This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2408257, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

A Multi-Cycle Test Set Based on a Two-Cycle Test Set with Constant Primary Input Vectors

Irith Pomeranz

***Abstract*— Test compaction can be achieved by using multi-cycle tests. To avoid the computationally intensive process of sequential test generation, multi-cycle tests can be generated by extending two-cycle tests. However, the scan-in state of a two-cycle test is not always effective for a multi-cycle test when the primary input vectors are held constant during the functional clock cycles of a test. This paper studies the extent of this issue by considering exhaustive two-cycle and multi-cycle test sets with constant primary input vectors for finite-state machine benchmarks. Based on the results of this study, it describes an efficient test compaction procedure that modifies selected two-cycle tests in a given test set in order to make them more effective as a source for multi-cycle tests with constant primary input vectors. Experimental results are presented to demonstrate the importance of this step to test compaction.**

***Index Terms*— Broadside tests, multi-cycle tests, test compaction, test generation, transition faults.**

I. Introduction

Broadside tests are used for detecting delay faults in standard-scan circuits. A broadside test starts with a scan-in operation. This is followed by two consecutive functional clock cycles. The first functional clock cycle is applied under a slow clock in order to allow signal-transitions that started because of the scan-in operation to subside. The second functional clock cycle is applied under a fast clock in order to activate delay faults and capture their effects. The test ends with a scan-out operation.

In a multi-cycle broadside test there may be more than two functional clock cycles between the scan operations. For at-speed test application, the first functional clock cycle is applied under a slow clock. The remaining functional clock cycles are applied under a fast clock.

Multi-cycle scan-based tests have several applications. They were shown to be useful for circuits with multiple clock domains, for test compaction, to enhance defect detection, and to avoid overtesting of delay faults [1]-[12]. Test compaction was considered in [1], [2], [9] and [11]. The use of multi-cycle tests contributes to test compaction as follows.

A functional clock cycle between the scan operations of a test is associated with a present-state and a primary input vector that together define an input pattern to the combinational logic of the circuit. With more patterns between the scan-in and scan-out operations, and by applying these patterns at-speed, the test can detect more faults. This allows the number of tests to be reduced. A reduction in the number of tests implies a reduction in the number of scan operations required for applying the test set. This contributes to a reduction in the test application time. It also reduces the test data volume since fewer scan-in and scan-out states need to be stored.

The procedure from [1] generates a compact multi-cycle test set targeting single stuck-at faults. The procedure from [2] compacts a given single-cycle test set by combining pairs of tests into multi-cycle tests targeting single stuck-at faults. The procedure from [9] generates a compact multi-cycle test set targeting transition faults starting from a two-cycle broadside test set. The procedure extends the tests into multi-cycle tests by adding subsequences of primary input vectors, which are applied in functional mode, between the scan operations of the tests. It then removes tests that become unnecessary. The procedure from [11] also starts from a given two-cycle test set. It extends a test into a multi-cycle test by adding runs of the same primary input vectors to create a special type of test that allows changes in the primary input vector to occur under a slow clock.

The advantage of starting from a single-cycle or two-cycle test set is that generating multi-cycle tests directly requires sequential test generation, which is significantly more computationally intensive than the generation of single-cycle or two-cycle tests. Specifically, for a circuit with G lines, a multi-cycle test with L clock cycles requires the test generation procedure to consider GL lines. The number of lines is G for a single-cycle test, and $2G$ for a two-cycle test. Since the worst-case computational complexity of test generation (for example, using the D-algorithm) is exponential in the number of lines, avoiding the use of a test generation procedure for multi-cycle tests is advantageous. The procedures from [2], [9] and [11] require conventional test generation for the single-cycle or two-cycle test set. They obtain multi-cycle tests without performing sequential test generation.

The multi-cycle tests generated by the procedures from [2], [9] and [11] use scan-in states and primary input vectors from a given single-cycle or two-cycle test set. Some of the scan-in states may not be effective as scan-in states of multi-cycle tests. For circuits where many of the scan-in states of a single-cycle test set are not effective as scan-in states of multi-cycle tests, the procedure from [2] produces test sets where most of the tests are single-cycle tests. As a result, it does not benefit fully from the ability of multi-cycle tests to provide test compaction. The procedure from [9] compensates for this effect by allowing arbitrary primary input subsequences, which are not based on the two-cycle test set, to be applied during the functional clock cycles between the scan operations of a multi-cycle test. The procedure from [11] also allows a multi-

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 article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2408257, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

cycle test to include several different primary input vectors.

