PRESERVE: Static Test Compaction that Preserves Individual Numbers of Tests

Irith Pomeranz

***Abstract*— A comprehensive test set targets faults of different types to ensure that defects of different types are detected. When test compaction is carried out for such a test set, it is advantageous if the compacted test set contains a compact test set for each fault type separately. In this case, if one (or more) of the fault types is found to be more important to detect, a compact test set for it can be extracted without further processing. This paper describes the first test compaction procedure for transition and stuck-at faults, where by construction, the compact test set contains compact test sets for each fault type separately. Experimental results for benchmark circuits demonstrate the ability to compact a comprehensive test set under this condition.**

***Index Terms*— Broadside tests, skewed-load tests, stuck-at faults, test compaction, transition faults.**

I. Introduction

Different types of faults are targeted during test generation to derive a comprehensive test set capable of detecting different types of detects [1]-[7]. Faults can be static (such as stuck-at faults or single-pattern cell-aware faults [8]) or dynamic (such as transition faults or two-pattern cell-aware faults). Among static and dynamic cell-aware faults there are faults that model different types of defects, and can also be considered as different types of faults. Different types of faults can be considered together by a test generation procedure that is designed for this purpose [1]. Another option is to use a generalized fault model to represent different types of faults uniformly [2], [3], [6]. In the more commonly used approach, faults of different types are targeted consecutively, topping off a test set for one type of faults with tests for undetected faults of another type [4], [5].

For simplicity of discussion, this paper considers the scenario where a test set is generated for stuck-at and transition faults. However, the discussion is applicable whenever several different types of faults are considered, including cell-aware faults that are partitioned based on the types of defects they model. Considering transition and stuck-at faults, and the scenario where top-off tests are used, a test set for transition faults is likely to be topped off with tests for stuck-at faults. The reason is that broadside (launch-off-capture) and skewed-load (launch-off-shift) tests for transition faults [9]-[10] already detect most of the stuck-at faults. Therefore, only small numbers of tests for undetected stuck-at faults need to be added.

In the approach described in [7], single-cycle tests for stuck-at faults are transformed into skewed-load tests to ensure that tests for stuck-at faults also detect transition faults (this is not the case for single-cycle tests). The transformation allows higher levels of test compaction to be achieved when the set of target faults consists of both transition and stuck-at faults. In general, higher levels of test compaction are achieved if a single test set is computed where every test can detect target faults of all the fault types. The transformation of single-cycle tests into skewed-load tests as in [7] does not require any fault simulation, and it is guaranteed that a skewed-load test will detect the same stuck-at faults as the single-cycle test from which it is derived.

Suppose that a test set T_i is generated for transition and stuck-at faults (the discussion applies to any other combination of fault types). It is important for the discussion in this paper to partition T_i into a subset $T_{tr_i} \subseteq T_i$ of tests needed only for transition faults, a subset $T_{sa_i} \subseteq T_i$ of tests needed only for stuck-at faults, and a subset $T_{com_i} \subseteq T_i$ of tests needed for both types of faults. Such a partition is not unique, and it will be obtained by construction in the procedure described in this paper. The partition is illustrated by Figure 1(a). With this partition, the subset $T_{tr_i} \cup T_{com_i}$ is a transition fault test set, and the subset $T_{sa_i} \cup T_{com_i}$ is a stuck-at test set. As discussed later, it is advantageous if $T_{tr_i} \cup T_{com_i}$ is a compact test set for transition faults, and $T_{sa_i} \cup T_{com_i}$ is a compact test set for stuck-at faults.

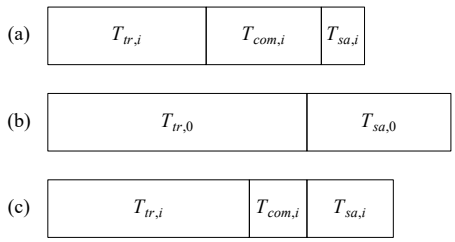

Fig. 1. Test set for transition and stuck-at faults

When T_i is compacted for the two fault types together, as in [7], a set covering procedure [11] is needed for finding transition and stuck-at test sets from the tests in T_i. From these test sets it is possible to find T_{tr_i}, T_{sa_i} and T_{com_i}. The transition and stuck-at test sets $T_{tr_i} \cup T_{com_i}$ and $T_{sa_i} \cup T_{com_i}$ may not be as compact as individual test sets that are computed for transition and stuck-at faults directly. This is illustrated by Figure 1(b) that shows compact test sets $T_{tr,0}$ and $T_{sa,0}$ for transition and stuck-at faults, respectively. In general, such test sets do not share any tests, and $T_{com,0} = \emptyset$. Although $|T_{tr,0}| + |T_{sa,0}| > |T_i|$, we have that $|T_{tr,0}| \leq |T_{tr,i}| + |T_{com,i}|$ and $|T_{sa,0}| \leq |T_{sa,i}| + |T_{com,i}|$. The increase in the number of tests in the individual transition and stuck-at test sets contained in T_i supports the reduction in the total number of tests of T_i.

