Test Compaction by Test Removal under Transparent-Scan

Irith Pomeranz

***Abstract*— This paper describes a new approach to test compaction under transparent-scan. Transparent-scan achieves higher levels of test compaction than possible with conventional scan-based tests by interleaving scan shift cycles and functional clock cycles in arbitrary ways. Earlier approaches relied on the computation of a single transparent-scan sequence, and the omission of test vectors from it. However, a single transparent-scan sequence can be prohibitively long. In the approach described in this paper, a transparent-scan test set consists of several sequences whose lengths are limited. Test compaction is achieved by combining sequences into longer sequences that detect more faults, and removing from the test set entire sequences that become unnecessary. Experimental results for benchmark circuits demonstrate the ability of the procedure to achieve test compaction without creating prohibitively long transparent-scan sequences.**

***Index Terms*— full-scan circuits, test compaction, test generation, transparent-scan.**

I. Introduction

Test compaction procedures reduce the number of tests that need to be applied for the detection of target faults, thus reducing the test application time and test data volume [1]-[4]. By using multicycle tests it is possible to reduce the number of tests further compared to single-cycle or two-cycle tests [5]-[11]. A multicycle test typically has multiple functional clock cycles between its scan operations. The reduction in the number of tests is achieved because each functional clock cycle can activate and propagate additional faults, resulting in a test that detects more faults. The test application time is reduced because scan operations dominate the test application time, and fewer tests require fewer scan operations.

Another approach to reducing the test application time is transparent-scan [12]. Under transparent-scan, the scan enable and scan chain inputs of a standard-scan circuit are considered as regular primary inputs, and the scan chain outputs are considered as regular primary outputs. As a regular primary input, the scan enable input is allowed to assume arbitrary sequences of values. This allows, e.g., shorter scan operations (shorter runs of scan shift cycles) to be used compared with a conventional scan-based test set. In general it allows to create arbitrary sequences of scan shift and functional clock cycles. This can be used for reducing the number of clock cycles that are required for detecting a given set of faults.

In the procedure described in [12], a conventional scan-based test set is first translated into a transparent-scan sequence. This is achieved by translating every scan shift cycle and every functional clock cycle of the test set into a corresponding clock cycle of the transparent-scan sequence. The transparent-scan sequence is then compacted by omitting clock cycles that are not necessary for the detection of target faults. The omission of clock cycles creates runs of scan shift and functional clock cycles that do not exist in the initial sequence. Test compaction is achieved because the sequence requires fewer clock cycles for test application.

To enhance the ability to omit clock cycles from a transparent-scan sequence, the procedure described in [13] embeds broadside tests in the transparent-scan sequence. This is achieved by replacing pairs of scan shift cycles with pairs of functional clock cycles when these are surrounded by additional scan shift cycles. After the modification, more clock cycles contribute to the detection of faults, and other clock cycles can be omitted.

The procedure described in [14] achieves test compaction by sharing transparent-scan sequences among different circuits. The original primary inputs and outputs of every circuit are included in a single scan chain together with its state variables. Consequently, every circuit has two inputs. Every circuit also has its own conventional scan-based test set. The procedure translates every test into a two-bit transparent-scan sequence. All the sequences have a similar format but possibly different lengths for different circuits. This allows sharing of test sequences among different circuits.

This paper describes a new approach to test compaction for a single circuit under transparent-scan that makes the following contribution.

Instead of computing a single transparent-scan sequence as in [12] and [13], the procedure described in this paper computes a set of shorter transparent-scan sequences where the length of a sequence is limited. The importance of limiting the length of a transparent-scan sequence results from the fact that this length can be prohibitive. Let us consider a circuit with k state variables that are included in a single scan chain. Suppose that a conventional single-cycle test set C for the circuit contains m tests. The number of clock cycles required for applying the test set C is $(m+1)k+m$. This number is obtained by overlapping the scan-out operation of one test with the scan-in operation of the next test. In this case, m tests require $m+1$ scan operations of k clock cycles, and m functional clock cycles. If C is translated into a single transparent-scan sequence T, the length of T is also equal to $(m+1)k+m$. A test compaction procedure that is applied to T needs to consider the set of target faults F under a sequence of this length.

The procedure described in this paper first translates every test $c_i \in C$ into a transparent-scan sequence S_i, and includes S_i in a test set that is denoted by S. The length of S_i is equal to $2k+1$ for a circuit with k state variables that are included in a single scan chain. If all the transparent-scan sequences in S are combined, by overlapping the scan-out operation of one test with the scan-in operation of the next test, a single transparent-scan sequence of length $(m+1)k+m$ will be obtained. However, the procedure limits the number of sequences that it may combine into a single sequence by using a constant that is denoted by μ. The number of clock cycles required for applying μ conventional scan-based tests is $(\mu+1)k+\mu$. This is also the largest length of a transparent-scan sequence that the procedure may create by combining sequences in S. By keeping $\mu \ll m$, the procedure avoids the need to consider all the target faults under the $(m+1)k+m$ clock cycles of a single transparent-scan sequence.