The option of compensating for the scan-in states with arbitrary primary input subsequences does not exist when the primary input vector is held constant during all the functional clock cycles of each test. This is sometimes necessary for addressing tester limitations that prevent primary input vectors from being changed at the speed of a fast clock during a test.

This paper describes an approach for addressing this issue without performing sequential test generation. The paper consists of two (independent) parts. The first part considers exhaustive two-cycle and multi-cycle test sets with constant primary input vectors for transition faults in finite-state machine benchmarks. These test sets demonstrate the extent to which scan-in states from a two-cycle test set are effective as scan-in states for a multi-cycle test set. The second part of the paper describes an efficient test compaction procedure that generates a compact multi-cycle test set for transition faults starting from a given compact two-cycle test set. This test set can be generated by any test generation procedure for two-cycle tests (the test compaction procedure does not use exhaustive test sets, and it is applicable to larger circuits). The test compaction procedure includes a step where it modifies a scan-in state, and the corresponding primary input vector, in order to make them more suitable for a multi-cycle test. This step is applied selectively when it appears that a two-cycle test is not effective as a source for a multi-cycle test. Experimental results demonstrate the importance of this step.

The modification of scan-in states can also be applied with the test compaction procedures from [2], [9] and [11]. However, it is more important when the primary input vectors are held constant during the functional clock cycles of a test. The test compaction procedure that modifies scan-in states is expected to provide higher levels of test compaction than a sequential test generation procedure with dynamic test compaction for the following reason. A sequential test generation procedure with dynamic test compaction uses only unspecified values of a test to detect additional faults. The modification of a scan-in state allows specified values to be complemented if this leads to the detection of more faults. The possibility of complementing specified values provides a higher degree of flexibility for the procedure to detect more faults with every multi-cycle test.

To address the constraints of a test data compression method [13]-[16], the two-cycle test set can be generated under such constraints. The modification of a two-cycle test can be performed under the same constraints. Specifically, a modification can be avoided if the resulting test does not satisfy the constraints.

The application of several consecutive functional clock cycles at-speed under a multi-cycle test requires a delay fault model where the extra delay of a fault is considered explicitly [17]-[18]. In this paper, transition faults with an extra delay of a single clock cycle are considered.

The paper is organized as follows. Section II describes the exhaustive simulation experiment and its results. Section III describes the efficient test compaction procedure. Section IV presents experimental results of test compaction.

II. Exhaustive Test Sets

This section studies the extent to which the scan-in states that are effective for a two-cycle test set are also effective for a multi-cycle test set by considering exhaustive test sets for finite-state machine benchmarks.

A multi-cycle test with a scan-in state s_i, a primary input vector v_i, and l_i functional clock cycles is denoted by $< s_i, v_i, l_i >$. After s_i is scanned in, v_i is held constant during l_i functional clock cycles. The test ends with a scan-out operation. As a special case, $l_i = 2$ yields a two-cycle test.

Let S be the set of states of the circuit, and let V be its set of primary input vectors. An exhaustive L-cycle test set E_L consists of every test $< s_i, v_j, l_k >$ such that $s_i \in S$, $v_j \in V$ and $2 \leq l_k \leq L$.

For the discussion in this section, E_L is partitioned into subsets according to the scan-in state. For every $s_i \in S$, the subset denoted by $E_{L,i}$ consists of every test $< s_i, v_j, l_k >$ such that $v_j \in V$ and $2 \leq l_k \leq L$.

Let F be the set of detectable target faults (detectable transition faults in this paper). Fault simulation with fault dropping of F under $E_{L,i}$ yields a set of detected faults that is denoted by $D_{L,i}$.

A greedy set covering procedure is used for selecting a subset of scan-in states S_L such that every detectable fault in F has a multi-cycle test with at most L functional clock cycles, and a scan-in state in S_L. The set S_L contains the states that are effective as scan-in states of multi-cycle tests with at most L functional clock cycles. It allows the discussion in this section to focus on the scan-in states, and eliminates the effects of the primary input vectors and of the specific number of functional clock cycles.

