leaving t_4 as redundant. Evidently, this way of dominance checking is not effective. In view of this, the following iterative processing

is performed to ensure an effective dominance checking that leads to a minimal solution to the static compaction problem. The process of redundant fault elimination is carried out by identifying the best set of test matrices to retain for the next cycle of the selection of essential test matrices. After selecting the current set of essential matrices, dropping the faults they cover from the fault set and updating F, a heuristic, *weight*, is assigned to each of the remaining n test matrices. The test matrix with the greatest *weight* is separated as rt_1 (with cover C_1). *weight* is re-calculated for the remaining n-1 test matrices as follows.

$weight_k = |S_k|$, where $S_k = F_k \setminus \mathcal{F}$ and $\mathcal{F} = C_1$, where F_k is the subset of faults in F that the *k-th* remaining test matrix detects.

The next greatest *weight* determines rt_2 (with its cover C_2) to be separated. Next the *weights* are re-evaluated with $\mathcal{F} = C_1 \cup C_2$. Thus, generally, for the *i-th* iteration,

$$\mathcal{F}' = \bigcup_{j=1}^{i-1} C_j$$

This is repeated until $\mathcal{F} = F$ after separating rt_l. At this stage, the remaining (n-l) test matrices are identified as redundant and hence are discarded. Another cycle of the compaction algorithm then begins with the selection of the next hierarchy of essential test matrices. This continues until no more redundant test matrices are identified. Using the heuristic ensures effective algorithmic dominance checking.

Logic optimization is often expressed as a minimum-covering problem to enhance algorithmic solution. A direct application of the concept of minimum-covering problem to static compaction of test stimuli for practical circuits will require a large amount of memory, which may not be available. Thus this situation will impose a limitation on the applicability of the method to practical circuits. To go around this problem, the two-dimensional data space is reduced to the more manageable one-dimensional data space. This is achieved as follows.

First, each fault has a detection counter associated with it. Any time the fault is detected by a test matrix the counter associated with the fault is incremented. Also, a mapping of test matrices to faults is established. All faults with counter = 1 are critical faults and the respective faults that map onto them constitute the essential matrices. An essential test matrix identified at the *i*-th selection of essential matrices is called an *i*-nary essential matrix.

Simulation

To establish the matrix-to-fault mapping, simulations are performed. If the faults are injected one after another and all test matrices are used to simulate each injected fault in order to determine which matrix detects which fault, then the process will consume much time. To cut down on simulation time the following fault-tracing approach (similar to the back-tracing process in (Boateng *et al.,* 1998, 1999), (Yanagida *et al.,* 1995)) is used.

1. Test matrices are applied one-by-one to CUT.
2. At the application of each test matrix, the rules for fault tracing are used to trace the faults detectable under single fault assumption

Rules for fault tracing

Definition 3: The logic value c of multiple-input gate G which when applied to any of the inputs of G will force the output of G to a given logic level cv, is called the controlling value of G. cv is the controlled value of G. The complements of c and cv are called the non-controlling respective non-controlled logic values of G.

Table 3 gives the controlling and controlled logic values of some common gates. For the exclusive-OR gate, both logic values 1 and 0 are non-controlling logic values (and hence non -controlled logic values). After the test matrix has been applied to the primary inputs and values have been propagated to the primary outputs of CUT, fault tracing proceeds backward from the observed primary outputs to the primary inputs. This fault tracing is carried out

under the single fault assumption. Different tracing criteria are used under different fault models. Under the stuck-at fault model, the fault detectable at an on-path line L with logic value v is L-stuck-at-$\bar{v}$

Table 3: Controlling and controlled logic values

Gate	Logic Value	
	Controlling	*Controlled*
OR	**1**	**1**
NOR	**1**	**0**
AND	**0**	**0**
NAND	**0**	**1**

For gate G whose output fault has been currently traced, the following are the rules used in deciding the inputs (of G) along which fault tracing continues.

Rule 1: For a one-input gate G, fault tracing along current path continues along the input.

Rule 2: For a multiple-input gate G with a non-controlled output value, fault tracing along current path continues along all inputs that are on sensitized paths.

Rule 3: For a multiple-input gate G with a controlled output value, a controlling value at only one input i and non-controlling value at all other inputs, fault tracing along current path continues along i.

Rule 4: For a multiple-input gate G with a controlled output value and a controlling value at multiple inputs *i* that are on sensitized paths, if there exist a re-convergence between a stem s and G, and if the logic values at all i are simultaneously controllable from s, then fault tracing along current path skips the re-convergent loop and continues along s. Otherwise fault tracing along current path terminates at G.