Another important difference between the test compaction procedure described in this paper and earlier ones is in the approach to test compaction. To achieve test compaction, the procedures from [12] and [13] rely on the omission of clock cycles from a transparent-scan sequence. Instead, the procedure described in this paper removes entire sequences from S. It is important to note in this regard that, for the discussion in this paper, the conventional scan-based test set C is already compact. Being compact, no test can be removed from C without losing fault coverage. The test compaction procedure combines transparent-scan sequences in S to obtain new transparent-scan sequences of limited lengths, and modifies the sequences to increase the numbers of faults they detect. This allows it to remove other sequences from S.

In addition, in [13] as well as in works on the compaction of functional test sequences, the omission of clock cycles, and the modification of a sequence, are implemented by separate procedures. This is appropriate since the goal of the modification in earlier works is to enhance the ability to achieve test compaction. In the procedure described in this paper, the goal is to increase the number of faults

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

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

that a combined sequence detects in order to allow other sequences to be removed. The omission of clock cycles can contribute to this goal. Furthermore, it can contribute in ways that cannot be materialized if the two procedures are applied separately. Therefore, the modification of the sequence includes the omission of clock cycles in the procedure described in this paper.

For simplicity of discussion, all the state variables of a circuit are assumed to be included in a single scan chain of length k, with a scan enable input s_{en}, and a scan chain input s_{in}. This affects only the translation of a conventional scan-based test into a transparent-scan sequence. A scan shift cycle has $s_{en} = 1$, and a functional clock cycle has $s_{en} = 0$.

The conventional scan-based test set C consists of single-cycle tests for single stuck-at faults. The use of two-cycle or multicycle tests affects only the translation of C into an initial set S of transparent-scan sequences. Different fault models can be accommodated by using different fault simulation procedures.

To increase the opportunities for faults to be detected under transparent-scan, original primary outputs and scan chain outputs are observed at every clock cycle of a transparent-scan sequence. Output compaction is appropriate for implementing this option. No other hardware cost is associated with transparent-scan. The computational cost of transparent-scan is related to the need to perform sequential fault simulation for transparent-scan sequences. This cost is reduced (but not eliminated) by limiting the lengths of the transparent-scan sequences as suggested in this paper.

The paper is organized as follows. Section II discusses the transparent-scan test set S that is obtained from a conventional scan-based test set C. The test set S is the initial test set for the test compaction procedure that is described in Section III. Section IV provides experimental results.

II. Initial Test Set

This section describes the derivation of an initial set of transparent-scan sequences S from a given conventional single-cycle test set C.

A conventional single-cycle test $c_i \in C$ has the form $c_i = \langle p_i, v_i \rangle$, where p_i is a scan-in state, and v_i is a primary input vector. After p_i is scanned in, the primary input vector v_i is applied during a functional clock cycle. The test ends with a scan-out operation.

The test $c_i = \langle p_i, v_i \rangle$ is translated into a transparent-scan sequence of length $2k + 1$ that includes k scan shift cycles for a scan-in operation, a functional clock cycle, and k scan shift cycles for a scan-out operation. A clock cycle of the sequence is denoted by u. The sequence assigned to the original primary inputs is denoted by V_i. The original primary input vector at clock cycle u is denoted by $V_i(u)$. The sequence assigned to the scan enable input s_{en} is denoted by EN_i, with $EN_i(u)$ being the value of s_{en} at clock cycle u. The sequence assigned to the scan chain input s_{in} is denoted by IN_i, with $IN_i(u)$ being the value of s_{in} at clock cycle u. This notation is illustrated by Table I. For Table I, $c_i = \langle 00110, 01 \rangle$.

The transparent-scan sequence corresponding to c_i is obtained as follows. The scan-in operation is translated into k scan shift cycles with $0 \leq u \leq k - 1$, and $EN_i(u) = 1$. $IN_i(u)$ assumes the corresponding bits of p_i such that the circuit reaches state p_i at clock cycle k. In Table I, with $p_i = 00110$, $EN_i(u) = 1$ for $0 \leq u \leq 4$, $IN_i(0) = 0$, $IN_i(1) = 1$, $IN_i(2) = 1$, $IN_i(3) = 0$ and $IN_i(4) = 0$.

The functional clock cycle of the test is translated into $EN_i(u) = 0$ and $V_i(u) = v_i$ for $u = k$. In Table I, $EN_i(5) = 0$ and $V_i(5) = 01$.