When T_i is generated by topping off a transition fault test set with stuck-at tests, the transition fault test set $T_{tr,i} \cup T_{com,i}$ consisting of the first tests in T_i is compact. For stuck-at faults it is necessary to apply a set covering procedure to find the test set contained in T_i [11]. In this case, the stuck-at test set $T_{sa,i} \cup T_{com,i}$ may be larger than a test set computed directly for stuck-at faults.

In the general context where a single test set T_i targets several fault types to achieve test compaction as in Figure 1(a), the test sets for the individual fault types included in T_i are important for the following reason.

Depending on the types of defects present in faulty units, one (or more) of the fault types considered may be more effective than the others in modeling defects. Thus, a test set for one (or more) fault types may be more important to apply. Especially if T_i is a large test set, it may need to be truncated to address constraints related to the tester memory or test application time. In this case, it is important to be able to identify a smaller test set for the relevant fault types. In a specific scenario described in [12], a test set is generated with requirements for >99% stuck-at fault coverage, >95% transition fault coverage, some coverage of cell-aware and small delay faults, and an upper bound on the number of tests. Because of overheads related to initialization of the chip and translation of the test data, there is an overflew in test data volume. The test engineer calculates that 5% of the tests need to be removed. This needs to be done without compromising on the hard fault coverage requirements.

Based on this discussion, the procedure described in this paper produces a compact test set T_i for transition and stuck-at faults, but ensures that the subset of tests included in T_i for each one of the

Irith Pomeranz is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, U.S.A. (e-mail: pomeranz@ecn.purdue.edu).

This work was supported in part by NSF grant CCF-1714147

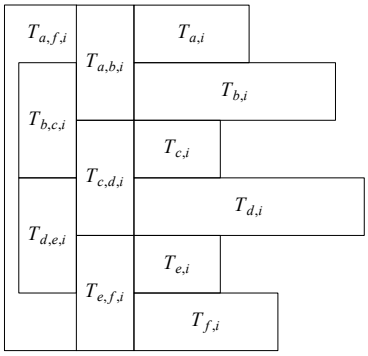

Fig. 2. Test set for six fault models

fault types alone is not larger than a compact test set for this fault type. Referring to Figure 1, the procedure produces T_i such that $|T_{tr,i}| + |T_{com,i}| = |T_{tr,0}|$ and $|T_{sa,i}| + |T_{com,i}| = |T_{sa,0}|$ while T_i is as small as possible. This is illustrated by Figure 1(c). In the scenario from [12], T_i allows one to avoid omitting tests that will decrease the stuck-at or transition fault coverage.

In the procedure described in this paper, the partition of T_i into $T_{tr,i}$, $T_{sa,i}$ and $T_{com,i}$ is obtained by construction. Thus, it is not required to use a set covering procedure for obtaining individual test sets for transition and stuck-at faults. The procedure starts with $T_0 = T_{tr,0} \cup T_{sa,0}$ and $T_{com,0} = \emptyset$. For $i \geq i$, it adds a test to $T_{com,i}$ only if it can remove one test from $T_{tr,i}$ and one test from $T_{sa,i}$. In this way, it preserves the numbers of tests in the transition and stuck-at test sets $T_{tr_i} \cup T_{com_i}$ and $T_{sa_i} \cup T_{com_i}$ included in T_i, while reducing the number of tests in T_i.

This is the first paper to describe a test compaction procedure for several fault types that preserves the numbers of tests in compact test sets for the individual fault types, and it achieves this goal by construction.

To extend the discussion to more than two fault types, it is important to note that the computational complexity of computing common tests increases as the number of fault types increases. In addition, fewer common tests are likely to exist. Instead, the procedure described in this paper can be applied without modification to pairs of fault types. This is illustrated in Figure 2, where six fault types denoted by a, b, ..., f are considered. Fault types a and b are considered to produce the set of common tests denoted by $T_{a,b,i}$ in Figure 2. Fault types c and d, and fault types e and f, are considered in a similar way. Next, common tests are found out of $T_{b,i}$ and $T_{c,i}$ to form the set denoted by $T_{b,c,i}$. In a similar way, $T_{d,e,i}$ and $T_{a,f,i}$ are formed. Every time a test is added to a common set, the total number of tests is reduced by one. After considering each fault type twice, additional common tests are less likely to exist.

The paper is organized as follows. Section II provides an overview of the test compaction procedure, and discusses how the procedure ensures that, as T_i is compacted, it continues to contain compact test sets for transition and stuck-at faults individually. The procedure described in Section III updates $T_0 = T_{tr,0} \cup T_{sa,0}$ when $T_{sa,0}$ detects transition faults that are not detected by $T_{tr,0}$, or $T_{tr,0}$ detects stuck-at faults that are not detected by $T_{sa,0}$. An iteration of test compaction is described in Sections IV and V. The need for an iterative procedure is discussed in Section VI. Experimental results for benchmark circuits are presented in Section VII.

II. Compact Individual Test Sets

The test compaction procedure described in this paper accepts compact test sets $T_{tr,0}$ and $T_{sa,0}$ for transition and stuck-at faults, respectively, as shown in Figure 1(b). Several options are possible for the test types in $T_{tr,0}$ and $T_{sa,0}$. In general, the test compaction procedure is developed under the assumption that every test type in the test sets it considers is suitable for every fault type it considers. The specific scenario considered in this paper is the following.