Rule 5: Fault tracing continues on a fan-out stem if it is identified as s in Rule 4, or if the fault at one or more of its branches has been currently traced.

Figures 1, 2 and 3 illustrate Rules 2, 3 and 4 respectively. Let us take the stuck-at fault for an example. In Figure 1, the single fault whose effect is observable/deductible at the output of G may be a stuck-at-0 fault located at the output itself or at any one of the inputs (under the stuck-at fault model, they are all on sensitized paths).

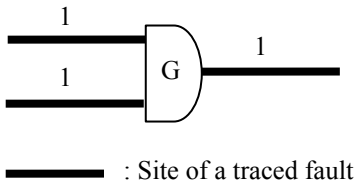

Fig. 1: An illustration of Rule 2

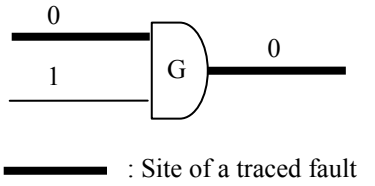

Fig. 2: An illustration of Rule 3

In other words, a single stuck-at-0 fault located at the output or any one of the inputs is detectable by the input vector < 1; 1 >. In the case of Figure 2, apart from stuck-at-1 fault at the output itself, the only other possible single fault on a sensitized path is a stuck-at-1 fault located at the single input with a controlling logic value of G. In Figure 3, if the fault whose effect is observable/deductible at the output of G4 is not located at the very output, then the effect must be on the two inputs of G4 simultaneously. The single fault whose effect can be propagated to the two inputs of G4, must be located at either the output of G1 (Rule 4: the stem of a re-convergent loop) or the single input of G1 having the controlling value of G1 (by subsequent application of Rule 3).

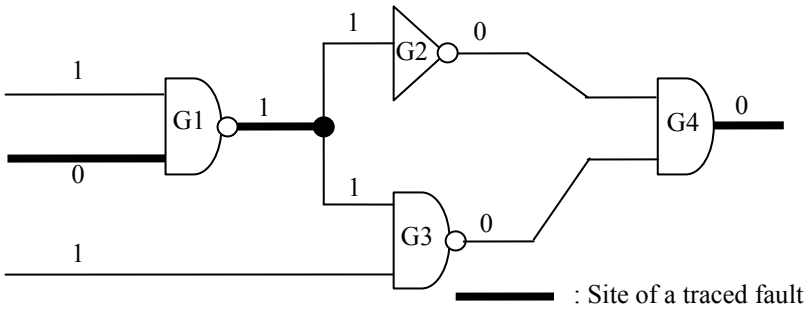

Fig. 3: An illustration of Rule 4

When tracing faults detectable by a multi-vector test matrix, the rules of this subsection are applied as follows.

1. In the case of a combinational circuit, the last (propagation) vector is used.
2. In the case of a sequential circuit, use is made of the time-frame expansion model as an equivalent circuit from the point of view of testing. This way the combinational replica of the circuit corresponding to each time frame has only one vector of the test matrix applied to it.

THE METHOD

This method has two phases. In the first phase, simulations are performed and fault tracing is carried out to establish the matrix-to-fault mapping. Initially each detection counter is set to 0. During fault tracing, detection counter incrementing is performed. In the second phase, the process of essential matrix identification and selection, and the process of identifying and discarding redundant matrices are performed.

Phase One

Step 1: Perform simulations to: (1) establish the matrix-to-fault mapping, and (2) increment detection counters as the faults to which they are associated are traced.

Step 2: Put all the test matrices in set *T*_set, and all the traced faults in *F*_set.

Step 3: Set each of the sets CT_set, T1_set, and F1_set to the empty set { }.

Phase Two

Compaction is carried out in this phase as follows.

Step 1: Find all current critical faults and identify their corresponding essential test matrices. Drop all faults detectable by the identified essential test matrices from *F*_set. Transfer the identified essential test matrices from T_set into CT_set.

Step 2: If *F*_set is empty, discard any test matrices in T set and end compaction.

Step 3: Find in *T*_set the test matrix, *rt*, with the greatest *weight*. Put *rt* into T1_set. Transfer all faults in *F*_set that are covered by *rt* into F1 set.

Step 4: If *F*_set is not empty, go to **Step 3**.

Step 5: For each test matrix in *T*_set, decrement the detection counter of each fault (in *F1*_set) it detects. Discard all the remaining test matrices in *T*_set. Empty the sets *T1*_set and *F1*_set into the sets *T*_set respective *F*_set, and go to **Step 1**.

At the end of compaction, the compacted test set is the set CT_set. Figure 4 shows the algorithmic flow of the method of static compaction of test matrices.