The scan-out operation at the end of the test is translated into k scan shift cycles where $EN_i(u) = 1$, for $k + 1 \leq u \leq 2k$. In Table I, $EN_i(u) = 1$ for $6 \leq u \leq 10$.

TABLE I
Transparent-Scan Sequence

	(a) Initial Translation			(b) Final Translation		
u	$V_i(u)$	$EN_i(u)$	$IN_i(u)$	$V_i(u)$	$EN_i(u)$	$IN_i(u)$
0	xx	1	0	01	1	0
1	xx	1	1	01	1	1
2	xx	1	1	01	1	1
3	xx	1	0	01	1	0
4	xx	1	0	01	1	0
5	01	0	x	01	0	0
6	xx	1	x	xx	1	x
7	xx	1	x	xx	1	x
8	xx	1	x	xx	1	x
9	xx	1	x	xx	1	x
10	xx	1	x	xx	1	x

Values that are not specified by c_i are left unspecified (x) in Table I(a). Some of these values are kept unspecified in order to allow overlapping of S_i with other transparent-scan sequences, while other values are specified in order to increase the ability of the sequence to detect target faults. The unspecified values are discussed next. Table I(b) shows the final sequence that is included in S based on the sequence from Table I(a).

The unspecified primary input vectors at clock cycles $0 \leq u < k$ are specified to be equal to v_i, or $V_i(u) = v_i$ for $0 \leq u < k$.

The unspecified primary input vectors at clock cycles $k+1 \leq u \leq 2k$ are left unspecified. If another sequence is overlapped with S_i, the primary input vectors from this sequence will be used.

The unspecified value on $IN_i(k)$ is assigned a zero value by default. This value is not important for fault detection since clock cycle $u = k$ is a functional clock cycle.

Considering the unspecified values of $IN_i(u)$, for $k+1 \leq u \leq 2k$, these values are needed for overlapping S_i with another sequence. Therefore, they are left unspecified.

After the initial set of transparent-scan sequences S is obtained, fault simulation with fault dropping is carried out for the set of target faults F. Considering a transparent-scan sequence $S_i \in S$, sequential fault simulation is carried out for F to find the subset of target faults that the sequence detects. This subset is denoted by F_i.

The tester memory required for transparent-scan is as follows. The original primary inputs are held constant for the duration of a subsequence that corresponds to a single-cycle test in C, and ends with its functional clock cycle. Therefore, the number of original primary input vectors that need to be stored is not increased relative to C. The scan enable sequence needs to be stored together with the scan chain input sequence. This doubles the amount of memory for the scan chain input sequence when the circuit has a single scan chain. With $m \geq 1$ scan chains, the increase because of the scan enable sequence is $(m+1)/m \leq 2$. Test compaction may reduce the number of original primary input vectors, the number of scan enable and scan chain input sequences, and their lengths.

III. Test Compaction Procedure

This section describes the test compaction procedure. The procedure is illustrated by Figure 1.

A. Combining Transparent-Scan Sequences

The test compaction procedure combines a pair of transparent-scan sequences, $S_i \in S$ and $S_j \in S$, in order to obtain a longer sequence. With a longer sequence, the procedure has more flexibility to interleave functional clock cycles and scan shift cycles in ways that increase the number of detected faults. Creating longer sequences that detect more faults allows other sequences to be removed from S. The combination of sequences is performed within the limit of $(\mu + 1)k + \mu$ on the length of a transparent-scan sequence.

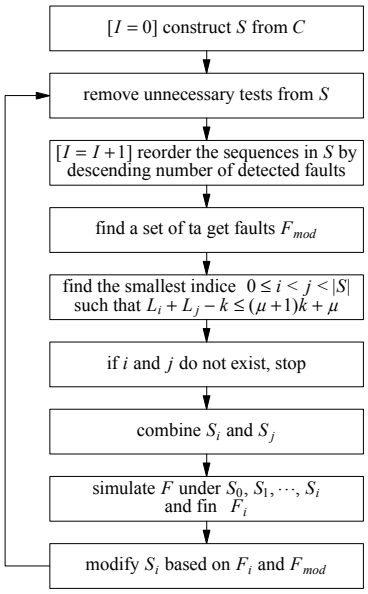

Fig. 1. Test compaction procedure

TABLE II
COMBINATION AND MODIFICATION OF TRANSPARENT-SCAN SEQUENCES

	(a) Combination			(b) Modification			
u	$V_i(u)$	$EN_i(u)$	$IN_i(u)$	$V_i(u)$	$EN_i(u)$	$IN_i(u)$	$\Omega_i(u)$
0	01	1	0	11	1	0	0
1	01	1	1	11	1	0	0
2	01	1	1	11	1	1	0
3	01	1	0	11	1	0	0
4	01	1	0	11	1	1	0
5	01	0	0	11	0	0	0
6	10	1	1	00	0	1	0
7	10	1	1	00	0	1	1
8	10	1	0	00	1	1	0
9	10	1	0	00	0	0	0
10	10	1	1	00	1	0	0
11	10	0	0	00	0	0	1
12	xx	1	x	xx	1	x	0
13	xx	1	x	xx	1	x	0
14	xx	1	x	xx	1	x	0
15	xx	1	x	xx	1	x	0
16	xx	1	x	xx	1	x	0