The transition fault test set $T_{tr,0}$ consists of both broadside and skewed-load tests. This is important for achieving the highest possible transition fault coverage. It is also important for test compaction. The stuck-at test set $T_{sa,0}$ is a single-cycle test set. Every single-cycle test is transformed into a skewed-load test as described in [7]. Initially, $T_0 = T_{tr,0} \cup T_{sa,0}$, and $T_{com,0} = \emptyset$.

Starting from T_0 the procedure produces compact test sets T_1, T_2, ... For $i \geq 1$, the procedure maintains the partition of T_i into $T_{tr,i}$, $T_{sa,i}$ and $T_{com,i}$ as shown in Figure 1(c). This is important for the following reason.

Let the set of transition faults be F_{tr}, and let the set of stuck-at faults be F_{sa}. To determine the fault coverage of T_i, F_{tr} is simulated only under $T_{tr,i} \cup T_{com,i}$, and F_{sa} is simulated only under $T_{sa,i} \cup T_{com,i}$. This implies that the procedure maintains $T_{tr,i} \cup T_{com,i}$ as a test set for transition faults, and $T_{sa,i} \cup T_{com,i}$ as a test set for stuck-at faults. The procedure ensures that the numbers of tests in these subsets do not increase for $i \geq 1$. It thus ensures that T_i continues to contain compact test sets for transition and stuck-at faults individually as it is compacted.

The procedure first updates T_0 by checking whether any of the tests in $T_{sa,0}$ detects transition faults that are not detected by $T_{tr,0}$. This is based on the experimental observation that, after a single-cycle test for stuck-at faults is transformed into a skewed-load test, it sometimes detects transition faults that are not detected by $T_{tr,0}$. This occurs even when a commercial tool generates broadside and skewed-load tests for transition faults. When this occurs, the skewed-load test is moved from $T_{sa,0}$ to $T_{com,0}$. This implies that the test is also used for detecting transition faults. For completeness, the procedure also simulates stuck-at faults under the tests in $T_{tr,0}$. If a stuck-at fault is detected, the procedure moves the test from $T_{tr,0}$ to $T_{com,0}$.

Moving tests from $T_{sa,0}$ or $T_{tr,0}$ to $T_{com,0}$ increases the number of tests in the test sets $T_{tr,0} \cup T_{com,0}$ and $T_{sa,0} \cup T_{com,0}$ for detecting transition and stuck-at faults, respectively. This is accepted to support the increase in the fault coverage when $T_{tr,0} \cup T_{com,0}$ is used as a transition fault test set, or $T_{sa,0} \cup T_{com,0}$ is used as a stuck-at test set. This is the only step where the numbers of tests in the individual test sets may be increased.

Next, in an iterative process, for $i \geq 1$, the procedure uses the test set T_{i-1} obtained in iteration $i-1$ to produce a test set T_i as shown in Figure 1(c). Initially, $T_i = T_{i-1}$. This implies that $T_{tr,i} = T_{tr,i-1}$, $T_{sa,i} = T_{sa,i-1}$, and $T_{com,i} = T_{com,i-1}$. The procedure repeats a process where it attempts to move a test t_j from either $T_{tr,i}$ or $T_{sa,i}$ to $T_{com,i}$. This is acceptable when the inclusion of t_j in $T_{com,i}$ allows the procedure to remove a test from $T_{sa,i}$ or $T_{tr,i}$, respectively. Effectively, adding t_j to $T_{com,i}$ removes two tests, one from $T_{tr,i}$ and one from $T_{sa,i}$, thus contributing to the compaction of T_i. With this process, the set $T_{com,i}$ is such that $|T_{tr,i}| + |T_{com,i}| = |T_{tr,i-1}| + |T_{com,i-1}|$, and $|T_{sa,i}| + |T_{com,i}| = |T_{sa,i-1}| + |T_{com,i-1}|$. Thus, test compaction is achieved without increasing the numbers of tests in the individual transition and stuck-at test sets.

The procedure has the option of modifying t_j to ensure that it can replace tests from $T_{tr,i}$ and $T_{sa,i}$. An iteration of the procedure without test modification is described in Section IV. An iteration of the procedure with test modification is described in Section V.

III. Additional Fault Coverage

This section describes the use of tests from $T_{sa,0}$ and $T_{tr,0}$ for the detection of transition and stuck-at faults, respectively.

Initially, $T_0 = T_{tr,0} \cup T_{sa,0}$, and $T_{com,0} = \emptyset$. Fault simulation with fault dropping is carried out for F_{tr} under $T_{tr,0}$, and F_{sa} under $T_{sa,0}$, to remove detected faults from consideration.

Next, for every test $t_j \in T_{sa,0}$, the procedure performs fault simulation with fault dropping of F_{tr} under t_j. If any faults are detected, the procedure moves t_j from $T_{sa,0}$ to $T_{com,0}$. With $T_{com,0} \neq \emptyset$, the transition fault test set contained in T_0 is $T_{tr,0} \cup T_{com,0}$, and it detects transition faults that are not detected by $T_{tr,0}$.