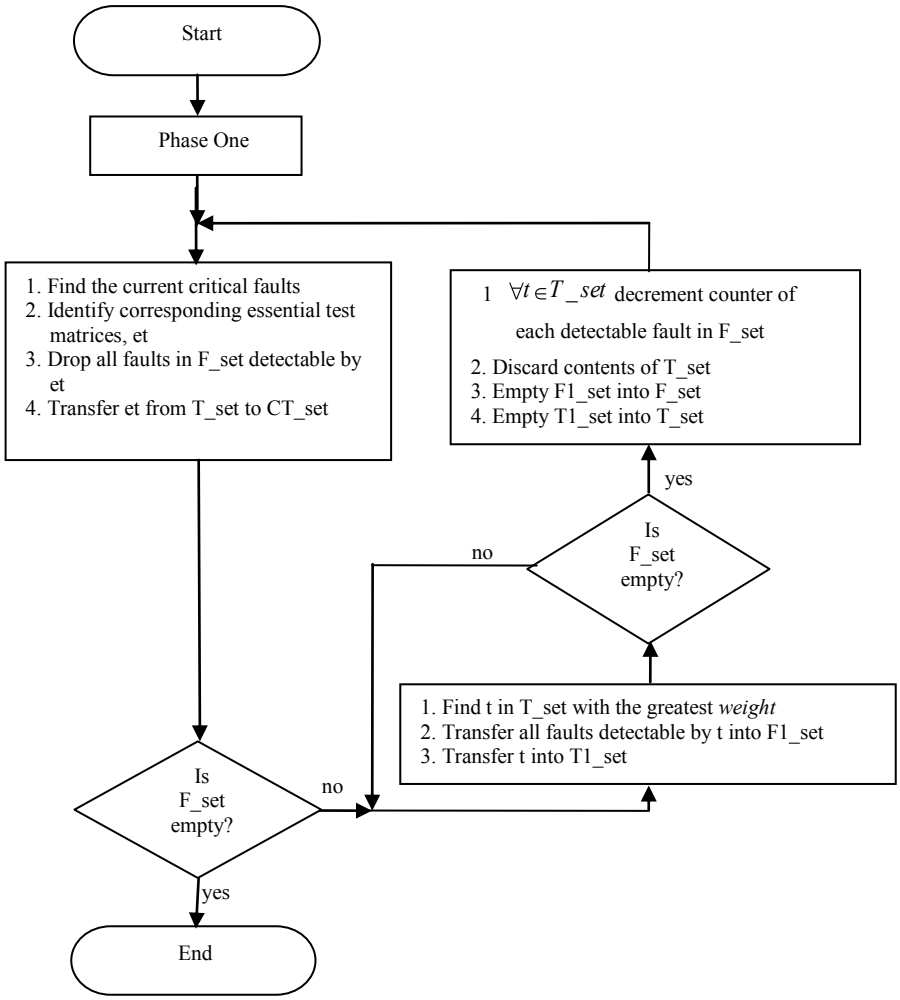

Fig. 4: Flow of the method of compaction

An Illustration

After Phase One, the proposed method will organize the data in Table 1 as shown in Figure 5. T_set = $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$; F_set = $\{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}\}$; *CT*_set = T1_set = F1_set = { }. Note that counter=1 is the criterion for the identification of critical faults and hence essential matrices. In Figure 5, arrows from the primary essential test matrices are bold.

After selecting the primary essential matrices, CT_set = $\{t_3, t_5, t_6, t_7\}$; T_set = $\{t_1, t_2, t_4\}$; *F*_set = $\{f_1, f_3\}$. The updated and re-ordered mapping is shown in Figure 6a. $weight_1$ (for t_1) is 1, $weight_2$ (for t_2) is 2 and $weight_3$ (for t_4) = 1, thus *T*_set = $\{t_1, t_4\}$; $rt_1=t_2$; T1_set = $\{t_2\}$; F1_set = $\{f_1, f_3\}$; *F*_set = { }. Since F_set is empty the counters of f_1, and f_3 are each decremented once (see Fig.6b), as they are detected by test matrices t_1 and t_4 respectively, and t_1 and t_4 are discarded.