The greedy set covering procedure was applied with $L =$ 2, 3, ..., 8. The results for several representative circuits are shown in Tables I-III, as follows.

Column i of every table shows the indices of all the states that are included in $S_2 \cup S_3 \cup ... \cup S_8$. Column s_i shows the binary representations of the states. The next seven columns show which scan-in states belong to every one of the sets S_L, for $2 \leq L \leq 8$. A one in row i column L indicates that $s_i \in S_L$. A zero in row i column L indicates that $s_i \notin S_L$.

In general, the tables are partitioned into three parts (some of the parts may be missing). The top part includes states that appear in every set S_L, for $2 \leq L \leq 8$. These states are effective for multi-cycle tests with all the values of L. The middle part includes states that appear in S_2, but do not appear in at least one set S_L, for $3 \leq L \leq 8$. These states are effective for two-cycle tests, but they are not necessary for multi-cycle tests with more than two cycles. The bottom part includes states that do not appear in S_2, but appear in at least one set S_L, for $3 \leq L \leq 8$. These states are not needed for two-cycle tests, but they are effective for multi-cycle tests with more than two cycles.

The circuit *bbsse* demonstrates the case where a two-cycle test set provides all the scan-in states that are needed for a multi-cycle test set. In this case, it is possible to construct a multi-cycle test set using only the scan-in states from a two-cycle test set, and it is not necessary to consider additional

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2408257, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE I

EXHAUSTIVE TEST SETS FOR $bbsse$

i	s_i	2	3	4	5	6	7	8
4	0100	1	1	1	1	1	1	1
6	0110	1	1	1	1	1	1	1
8	1000	1	1	1	1	1	1	1
13	1101	1	1	1	1	1	1	1
14	1110	1	1	1	1	1	1	1
1	0001	1	1	0	1	1	1	1
2	0010	1	1	0	0	0	0	0
3	0011	1	0	0	0	0	0	0
5	0101	1	0	0	0	0	0	0
7	0111	1	0	0	0	0	0	0
10	1010	1	0	0	0	0	0	0
11	1011	1	1	1	0	0	0	0

TABLE II

EXHAUSTIVE TEST SETS FOR $b01$

i	s_i	2	3	4	5	6	7	8
4	00100	1	1	1	1	1	1	1
6	00110	1	1	1	1	1	1	1
8	01000	1	1	1	1	1	1	1
14	01110	1	1	1	1	1	1	1
0	00000	1	1	0	0	0	0	0
2	00010	1	0	0	0	0	0	0
10	01010	1	0	0	0	0	0	0
29	11101	1	0	0	0	0	0	0
12	01100	0	0	0	0	0	1	1
13	01101	0	0	1	1	1	0	0
28	11100	0	1	0	0	0	0	0

TABLE III

EXHAUSTIVE TEST SETS FOR $ex3$

i	s_i	2	3	4	5	6	7	8
1	0001	1	1	1	1	1	1	1
0	0000	1	1	0	0	0	0	0
2	0010	1	0	1	1	1	1	1
3	0011	1	1	0	0	0	0	0
4	0100	1	1	0	0	0	0	0
8	1000	1	0	1	1	0	0	0
13	1101	1	1	0	0	0	0	0
14	1110	1	0	0	0	0	0	0
15	1111	1	0	0	0	0	0	0
7	0111	0	0	1	1	1	1	1
10	1010	0	1	0	0	1	1	1

scan-in states. This case is desirable for limiting the computational effort of the generation of a multi-cycle test set. It occurs for several other benchmark circuits.

The circuits $b01$ and $ex3$ demonstrate the case where a two-cycle test set contains some but not all the scan-in states that are effective for multi-cycle tests. In these cases, the multi-cycle test set uses some scan-in states that are not included in a two-cycle test set. This case also occurs for other benchmark circuits.

III. TEST COMPACTION PROCEDURE

This section describes an efficient test compaction procedure that accepts a compact two-cycle test set T_{init}, and a limit $L > 2$ on the number of functional clock cycles in a multi-cycle test. The procedure produces a compact multi-cycle test set T_{multi} where the tests have at most L functional clock cycles.