For completeness, the procedure also checks in a similar way whether tests for transition faults detect additional stuck-at faults. The procedure is summarized next as Procedure 0.

Procedure 0: Additional fault coverage

1) Assign $T_0 = T_{tr,0} \cup T_{sa,0}$.
2) Perform fault dropping fault simulation of F_{tr} under $T_{tr,0}$.
3) Perform fault dropping fault simulation of F_{sa} under $T_{sa,0}$.
4) For every test $t_j \in T_{sa,0}$:
   a) Perform fault dropping fault simulation of F_{tr} under t_j.
   b) If any faults are detected, move t_j to $T_{com,0}$.
5) For every test $t_j \in T_{tr,0}$:
   a) Perform fault dropping fault simulation of F_{sa} under t_j.
   b) If any faults are detected, move t_j to $T_{com,0}$.

Since no new tests are added to T_0, its number of tests does not change. However, it is possible to obtain $|T_{tr,0}| + |T_{com,0}| > |T_{tr,0}|$ or $|T_{sa,0}| + |T_{com,0}| > |T_{sa,0}|$ to support the increase in the fault coverage. The test compaction procedure considers $|T_{tr,0}| + |T_{com,0}|$ and $|T_{sa,0}| + |T_{com,0}|$ as the numbers of tests to preserve in compact test sets for transition and stuck-at faults individually.

IV. Test Compaction without Test Modification

This section describes iteration $i \geq 1$ of the test compaction procedure without test modification. The procedure reduces the number of tests in T_i relative to the number of tests in T_{i-1} by repeating a step where it moves a test from $T_{tr,i}$ or $T_{sa,i}$ to $T_{com,i}$ unmodified, and removes a test from $T_{sa,i}$ or $T_{tr,i}$, respectively.

Initially, $T_i = T_{i-1}$. This implies that $T_{tr,i} = T_{tr,i-1}$, $T_{sa,i} = T_{sa,i-1}$, and $T_{com,i} = T_{com,i-1}$.

In a preprocessing step to test compaction, the procedure performs fault simulation with fault dropping of F_{tr} under $T_{com,i}$ followed by $T_{tr,i}$. The subset of faults that a test $t_j \in T_{tr,i}$ detects is denoted by $D_{tr,i}(t_j)$. In a similar way, the procedure performs fault simulation with fault dropping of F_{sa} under $T_{com,i}$ followed by $T_{sa,i}$. The subset of faults that a test $t_j \in T_{sa,i}$ detects is denoted by $D_{sa,i}(t_j)$.

The procedure considers every test $t_j \in T_{tr,i} \cup T_{sa,i}$, each one alone. The tests are considered in a random order since it is unknown in advance which tests will be effective for test compaction.

When the procedure considers a test $t_j \in T_{tr,i}$, it attempts to find a test $t_k \in T_{sa,i}$ such that t_j detects all the stuck-at faults detected by t_k. In this case, the procedure moves t_j from $T_{tr,i}$ to $T_{com,i}$, and removes t_k from $T_{sa,i}$. Overall, the number of tests in T_i decreases by one (one test is moved, and one test is removed). The numbers of tests in $T_{tr,i} \cup T_{com.i}$ and $T_{sa,i} \cup T_{com.i}$ remain the same.

To check whether $t_j \in T_{tr,i}$ detects all the stuck-at faults detected by any test $t_k \in T_{sa,i}$, the procedure considers the set F_{sa} of all the stuck-at faults. It performs fault simulation of F_{sa} under t_j. The subset of detected faults is denoted by $E_{sa,i}(t_j)$.

The procedure compares $E_{sa,i}(t_j)$ with $D_{sa,i}(t_k)$ for every test $t_k \in T_{sa,i}$. If it finds a test t_k such that $D_{sa,i}(t_k) \subseteq E_{sa,i}(t_j)$, it uses t_j and t_k to reduce the number of tests in T_i. Specifically, it moves t_j from $T_{tr,i}$ to $T_{com,i}$, and removes t_k from $T_{sa,i}$. In this case it does not compare t_j with any other tests from $T_{sa,i}$. In a similar way, the procedure considers every test $t_j \in T_{sa,i}$.

It is important to note that the subsets $D_{sa,i}(t_k)$ and $D_{tr,i}(t_k)$ are obtained by fault simulation with fault dropping. Even when the test sets are compact, small subsets are obtained for tests at the end of the test set. This allows tests to be moved to $T_{com,i}$.

The procedure is summarized next as Procedure 1. Procedure 1 associates a flag denoted by $try(t_j)$ with every test $t_j \in T_{tr,i} \cup T_{sa,i}$. The flag $try(t_j)$ indicates whether t_j still needs to be considered for test compaction. It is initially one, and changed to zero after t_j has been considered.