To describe the combination of a pair of transparent-scan sequences, the length of S_i is denoted by L_i. The vector at clock cycle u of S_i is denoted by $S_i(u)$. We have that $S_i(u) = V_i(u)EN_i(u)IN_i(u)$.

When a pair of sequences, $S_i \in S$ and $S_j \in S$, is combined, the length of the resulting sequence is $L_i + L_j - k$ (because of the overlapping of the scan-out operation of S_i with the scan-in operation of S_j). The resulting sequence replaces S_i. The new S_i detects all the faults that S_i and S_j detect (all the faults in F_i and F_j). This always allows S_j to be removed from S. The procedure combines $S_i \in S$ and $S_j \in S$ such that $i < j$ and $L_i + L_j - k \leq (\mu+1)k + \mu$.

Table II(a) shows an example where two transparent-scan sequences $S_i \in S$ and $S_j \in S$ are combined into a single sequence that replaces S_i. The sequences are obtained from the conventional scan-based tests $c_i = \langle 00110, 01\rangle$ (whose transparent-scan sequence is shown in Table I), and $c_j = \langle 10011, 10\rangle$. The combined sequence can be further combined with another sequence.

In general, combining S_i and S_j, and placing the result in S_i, is performed as follows.

(1) The first k clock cycles of S_j are copied into the last k clock cycles of S_i. This overlaps the scan-out operation of S_i with the scan-in operation of S_j. Because of the unspecified values in the last clock cycles of S_i, none of the values that are important for the application of S_i is lost. This is implemented by assigning $S_i(u_i) = S_j(u_j)$ for $(u_i, u_j) = (L_i - k, 0)$, $(L_i - k + 1, 1)$, ..., $(L_i - 1, k - 1)$.

(2) The remaining clock cycles of S_j are concatenated to S_i. This is implemented by assigning $S_i(L_i) = S_j(u_j)$ and $L_i = L_i + 1$ for $u_j = k$, $k + 1$, ..., $L_j - 1$.

Fault simulation with fault dropping of F under S_0, S_1, ..., S_i updates the set of faults that S_i detects after it is combined with S_j.

B. Modifying S_i

Let S_i be obtained by combining S_i and S_j. Let F_i be the subset of faults that S_i detects after the combination. The test compaction procedure modifies S_i to increase the number of faults it detects by targeting two subsets of faults. (1) The procedure requires S_i to continue detecting all the faults in F_i. (2) The procedure attempts to ensure that S_i would detect as many additional faults as possible. The faults that the procedure targets are detected by sequences in S with small numbers of detected faults. Such sequences are likely to be the easiest to remove. The set of target faults is denoted by F_{mod}. The derivation of F_{mod} is described later.

The procedure accepts modifications of S_i where all the faults from F_i are detected, and the number of detected faults from F_{mod} does not decrease. Other modifications are not accepted.

A modification of S_i must also ensure that S_i can be overlapped with other sequences. For this purpose, it is necessary to maintain the scan-in operation at the beginning of S_i, the scan-out operation at the end of S_i, and the unspecified values at the end of S_i. Every other bit of S_i can be complemented. This includes the bits of $V_i(u)$ and $IN_i(u)$ for $0 \leq u < L_i - k$, and $EN_i(u)$ for $k \leq u < L_i - k$.

In addition, a clock cycle of S_i can be omitted. As discussed earlier, the modification of the sequence and the omission of clock cycles are considered together since both can contribute to an increase in the number of detected faults, and they may interact in ways that cannot be utilized if they are considered separately.

For uniformity, the option of omitting a clock cycle from S_i is implemented by introducing a sequence that is denoted by Ω_i. We have that $\Omega_i(u) = 0$ if clock cycle u is retained in S_i, and $\Omega_i(u) = 1$ if clock cycle u is omitted. After the modification of S_i is complete, clock cycle u is omitted from S_i for every u such that $\Omega_i(u) = 1$. Initially, $\Omega_i(u) = 0$ for $0 \leq u < L_i$. The procedure may complement $\Omega_i(u)$, for $k \leq u < L_i - k$, in order to omit clock cycle u.