Next *F*_set and *T*_set are updated with the contents of F1_set and T1_set thus F_set = $\{f_1, f_3\}$ and *T*_set = $\{t_2\}$. From Figure 6b the counters of f_1, and f_3 are each equal to one, thus they are both secondary critical faults. This implies that

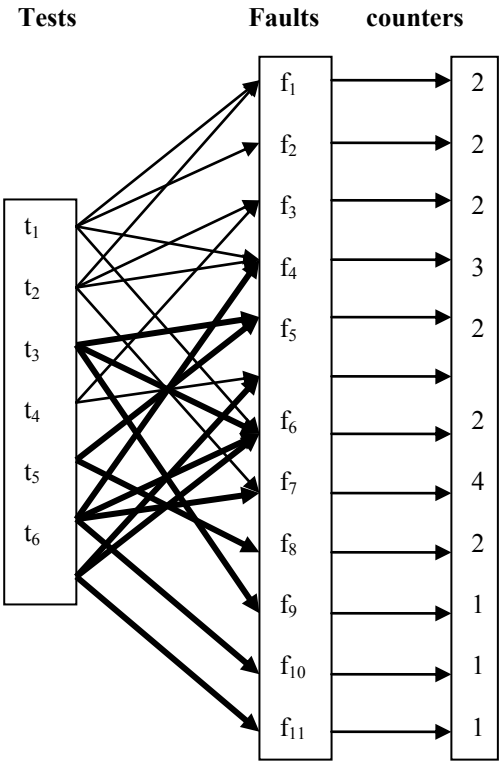

Fig. 5: Initial matrix-to-fault mapping

t_2, which detects them, is a secondary essential test matrix and hence it is appended to CT_set and the two faults (secondary critical faults) it detects dropped. Now F_set is empty and the test compaction algorithm terminates with CT_set = $\{t_2, t_3, t_5, t_6, t_7\}$.

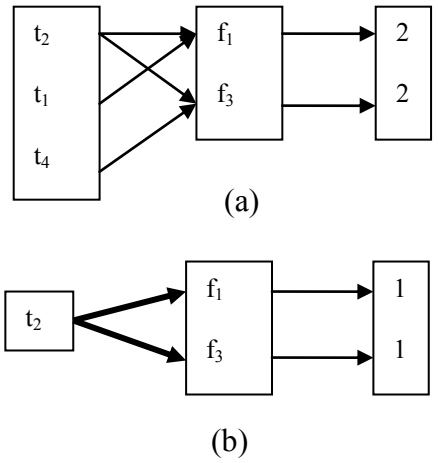

Fig. 6: Reduced matrix-to-fault mappings

EXPERIMENTAL RESULTS

Table 4: Results for some ISCAS benchmark circuits

Circuit	*n(T)*	*n(T')*	*r*	*%r*	*cpu*
c432	39	37	2	5.13	0.05
c499	88	56	32	36.36	0.20
c880	43	35	8	18.60	0.07
c1355	155	115	40	25.81	0.65
c1908	176	131	45	25.57	0.77
c2670	89	82	7	7.87	0.57
c3540	172	133	39	22.67	1.02
c5315	111	95	16	14.41	1.22
c6288	65	32	33	50.77	1.87
c7552	140	120	20	14.29	2.37
cs386	77	68	9	11.69	0.05
cs510	67	58	9	13.43	0.07
cs526	69	59	10	14.49	0.07
cs820	115	104	11	9.57	0.13
cs832	115	105	10	8.70	0.12
cs838	87	83	4	4.60	0.18
cs953	105	30	75	71.43	0.32
cs1196	170	116	54	31.76	0.37
cs1238	176	113	63	35.80	0.47
cs5378	144	129	4	3.01	1.70
Average			**24.55**	**21.17**	

The proposed method has been developed and implemented in the C language in an experimental setup. Experiments have been conducted on a SunOS 5.6 workstation. First, test sets targeting stuck-at faults were generated for some ISCAS benchmark combinational and full -scan sequential circuits ((Brglez and Fujiwara 1985), (Brglez, Bryan and Kozminski 1989)). The test generation algorithm used incorporates a dynamic compaction feature which was active during the generation of the test sets. Experiments were then conducted on the generated test sets.

Results obtained for the benchmark circuits are shown in Table 4. In the table, $n(T)$ and $n(T')$ signify the number of test matrices in the sets T and T', respectively. T is the original test set, T' is the final compacted test set. r is the reduction ($n(T)$ - $n(T')$) in the number of test matrices as a result of compaction. %r is r expressed as a percentage of $n(T)$. *cpu* is the CPU time (in seconds) consumed by the compaction process. Table 4 shows that, on the average, the proposed method achieved more than a further 21% reduction in the number of test matrices in a dynamically compacted test set.

Table 5 shows data on sets of primary essential matrices that were identified in the course of test compaction. $n(T'')$ is the number of test matrices in the set T'' of primary essential matrices. %m is percentage of the number of test matrices in the final compacted test set that are primary essential matrices. %f is the percentage of the number of faults in the cover of the final compacted test set (and hence that of the original test set) that are covered by primary essential matrices.