Procedure 1: Iteration of test compaction without test modification

1) Assign $T_i = T_{i-1}$.
2) Perform fault simulation with fault dropping of F_{tr} under $T_{com,i}$ followed by $T_{tr,i}$. For every test $t_j \in T_{tr,i}$, find the subset of faults $D_{tr,i}(t_j)$ that the test detects.
3) Perform fault simulation with fault dropping of F_{sa} under $T_{com,i}$ followed by $T_{sa,i}$. For every test $t_j \in T_{sa,i}$, find the subset of faults $D_{sa,i}(t_j)$ that the test detects.
4) For every test $t_j \in T_{tr,i} \cup T_{sa,i}$ assign $try(t_j) = 1$.
5) If $try(t_j) = 0$ for every test $t_j \in T_{tr,i} \cup T_{sa,i}$, stop.
6) Select a test $t_j \in T_{tr,i} \cup T_{sa,i}$ with $try(t_j) = 1$. Assign $try(t_j) = 0$.
7) If $t_j \in T_{tr,i}$:
   a) Perform fault simulation of F_{sa} under t_j, and find the subset of detected faults $E_{sa,i}(t_j)$.
   b) Assign $found = 0$. For every test $t_k \in T_{sa,i}$, as long as $found = 0$, if $D_{sa,i}(t_k) \subseteq E_{sa,i}(t_j)$:
      i) Move t_j from $T_{tr,i}$ to $T_{com,i}$.
      ii) Remove t_k from $T_{sa,i}$.
      iii) Assign $found = 1$.
8) If $t_j \in T_{sa,i}$:
   a) Perform fault simulation of F_{tr} under t_j, and find the subset of detected faults $E_{tr,i}(t_j)$.
   b) Assign $found = 0$. For every test $t_k \in T_{tr,i}$, as long as $found = 0$, if $D_{tr,i}(t_k) \subseteq E_{tr,i}(t_j)$:
      i) Move t_j from $T_{sa,i}$ to $T_{com,i}$.
      ii) Remove t_k from $T_{tr,i}$.
      iii) Assign $found = 1$.

After applying Procedure 1, the number of tests in T_i is decreased by $|T_{com,i}| - |T_{com,i-1}|$ relative to the number of tests in T_{i-1}. We also have that $|T_{tr,i}| + |T_{com,i}| = |T_{tr,i-1}| + |T_{com,i-1}| = ... = |T_{tr,0}| + |T_{com,0}|$ and $|T_{sa,i}| + |T_{com,i}| = |T_{sa,i-1}| + |T_{com,i-1}| = ... = |T_{sa,0}| + |T_{com,0}|$.

V. Test Compaction with Test Modification

An iteration of test compaction with test modification is described in this section.

An iteration i with test modification proceeds as in Section IV to consider every test $t_j \in T_{tr,i} \cup T_{sa,i}$. Suppose that $t_j \in T_{tr,i}$. Without test modification, after computing $E_{sa,i}(t_j)$, the procedure searches for a test $t_k \in T_{sa,i}$ such that $D_{sa,i}(t_k) \subseteq E_{sa,i}(t_j)$. If this condition is satisfied, t_j and t_k are used for compacting T_i. When test modification is an option, it is applied to t_j and t_k if $|D_{sa,i}(t_k) \setminus E_{sa,i}(t_j)| \leq \Delta$, for a constant Δ. Thus, t_j detects all but Δ faults that t_k detects.

The goal of the test modification procedure is to obtain a test $t_{j,k}$ that detects all the faults in $D_{tr,i}(t_j) \cup D_{sa,i}(t_k)$. A smaller value of Δ makes it more likely that $t_{j,k}$ can be found. Test generation targeting the faults in $D_{tr,i}(t_j) \cup D_{sa,i}(t_k)$ can be used for this purpose. The implementation used in this paper starts from $t_{j,k} = t_j$. It modifies $t_{j,k}$ by complementing its bits one by one. A complementation is accepted if it does not cause $t_{j,k}$ to lose the detection of