The possibility of complementing $EN_i(u)$ or $\Omega_i(u)$, for $k \leq u < L_i - k$, contributes to test compaction that cannot be achieved with conventional scan-based tests. The complementation of these bits allows arbitrary subsequences of scan shift cycles and functional clock cycles to be formed that are only possible under transparent-scan. Complementing some of the other bits ensures that the overall effect of using transparent-scan is as significant as possible.

When the procedure modifies $V_i(u)$, it ensures that the number of different original primary input vectors in S_i does not increase. This is achieved by considering a single original primary input for modification, and complementing it for every clock cycle u such that $0 \leq u < L_i - k$. Denoting the number of original primary inputs by n, and an original primary input by a_p, for $0 \leq p < n$, the set of bits that the procedure considers for complementation is $B = \{a_p : 0 \leq p < n\} \cup \{EN_i(u) : k \leq u < L_i - k\} \cup \{IN_i(u) : 0 \leq u < L_i - k\} \cup \{\Omega_i(u) : k \leq u < L_i - k\}$.

Table II(b) shows a modified transparent-scan sequence that may be obtained from the sequence in Table II(a). The modified sequence is obtained by complementing a_0, $EN_i(u)$ for $u = 6$, 7 and 9, $IN_i(u)$ for $u = 1$, 4, 8 and 10, and $\Omega_i(u)$ for $u = 7$ and 11. Clock cycles $u = 7$ and 11 need to be omitted to obtain the final sequence.

The procedure considers the bits from B one at a time in a random order. Before this process starts, $D_{mod,i} = \emptyset$ indicates that S_i does not detect any faults from $F_{mod} - F_i$. When a bit $b \in B$ is considered, the procedure complements it to obtain from S_i a sequence that is denoted by $S_{i,b}$. The procedure performs fault simulation of F_i under $S_{i,b}$ to determine whether all the faults in F_i are detected. If they are, the procedure simulates $F_{mod} - F_i$ under $S_{i,b}$. Let the subset of detected faults be $D_{mod,i,b}$. The procedure accepts the complementation if $|D_{mod,i,b}| \geq |D_{mod,i}|$. In this case, it assigns $S_i = S_{i,b}$. Otherwise, it discards $S_{i,b}$.

C. Additional Procedures and Computational Effort

The procedure from Figure 1 removes unnecessary sequences from the initial test set S, and after it combines a pair of sequences. When a test $c_i \in C$ is translated into a transparent-scan sequence $S_i \in S$, the original primary outputs and scan chain output are observed at every clock cycle of S_i. As a result, S_i may detect more faults than c_i. Considering the initial transparent-scan test set S, a sequence $S_i \in S$ may be unnecessary even though c_i is necessary in C. In addition, as the test compaction procedure combines and modifies sequences in S to increase the numbers of faults they detect, other sequences in S become unnecessary.

To identify sequences that can be removed from S, the test compaction procedure performs fault simulation followed by forward-looking reverse order fault simulation of S. A sequence that does not detect any faults is removed from S by this process.

For the procedure to be as effective as possible, it is advantageous if the sequences is combines, S_i and S_j, detect the largest numbers of faults in S. The test compaction procedure uses fault simulation with fault dropping to reorder S such that sequences appear by order of descending number of detected faults (third step in Figure 1). For a sequence S_i, the set of detected faults is denoted by F_i. If $|F_i| < |F_j|$ is obtained for $i < j$, reordering ensures that S_j appears in S before S_i. After reordering, the procedure repeats the fault simulation process to recompute the sets of detected faults. Reordering and fault simulation with fault dropping are performed until fault simulation with fault dropping yields $|F_0| \geq |F_1| \geq |F_2| \geq ....$

In this order, the procedure combines S_i and S_j, for the lowest i and j, whose combined length does not exceed the limit given by μ. By using the lowest possible values of i and j, the procedure considers the sequences with the largest sets of detected faults.

The sequences that are the most likely to be removed from S are the ones at the end of S, detecting the smallest numbers of faults. The procedure defines the set F_{mod} of target faults for the modification of S_i based on these sequences (fourth step in Figure 1) as follows.

The procedure first assigns $F_{mod} = \emptyset$. For every sequence $S_a \in S$ with the smallest $|F_a|$, the procedure adds the faults from F_a to F_{mod}. It should be noted that, even if fault simulation of S_0, S_1, ..., S_i (eighth step in Figure 1) updates F_i to include faults from F_{mod}, the procedure still targets the faults in F_i and $F_{mod} - F_i$ for the modification of S_i.