Based on the results of the previous section, it is likely that at least some of the scan-in states from T_{init} will be useful for constructing multi-cycle tests. The test compaction procedure described in this section includes the option of using unmodified scan-in states from T_{init}. It is also likely that new scan-in states will be needed. For this, the procedure includes the option of modifying scan-in states, and primary input vectors, from T_{init} in order to make them more suitable for multi-cycle tests.

A parameter denoted by μ determines whether or not the procedure will modify scan-in states and primary input vectors. The procedure is run first with $\mu = 0$, for which it does not modify tests. It is then run with $\mu = 1$, for which it modifies tests.

For every value of μ, the procedure increases the numbers of functional clock cycles in the multi-cycle tests it constructs gradually. This helps to avoid using unnecessarily high numbers of functional clock cycles. For a parameter denoted by λ, the procedure adds to T_{multi} λ-cycle tests. For every value of μ, the procedure iterates with $\lambda = 3$, 4, ..., L. To start a new iteration, the procedure assigns $T_{init} = T_{multi}$. It then constructs a new multi-cycle test set T_{multi} based on the new T_{init}.

In an iteration with parameters μ and λ, the procedure constructs T_{multi} by considering the tests in T_{init} one at a time. In the first iteration (with $\mu = 0$ and $\lambda = 3$), all the tests in T_{init} are two-cycle tests. In later iterations, T_{init} may contain multi-cycle tests with more than two functional clock cycles. When $\mu = 1$, T_{init} may contain a test $t_i =< s_i, v_i, l_i >$ such that $l_i > \lambda$. Such a test was added to T_{multi} when $\mu = 0$ was considered, and it was copied to T_{init} at the beginning of the iteration. The procedure includes such a test in T_{multi} without attempting to increase its number of functional clock cycles or modify its scan-in state or primary input vector. The fact that the test already has $l_i > \lambda$ functional clock cycles is taken as an indication that its scan-in state is suitable for a multi-cycle test, and there is no need to modify it. For the remaining tests in T_{init}, the procedure selects the option that detects the largest number of faults.

An iteration proceeds as described next. The discussion is illustrated by considering ISCAS-89 benchmark $s344$ with a two-cycle test set that consists of 26 tests and detects 608 transition faults. For this example, $L = 4$. The example considers the iteration where $\mu = 1$ and $\lambda = 3$. Thus, T_{init} is obtained after considering $\mu = 0$ with $\lambda = 3$ and 4. This test set contains 22 tests, with nine four-cycle tests, six three-cycle tests, and seven two-cycle tests.

The procedure reorders T_{init} such that the tests appear by order of increasing number of functional clock cycles. This gives a higher priority to adding functional clock cycles to tests with lower numbers of functional clock cycles. By increasing the numbers of functional clock cycles in as many tests as possible, the procedure increases the numbers of faults that the tests detect. This allows it to remove tests that become unnecessary, leading to test compaction.

Initially, $T_{multi} = \emptyset$ and F includes all the target faults that are detected by T_{init}.

For every test $t_i =< s_i, v_i, l_i > \in T_{init}$ such that $l_i > \lambda$, the procedure performs fault simulation with fault dropping of F under t_i, and adds t_i to T_{multi}.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2408257, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE IV
EXAMPLE OF UNMODIFIED TESTS

i	t_i	det
13	$< 001101101001011, 011010101, 4 >$	271
14	$< 001100010001111, 001111001, 4 >$	293
15	$< 000010111111001, 010101111, 4 >$	348
16	$< 000101000001010, 011011010, 4 >$	376
17	$< 011011011000101, 010010100, 4 >$	413
18	$< 001100110001101, 011001010, 4 >$	418
19	$< 010101111000010, 000101100, 4 >$	436
20	$< 110100100010111, 000110111, 4 >$	491
21	$< 000000000101111, 010101000, 4 >$	499

TABLE V
EXAMPLE OF A MODIFIED TEST

test	det
$t_5 =< 011010111001110, 000100010, 2 >$	$n_5 = 3$
$t_{5,1} =< 011010111001110, 000100010, 3 >$	$n_{5,1} = 3$
$t_{5,2} =< 110010110001111, 000100010, 3 >$	$n_{5,2} = 7$

In the example of $s344$, the tests shown in Table IV are added to T_{multi} without being modified. The index i of a test is its index in T_{init}. The number of transition faults that T_{multi} detects after every test is simulated and added to it is shown in the last column of Table IV.