TABLE I
EXPERIMENTAL RESULTS

			tests				tot/		com				f.c.			tot/max0
circuit	proc	iter	tr	sa	com	tot	tot0	max0	tr0	sa0	tr1	sa1	tr	sa	ntime	[7]
aes_core	0	0	273	208	0	481	1.000	1.762	0	0	0	0	99.992	100.000	3.00	
aes_core	1	1	265	200	8	473	0.983	1.733	1	7	0	0	99.992	100.000	27.41	
aes_core	2	1	262	197	11	470	0.977	1.722	1	7	0	3	99.992	100.000	1952.10	
aes_core	2	2	261	196	12	469	0.975	1.718	1	7	0	4	99.992	100.000	3611.61	0.972
s35932	0	0	30	20	0	50	1.000	1.667	0	0	0	0	89.781	89.809	3.83	
s35932	1	1	28	18	2	48	0.960	1.600	1	1	0	0	89.781	89.809	10.13	
s35932	2	1	27	17	3	47	0.940	1.567	1	1	1	0	89.781	89.809	293.50	1.552
systemcaes	0	0	158	121	0	279	1.000	1.766	0	0	0	0	99.841	99.995	3.03	
systemcaes	1	8	118	81	40	239	0.857	1.513	30	10	0	0	99.841	99.995	144.35	
systemcaes	2	1	105	68	53	226	0.810	1.430	30	10	7	6	99.841	99.995	2295.13	
systemcaes	2	8	81	44	77	202	0.724	1.278	32	10	22	13	99.841	99.995	9879.02	0.987
s38584	0	0	393	146	0	539	1.000	1.372	0	0	0	0	93.785	95.852	4.34	
s38584	1	2	353	106	40	499	0.926	1.270	10	30	0	0	93.785	95.852	70.13	
s38584	2	1	329	82	64	475	0.881	1.209	10	30	2	22	93.785	95.852	5654.68	
s38584	2	6	310	63	83	456	0.846	1.160	10	34	6	33	93.785	95.852	17593.30	0.930
s5378	0	0	215	122	0	337	1.000	1.567	0	0	0	0	97.866	99.131	3.34	
s5378	1	2	181	88	34	303	0.899	1.409	9	25	0	0	97.866	99.131	61.89	
s5378	2	1	161	68	54	283	0.840	1.316	9	25	1	19	97.866	99.131	3246.17	
s5378	2	9	127	34	88	249	0.739	1.158	17	25	15	31	97.866	99.131	14658.28	0.949
wb_dma	0	0	138	64	2	204	1.000	1.457	0	0	0	0	99.692	100.000	3.12	
wb_dma	1	3	106	32	34	172	0.843	1.229	18	14	0	0	99.692	100.000	81.39	
wb_dma	2	1	98	24	42	164	0.804	1.171	18	14	2	6	99.692	100.000	2652.69	
wb_dma	2	3	96	22	44	162	0.794	1.157	18	14	2	8	99.692	100.000	6611.54	0.912
b20	0	0	243	196	188	627	0.951	1.455	0	0	0	0	95.446	95.744	5.07	
b20	1	7	137	90	294	521	0.791	1.209	76	30	0	0	95.446	95.744	45.13	
b20	2	1	110	63	321	494	0.750	1.146	76	30	7	20	95.454	95.744	405.48	
b20	2	5	87	40	344	471	0.715	1.093	83	32	14	27	95.454	95.744	997.96	1.729
b14	0	0	235	235	113	583	0.975	1.675	0	0	0	0	94.482	95.532	4.74	
b14	1	15	36	36	312	384	0.642	1.103	177	22	0	0	94.482	95.532	63.14	
b14	2	1	33	33	315	381	0.637	1.095	177	22	0	3	94.488	95.532	105.78	
b14	2	2	32	32	316	380	0.635	1.092	178	22	0	3	94.488	95.532	139.44	1.668
tv80	0	0	647	459	41	1147	0.996	1.667	0	0	0	0	98.080	99.591	3.96	
tv80	1	8	286	98	402	786	0.682	1.142	299	62	0	0	98.080	99.591	161.76	
tv80	2	1	259	71	429	759	0.659	1.103	299	62	16	11	98.080	99.591	1269.03	
tv80	2	5	231	43	457	731	0.635	1.062	309	63	26	18	98.080	99.591	4091.78	1.026
s9234	0	0	296	157	3	456	1.000	1.525	0	0	0	0	93.313	93.475	4.59	
s9234	1	5	219	80	80	379	0.831	1.268	65	12	0	0	93.313	93.475	76.47	
s9234	2	1	192	53	107	352	0.772	1.177	65	12	4	23	93.313	93.475	705.33	
s9234	2	10	151	12	148	311	0.682	1.040	76	14	17	38	93.313	93.475	2240.39	1.088
s38417	0	0	499	116	6	621	0.994	1.230	0	0	0	0	99.568	99.471	3.52	
s38417	1	3	422	39	83	544	0.870	1.077	0	77	0	0	99.568	99.471	192.86	
s38417	2	1	406	23	99	528	0.845	1.046	0	77	1	15	99.568	99.471	2655.80	
s38417	2	2	402	19	103	524	0.838	1.038	0	79	2	16	99.568	99.471	4084.56	0.866
s13207	0	0	436	244	1	681	0.999	1.558	0	0	0	0	96.824	98.462	4.05	
s13207	1	3	222	30	215	467	0.685	1.069	139	75	0	0	96.824	98.462	81.20	
s13207	2	1	205	13	232	450	0.660	1.030	139	75	8	9	96.824	98.462	594.46	
s13207	2	2	203	11	234	448	0.657	1.025	139	75	9	10	96.824	98.462	893.32	0.891
b15	0	0	397	306	104	807	0.989	1.611	0	0	0	0	98.311	98.660	3.85	
b15	1	7	123	32	378	533	0.653	1.064	245	29	0	0	98.311	98.660	66.80	
b15	2	1	113	22	388	523	0.641	1.044	245	29	5	5	98.317	98.660	189.39	
b15	2	4	101	10	400	511	0.626	1.020	246	31	8	11	98.325	98.660	359.70	1.204
s15850	0	0	355	130	2	487	0.992	1.364	0	0	0	0	95.116	96.682	4.23	
s15850	1	4	227	2	130	359	0.731	1.006	40	88	0	0	95.116	96.682	88.01	0.971
spi	0	0	460	405	1	866	1.000	1.879	0	0	0	0	99.708	99.985	3.10	
spi	1	8	63	8	398	469	0.542	1.017	359	38	0	0	99.708	99.985	56.70	
spi	2	1	58	3	403	464	0.536	1.007	359	38	3	2	99.708	99.985	82.35	
spi	2	2	56	1	405	462	0.533	1.002	359	38	5	2	99.708	99.985	92.23	0.938
des_area	0	0	112	118	0	230	1.000	1.949	0	0	0	0	100.000	100.000	3.01	
des_area	1	8	0	6	112	118	0.513	1.000	112	0	0	0	100.000	100.000	76.47	1.723