The worst-case computational effort of the test compaction procedure is determined by the modification of combined sequences, as follows. For a conventional scan-based test set C that contains m tests, the procedure obtains an initial transparent-scan test set with m sequences. The procedure combines a pair of sequences in every iteration. Therefore, the number of iterations is $O(m)$. This is also the number of combined sequences that the procedure considers.

To modify a combined sequence, the procedure considers $O(n+k)$ bits in the set B. For every bit it simulates at most $|F_i \cup F_{mod}|$ faults, or $O(|F|)$ faults. With a constant for μ, the length of a sequence is $O(k)$. Overall, the computational effort of the procedure is determined by the need to perform fault simulation for $O(m(n+k)k|F|)$ faults and clock cycles.

IV. Experimental Results

The test compaction procedure from Figure 1 is applied to single stuck-at faults in benchmark circuits using a compact scan-based test set for C. The procedure is also applied to sequence detectors for the all-one sequence of length 64, 128, 256 and 512 to illustrate its performance for a functional unit as its size is increased.

The procedure is applied with $\mu = 4$ to limit the length of a transparent-scan sequence. Since the omission of clock cycles may reduce the length of a sequence, it is possible for the procedure to combine more than four sequences without exceeding this limit every time $k + 1$ or more clock cycles are omitted.

The level of test compaction is measured by the number of clock cycles that are required for applying S. This number is computed as $cyc = k + \sum\{L_i - k : S_i \in S\}$, taking into consideration that scan-out and scan-in operations will be overlapped for consecutive tests. For ease of references, cyc_I is the number of clock cycles required for applying the test set obtained in iteration I. The initial test set corresponds to iteration $I = 0$, with cyc_0 clock cycles.

The results are shown in Table III as follows. The first row for every circuit describes the initial transparent-scan test set S. Additional rows describe the test set S when the number of clock cycles, cyc_I, is reduced to approximately $0.9cyc_0$, $0.8cyc_0$, ... The last row describes the final test set.

After the circuit name, column sv shows the number of state variables, k. Column $iter$ shows the iteration of the test compaction procedure. Column $Fmod$ shows the fraction of faults in F_{mod} for the iteration, $|F_{mod}|/|F|$.

Column seq shows the number of transparent-scan sequences in S. Column max shows the maximum length of a transparent-scan sequence. Column $cycles$ shows the total number of clock cycles cyc_I required for applying all the sequences in S. Column $ratio$ shows the ratio cyc_I/cyc_0.

Column $f.c.$ shows the fault coverage of S. The test compaction procedure ensures that the fault coverage does not decrease. Column $ntime$ shows the normalized run time, where the cumulative run time of the test compaction procedure is divided by the run time for obtaining the initial transparent-scan test set S from C. This includes only fault simulation with fault dropping and forward-looking reverse order fault simulation of S.

The following points can be seen from Table III. There are many circuits where the test compaction procedure reduces the number of clock cycles required for applying the transparent-scan test set significantly compared with the initial test set, and compared with a compact conventional scan-based test set.

For some circuits, the first significant reduction in the number of clock cycles is achieved after a small number of iterations. This implies that sequences, other than the ones combined, can be removed from the test set after a small number of iterations. For other circuits, it takes several iterations before other sequences can be removed.

The normalized run time is similar for benchmark circuits of different sizes. This indicates that the procedure scales similar to a fault simulation procedure for transparent-scan sequences of minimal length, as in the initial test set. In a large design, fault simulation is expected to be applicable to logic blocks within the design. To further demonstrate the scalability of the procedure, Figure 2 shows the normalized run time as a function of the number of state variables when the number of clock cycles is reduced to approximately $0.9cyc_0$. Figure 2 demonstrates that the normalized run time does not increase with the number of state variables, which is used for measuring the circuit size. The sequence detectors further demonstrate the scalability