CONCLUSION

In this method static compaction of test stimuli has been structured as a minimum covering problem. A proposed method based on this minimization concept has been developed. The method is general, easy to implement, and does not assume any knowledge of the test generation algorithm. It employs simulation and fault tracing procedures that render repeated and time-consuming fault simulation unnecessary.

Table 5: Data on first set of essential test matrices (patterns) selected

Circuit	*n(T)*	*n(T″)*	*n(T′)*	*%m*	*%f*	*% cpu*
c432	39	37	37	100.00	100.00	100.00
c499	88	45	56	80.36	95.96	90.00
c880	43	33	35	94.29	99.55	100.00
c1355	155	113	115	98.26	98.89	96.92
c1908	176	127	131	96.95	99.24	97.40
c2670	89	80	82	97.56	99.95	100.00
c3540	172	123	133	92.48	99.02	90.20
c5315	111	90	95	94.74	99.67	92.62
c6288	65	16	32	50.00	97.33	65.78
c7552	140	118	120	98.33	99.85	98.31
cs386	77	67	68	98.53	99.87	100.00
cs510	67	55	58	94.83	98.82	100.00
cs526	69	57	59	96.61	99.43	100.00
cs820	115	103	104	99.04	99.94	100.00
cs832	115	104	105	99.05	99.94	100.00
cs838	87	82	83	98.80	99.94	100.00
cs953	105	17	30	56.67	93.37	78.13
cs1196	170	108	116	93.10	99.08	89.19
cs1238	176	95	113	84.07	96.78	70.21
cs5378	144	128	129	99.22	99.99	97.06
Average				**91.14**	**99.99**	**93.29**

A data structure that ensures efficient run-time storage is used. Experimental results show that, for the circuit used, an average further reduction of the size of a dynamically compacted test set by over 21% is achievable by the proposed method.

REFERENCES

Boateng, K. O., Konishi, H. and Nakata, T. (2001). "A Method of Static Compaction of Test Stimuli", Proc. IEEE 10th Asian Test Symposium, pp.137-142, November, 2001.

Boateng, K. O., Takahashi, H. and Takamatsu, Y. (1998). "Multiple gate delay fault diagnosis using test-pairs for marginal delays," IEICE Transaction on Information and Systems, Vol.E81-D, No.7, pp.706-715, July 1998.

Boateng, K. O., Takahashi, H. and Takamatsu, Y. (1999) "Diagnosing delay faults in combinational circuits under the ambiguous delay model," IEICE Transaction on Information and Systems, Vol.E82-D, No.12, pp. 1563-1571, Dec. 1999.

Brglez, F., Bryan, D. and Kozminski, K. (1989). "Combinational Profiles of Sequential Benchmark Circuits," Proc. of the Int. Symp. on Circuits and Systems, pp.1929-1934, May 1989.

Brglez, F. and Fujiwara, H. (1985). "A Neutral Netlist of 10 Combinational Benchmark Designs and a Special Translator in Fortran," Proc. of the Int. Symp. on Circuits and Systems, June 1985.

Hamzaoglu, I. and Patel, J. H. (1997). "Test set compaction algorithms for combinational circuits," Proc. of ACM International Conference on CAD, pp.283-289, 1997.

Hsiao, M. S. and Chakradhar, S. T. (1998a), "Partitioning and reordering techniques for static test compaction of sequential circuits," *Proc. of the 7th IEEE Asian Test Symposium*, pp.452-457, 1998.

Hsiao, M. S. and Chakradhar, S. T. (1998b),. "State relaxation based subsequence removal for fast static compaction in sequential circuits," *Proc. of Design Automation and Test in Europe Conf.,* pp.577-582, 1998.

Hsiao, M. S., Rudnick, E. M. and Patel, J. H. (1997). "Fast algorithms for static compaction of sequential circuit test vectors," Proc. of IEEE VLSI Test Symposium, pp.188-195, 1997.

Kajihara, S. and Saluja, K. (1998). "On test pattern compaction using random pattern fault simulation," Proc. of IEEE International Conference on VLSI Design, pp.464 -469, 1998.

Pomeranz, I. and Reddy, S. M. (1998). "On test pattern compaction objectives for combinational and sequential circuits," Proc. of IEEE International Conference on VLSI Design, pp.279-284, 1998.

Yanagida, N., Takahashi, H. and Takamatsu, Y. (1995). "Multiple fault diagnosis by sensitizing input pairs," IEEE Design & Test of Computers, Vol.12, No.3, pp.44-52, Sept. 1995.