any of the faults it already detects out of $D_{tr,i}(t_j) \cup D_{sa,i}(t_k)$. It terminates successfully if $t_{j,k}$ detects all these faults.

Step 7)b) of Procedure 1 is modified as follows to accommodate the test modification option. A similar modification is performed to Step 8)b).

Procedure 2: Step 7)b) of test compaction with test modification

1) Assign $found = 0$. For every test $t_k \in T_{sa,i}$, as long as $found = 0$, if $|D_{sa,i}(t_k) \setminus E_{sa,i}(t_j)| \leq \Delta$:
   a) Attempt to find a test $t_{j,k}$ that detects all the faults in $D_{tr,i}(t_j) \cup D_{sa,i}(t_k)$. If $t_{j,k}$ is found:
      i) Add $t_{j,k}$ to $T_{com,i}$.
      ii) Remove t_j from $T_{tr,i}$.
      iii) Remove t_k from $T_{sa,i}$.
      iv) Assign $found = 1$.

VI. ITERATIONS

This section discusses the need for iterating the procedures from Sections IV and V.

At the beginning of iteration $i \geq 1$, $T_i = T_{i-1}$. The procedure performs fault simulation with fault dropping of F_{tr} first under $T_{com,i}$ and then under $T_{tr,i}$. Similarly, it performs fault simulation with fault dropping of F_{sa} first under $T_{com,i}$ and then under $T_{sa,i}$. This associates a set of detected faults $D_{tr,i}(t_j)$ with every test $t_j \in T_{tr,i}$, and $D_{sa,i}(t_j)$ with every test $t_j \in T_{sa,i}$.

If iteration i moves tests from $T_{tr,i}$ or $T_{sa,i}$ to $T_{com,i}$, fault simulation at the beginning of iteration $i+1$ can result in smaller subsets of detected faults $D_{tr,i+1}(t_j)$ for $t_j \in T_{tr,i+1}$ or $D_{sa,i+1}(t_j)$

for $t_j \in T_{sa,i+1}$. With smaller subsets of detected faults, the conditions $D_{tr,i+1}(t_k) \subseteq E_{tr,i+1}(t_j)$, $D_{sa,i+1}(t_k) \subseteq E_{sa,i+1}(t_j)$, $|D_{tr,i+1}(t_k) \backslash E_{tr,i+1}(t_j)| \leq \Delta$ and $|D_{sa,i+1}(t_k) \backslash E_{sa,i+1}(t_j)| \leq \Delta$ may be satisfied for additional pairs of tests. Thus, it may be possible to compact T_{i+1} further.

Starting from $T_0 = T_{tr,0} \cup T_{sa,0}$, Procedure 0 is applied once to update the partition of the test set T_0 into subsets. The test modification step included in Procedure 2 implies that Procedure 2 has a higher computational complexity than Procedure 1. Therefore, Procedure 1 is applied first. It is applied iteratively as long as it reduces the number of tests in T_i. Finally, Procedure 2 is applied iteratively as long as it reduces the number of tests in T_i.

VII. Experimental Results

This section presents the results of the test compaction procedure for benchmark circuits.

Procedure 2 is applied with $\Delta = 4$ as the maximum number of additional faults that a modified test needs to detect.

The results are given in Table I. The first row for every circuit describes the test set obtained by Procedure 0. The second row describes the test set obtained in the last iteration of Procedure 1 where it reduces the number of tests. If Procedure 2 reduces the number of tests, the next rows show the results after the first iteration, and the last one that reduces the number of tests. Intermediate iterations (not reported) show that the procedure can be terminated earlier without a significant loss in test compaction.

For every test set, column $proc$ shows which procedure is applied. Column $iter$ shows the iteration i of the procedure. Column $tests$ shows the numbers of tests in $T_{tr,i}$, $T_{sa,i}$, $T_{com,i}$ and T_i. With both broadside and skewed-load tests in $T_{tr,0}$, it is possible to reach a transition fault coverage that is similar to the stuck-at fault coverage, and a similar level of test compaction.

Column $tot/$ shows two ratios. Subcolumn $tot0$ shows the ratio $|T_i|/|T_0|$. This is the reduction in the total number of tests. Subcolumn $max0$ shows the ratio $|T_i|/max\{|T_{tr,0}| + |T_{com,0}|, |T_{sa,0}| + |T_{com,0}|\}$. A compact test set for transition and stuck-at faults is at least as large as a compact test set for one of the fault types alone. The larger of the two test sets contains the number of tests in the denominator. The ratio shows by how much the compact test set for both fault types is larger than this lower bound. The circuits are arranged by decreasing order of this ratio.