TABLE III
EXPERIMENTAL RESULTS

circuit	sv	iter	Fmod	seq	max	cycles	ratio	f.c.	ntime
s1423	74	0	-	26	149	2024	1.000	99.076	1.00
s1423	74	13	0.00198	10	371	1762	0.871	99.076	28.09
s1423	74	16	0.00132	6	371	1665	0.823	99.076	30.67
s5378	179	0	-	100	359	18179	1.000	99.131	1.00
s5378	179	14	0.00326	76	899	16191	0.891	99.131	35.16
s5378	179	29	0.00348	55	899	14330	0.788	99.131	52.04
s5378	179	43	0.00304	37	899	12634	0.695	99.131	65.15
s5378	179	61	0.00043	18	899	10993	0.605	99.131	78.53
s9234	228	0	-	111	457	25647	1.000	93.475	1.00
s9234	228	80	0.00029	27	1136	23352	0.911	93.475	98.31
s13207	669	0	-	183	1339	123279	1.000	98.462	1.00
s13207	669	4	0.00540	153	3344	105854	0.859	98.462	20.36
s13207	669	5	0.00448	138	3344	96474	0.783	98.462	22.44
s13207	669	8	0.00224	108	3346	78373	0.636	98.462	27.92
s13207	669	10	0.00153	98	3346	73011	0.592	98.462	30.55
s13207	669	15	0.00071	74	3346	60187	0.488	98.462	35.08
b04	66	0	-	42	133	2880	1.000	99.851	1.00
b04	66	7	0.00966	29	331	2464	0.856	99.851	21.20
b04	66	12	0.00817	21	331	2201	0.764	99.851	28.20
b04	66	18	0.00669	13	331	1990	0.691	99.851	33.51
b04	66	22	0.00149	8	331	1803	0.626	99.851	36.12
b14	247	0	-	327	495	81343	1.000	94.870	1.00
b14	247	33	0.01262	261	1234	72983	0.897	94.870	59.20
b14	247	71	0.01272	202	1234	67276	0.827	94.870	103.31
des_area	128	0	-	115	257	14963	1.000	100.000	1.00
des_area	128	13	0.00101	95	639	13386	0.895	100.000	34.79
des_area	128	24	0.00101	82	639	11936	0.798	100.000	48.33
des_area	128	35	0.00101	70	639	10441	0.698	100.000	60.23
des_area	128	45	0.00101	57	639	8969	0.599	100.000	73.01
des_area	128	54	0.00081	45	639	7430	0.497	100.000	79.34
des_area	128	66	0.00101	33	639	5958	0.398	100.000	89.25
des_area	128	78	0.00101	21	639	4468	0.299	100.000	99.30
des_area	128	89	0.00040	9	639	2915	0.195	100.000	108.07
des_area	128	92	0.00010	6	639	2574	0.172	100.000	111.01
i2c	128	0	-	45	257	5933	1.000	100.000	1.00
i2c	128	21	0.00214	20	641	5309	0.895	100.000	53.79
i2c	128	30	0.00043	10	641	4956	0.835	100.000	63.53
pci_spoci_ctrl	60	0	-	146	121	8966	1.000	99.942	1.00
pci_spoci_ctrl	60	84	0.01514	52	299	8003	0.893	99.942	94.40
pci_spoci_ctrl	60	102	0.00058	34	299	7908	0.882	99.942	108.32
sasc	117	0	-	22	235	2713	1.000	100.000	1.00
sasc	117	6	0.00059	14	578	2439	0.899	100.000	47.22
sasc	117	9	0.00356	8	578	2036	0.750	100.000	54.67
sasc	117	10	0.00237	6	578	1892	0.697	100.000	55.68
sasc	117	11	0.00119	4	578	1677	0.618	100.000	56.52
simple_spi	131	0	-	36	263	4883	1.000	100.000	1.00
simple_spi	131	11	0.00095	22	650	4333	0.887	100.000	45.69
simple_spi	131	18	0.00143	15	650	3876	0.794	100.000	54.06
simple_spi	131	24	0.00095	8	650	3383	0.693	100.000	59.13
spi	229	0	-	394	459	90849	1.000	99.985	1.00
spi	229	34	0.01185	321	1147	81708	0.899	99.985	55.24
spi	229	51	0.00985	265	1147	72631	0.799	99.985	69.61
spi	229	74	0.00785	198	1147	62342	0.686	99.985	85.01
spi	229	93	0.00446	140	1147	53181	0.585	99.985	94.23
systemcdes	190	0	-	78	381	15088	1.000	100.000	1.00
systemcdes	190	3	0.00030	65	928	13152	0.872	100.000	20.81
systemcdes	190	4	0.00045	58	928	11989	0.795	100.000	21.97
systemcdes	190	5	0.00106	48	928	10246	0.679	100.000	23.42
systemcdes	190	6	0.00030	38	928	8482	0.562	100.000	25.48
systemcdes	190	7	0.00151	31	928	7332	0.486	100.000	26.00
systemcdes	190	9	0.00061	20	928	5570	0.369	100.000	27.13
systemcdes	190	11	0.00061	12	928	4358	0.289	100.000	27.90
systemcdes	190	13	0.00015	5	928	3184	0.211	100.000	28.35
usb_phy	98	0	-	32	197	3266	1.000	100.000	1.00
usb_phy	98	10	0.00538	19	481	2889	0.885	100.000	38.13
usb_phy	98	12	0.00385	14	481	2584	0.791	100.000	41.91
usb_phy	98	16	0.00308	8	481	2222	0.680	100.000	46.81
usb_phy	98	18	0.00077	6	481	2144	0.656	100.000	48.21
all1sd.64	63	0	-	65	127	4223	1.000	100.000	1.00
all1sd.64	63	1	0.00262	1	191	191	0.045	100.000	1.80
all1sd.128	127	0	-	129	255	16639	1.000	100.000	1.00
all1sd.128	127	1	0.00131	1	383	383	0.023	100.000	1.70
all1sd.256	255	0	-	257	511	66047	1.000	100.000	1.00
all1sd.256	255	1	0.00065	1	767	767	0.012	100.000	1.70
all1sd.512	511	0	-	513	1023	263167	1.000	100.000	1.00
all1sd.512	511	1	0.00033	1	1535	1535	0.006	100.000	1.76