Column com shows the number of times a test is moved from $T_{tr,i}$ to $T_{com,i}$, followed by the number of times a test is moved from $T_{sa,i}$ to $T_{com,i}$, without and with modification.

Column $f.c.$ shows the transition fault coverage followed by the stuck-at fault coverage. Column $ntime$ shows the runtime for the computation of T_i divided by the runtime for fault simulation with fault dropping of T_0. This is referred to as the normalized runtime.

The ratio $tot/max0$ for the procedure from [7] is shown in the last column of Table I for comparison. A direct comparison with [7] is not possible since only skewed-load tests are used in [7], and test compaction does not preserve the numbers of tests for the individual fault types. However, the ratio $tot/max0$ can be computed using the final number of tests in [7] for tot, and the number of transition or stuck-at fault tests in the initial test set for $max0$.

The following points can be seen from Table I. Procedure 1 finds significant numbers of tests in $T_{tr,i}$ and $T_{sa,i}$ that can be used for removing tests from the other subset. This allows it to reduce the number of tests in T_i without increasing the number of tests required for detecting each fault type alone.

By modifying tests from $T_{tr,i}$ and $T_{sa,i}$, Procedure 2 finds additional tests that can replace pairs of tests. It thus reduces the number of tests in T_i further without increasing the number of tests required for detecting transition or stuck-at faults alone.

The ratio under column $tot/max0$ shows that the final test set T_i obtained by the test compaction procedure is within 10-20% of its lower bound for most of the circuits. This indicates that a significant level of test compaction is achieved. The ratio is in a similar range to the ratio obtained for [7].

For most of the circuits, tests for transition faults are more effective in detecting both fault types and thus providing test compaction. However, there are non-trivial numbers of stuck-at tests that contribute to test compaction.

The normalized runtime of Procedure 2 is significantly higher than that of Procedure 1. The runtime of Procedure 2 is higher for circuits where a slower reduction in the number of tests is obtained. When Procedure 2 performs a large number of iterations, it is possible to terminate it earlier without a significant loss in test compaction.

VIII. Concluding Remarks

This paper described a test compaction procedure for transition and stuck-at faults. The unique feature of the procedure is that it ensures that the compact test set contains compact test sets for each one of the fault types alone. To achieve this goal the procedure maintains a partition of the compact test set into a subset of tests that are used only for transition faults, a subset that is used only for stuck-at faults, and a common subset used for both fault types. Test compaction is achieved by increasing the number of common tests, and reducing the number of tests that are used only for one of the fault types. Experimental results for benchmark circuits demonstrate the ability to compact a test set under these conditions.

References

[1] L. N. Reddy, I. Pomeranz and S. M. Reddy, "COMPACTEST-II: A Method to Generate Compact Two-pattern Test Sets for Combinational Logic Circuits", in Proc. Intl. Conf. on Computer-Aided Design, 1992, pp. 568-574.

[2] R. Desineni, K. N. Dwarkanath and R. D. Blanton, "Universal Test Generation using Fault Tuples", in Proc. Intl. Test Conf., 2000, pp. 812-819.

[3] S. M. Reddy, G. Chen, J. Rajski, I. Pomeranz, P. Engelke and B. Becker, "A Unified Fault Model and Test Generation Procedure for Interconnect Opens and Bridges", in Proc. Europ. Test Symp., 2005, pp. 22-27.

[4] S. Goel and R. A. Parekhji, "Choosing the Right Mix of At-Speed Structural Test Patterns: Comparisons in Pattern Volume Reduction and Fault Detection Efficiency", in Proc. Asian Test Symp., 2005, pp. 330-336.

[5] S. Alampally, R. T. Venkatesh, P. Shanmugasundaram, R. A. Parekhji and V. D. Agrawal, "An Efficient Test Data Reduction Technique through Dynamic Pattern Mixing Across Multiple Fault Models", in Proc. VLSI Test Symp., 2011, pp. 285-290.

[6] C.-H. Wu and K.-J. Lee, "Transformation of Multiple Fault Models to a Unified Model for ATPG Efficiency Enhancement", in Proc. Int. Test Conf., 2016, pp. 1-10.

[7] I. Pomeranz, "Skewed-Load Tests for Transition and Stuck-at Faults", IEEE Trans. on Computer-Aided Design, Vol 38, No. 10, Oct. 2019, pp. 1969-1973.

[8] F. Hapke and J. Schloeffel, "Introduction to the Defect-oriented Cell-aware Test Methodology for Significant Reduction of DPPM Rates", in Proc. European Test Symp., 2012, pp. 1-6.

[9] J. Savir and S. Patil, "Scan-Based Transition Test", IEEE Trans. on Computer-Aided Design, Aug. 1993, pp. 1232-1241.

[10] J. Savir and S. Patil, "Broad-Side Delay Test", IEEE Trans. on Computer-Aided Design, Aug. 1994, pp. 1057-1064.

[11] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1995.

[12] Anonymous Reviewer.