of the procedure. For these circuits also, the normalized run time does not increase as the circuit size is increased.

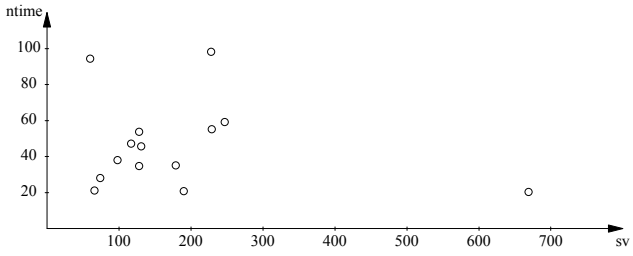

Fig. 2. Scalability

V. CONCLUDING REMARKS

This paper described a new approach to test compaction under transparent-scan. In this approach, a transparent-scan test set consists of several transparent-scan sequences whose lengths are limited. As sequences are combined into longer sequences that detect more faults, without exceeding the length limit, test compaction is achieved by removing entire sequences from the test set, and to a lesser extent, omitting clock cycles from the combined sequences. Experimental results for benchmark circuits demonstrated the ability of the procedure to achieve test compaction without creating prohibitively long transparent-scan sequences.

REFERENCES

[1] P. Goel and B. C. Rosales, "Test Generation and Dynamic Compaction of Tests", in Proc. Test Conf., 1979 pp. 189-192.

[2] I. Pomeranz, L. N. Reddy and S. M. Reddy, "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits", in Proc. Intl. Test Conf., 1991, pp. 194-203.

[3] J.-S. Chang and C.-S. Lin, "Test Set Compaction for Combinational Circuits", in Proc. Asian Test Symp., 1992, pp. 20-25.

[4] 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.

[5] S. Y. Lee and K. K. Saluja, "Test Application Time Reduction for Sequential Circuits with Scan", IEEE Trans. on Computer-Aided Design, Sept. 1995, pp. 1128-1140.

[6] I. Pomeranz and S. M. Reddy, "Static Test Compaction for Scan-Based Designs to Reduce Test Application Time", in Proc. Asian Test Symp., 1998, pp. 198-203.

[7] X. Lin and R. Thompson, "Test Generation for Designs with Multiple Clocks", in Proc. Design Autom. Conf., 2003, pp. 662-667.

[8] G. Bhargava, D. Meehl and J. Sage, "Achieving Serendipitous N-Detect Mark-Offs in Multi-Capture-Clock Scan Patterns", in Proc. Intl. Test Conf, 2007, Paper 30.2.

[9] D. Erb, K. Scheibler, M. Sauer, S. M. Reddy and B. Becker, "Multi-cycle Circuit Parameter Independent ATPG for Interconnect Open Defects", in Proc. VLSI Test Symp., 2015, pp. 1-6.

[10] I. Pomeranz, "A Multi-Cycle Test Set Based on a Two-Cycle Test Set with Constant Primary Input Vectors", IEEE Trans. on Computer-Aided Design, July 2015, pp. 1124-1132.

[11] S. Wang, H. T. Al-Awadhi, S. Hamada, Y. Higami, H. Takahashi, H. Iwata and J. Matsushima, "Structure-Based Methods for Selecting Fault-Detection-Strengthened FF under Multi-Cycle Test with Sequential Observation", in Proc. Asian Test Symp., 2016, pp. 209-214.

[12] I. Pomeranz and S. M. Reddy, "Transparent Scan: A New Approach to Test Generation and Test Compaction for Scan Circuits that Incorporates Limited Scan Operations", IEEE Trans. on Computer-Aided Design, Dec. 2003, pp. 1663-1670.

[13] I. Pomeranz and S. M. Reddy, "Test Compaction Methods for Transition Faults under Transparent-Scan", IET Computers & Digital Techniques, July 2009, pp. 315-328.

[14] I. Pomeranz, "Test Compaction by Sharing of Transparent-Scan Sequences among Logic Blocks", IEEE Trans. on VLSI Systems, April 2014, pp. 792-802.