Next, for every test $t_i =< s_i, v_i, l_i > \in T_{init}$ such that $l_i \leq \lambda$, the procedure performs the following steps.

It simulates F under t_i and finds the faults that are detected by t_i. The set of faults is denoted by D_i, and the number of faults is denoted by n_i. If $n_i = 0$, t_i is not needed for detecting any faults from F. In this case, the procedure does not consider t_i further. By not including in T_{multi} any test that is based on t_i, the procedure achieves test compaction.

If $n_i > 0$, the procedure considers the following options. Table V shows the options considered for $s344$ based on $t_5 \in T_{init}$. When this test is considered, T_{multi} detects 587 transition faults.

The first option is to include t_i as it is in T_{multi}. The test will detect n_i faults from F if it is added to T_{multi}.

The second option is to change the number of clock cycles in t_i to λ and include in T_{multi} the test $t_{i,1} =< s_i, v_i, \lambda >$. The procedure simulates D_i under $t_{i,1}$. If all the faults in D_i are detected, it also simulates $F - D_i$ under $t_{i,1}$. The set of detected faults (out of the simulated faults) is denoted by $D_{i,1}$. The procedure assigns to $n_{i,1}$ the number of faults in $D_{i,1}$.

The third option is represented by a test $t_{i,2}$ that detects $n_{i,2}$ faults from F. This option is considered only if $\mu = 1$. If $\mu = 0$, the procedure assigns $n_{i,2} = 0$ without computing $t_{i,2}$, and it will not select $t_{i,2}$. With $\mu = 1$, the computation of $t_{i,2}$ consists of modifying the scan-in state and primary input vector of $t_{i,1} =< s_i, v_i, \lambda >$ in order to increase the number of faults that the test detects.

To obtain a modified test without sequential test generation, the procedure complements bits of the scan-in state and primary input vector one at a time in a random order. A complementation is accepted if the number of detected faults does not decrease.

Earlier procedures that used single bit complementation show the effectiveness of this approach [19]-[21]. It is suitable for the test compaction procedure described in this paper since it allows all the faults in F to be targeted simultaneously without considering specific faults and without performing test generation. In addition, $t_{i,2}$ is required to detect the faults in D_i. Therefore, it is likely that its scan-in state will be close to s_i. Such a scan-in state can be obtained by complementing single bits of s_i.

The primary input vector is considered together with the scan-in state to ensure that it matches the modified scan-in state. The computation of $t_{i,2}$ and $n_{i,2}$ proceeds as follows.

Initially, $t_{i,2} =< s_i, v_i, \lambda >$ and $n_{i,2} = n_{i,1}$. The procedure considers the bits of s_i and v_i one at a time in a random order. When bit b is considered, the procedure complements its value to obtain the test $t_{i,2,b}$. It then simulates D_i under $t_{i,2,b}$. If all the faults in D_i are detected, it also simulates $F - D_i$ under $t_{i,2,b}$. Let the set of detected faults (out of the simulated faults) be $D_{i,2,b}$. The procedure assigns $n_{i,2,b} = |D_{i,2,b}|$. If $n_{i,2,b} \geq n_{i,2}$, the procedure assigns $t_{i,2} = t_{i,2,b}$ and $n_{i,2} = n_{i,2,b}$.

With this process, if $t_{i,2}$ does not detect all the faults in D_i, a complementation is accepted if at least the same number of faults from D_i are detected. This helps in modifying $t_{i,2}$ such that it would detect all the faults from D_i. Without detecting all the faults from D_i, $t_{i,2}$ will not be selected.

The procedure iterates through all the bits of $t_{i,2}$ for N_B iterations, where N_B is a constant.

The procedure selects one of the three options, t_i, $t_{i,1}$ or $t_{i,2}$, based on the numbers of detected faults using the following criteria. The selected test is denoted by $t_{i,sel}$.

If $n_i \geq n_{i,1}$ and $n_i \geq n_{i,2}$, the procedure assigns $t_{i,sel} = t_i$. This implies that t_i is selected if $n_i = n_{i,1} = n_{i,2}$.

Otherwise, if $n_{i,1} \geq n_i$ and $n_{i,1} \geq n_{i,2}$, the procedure assigns $t_{i,sel} = t_{i,1}$.

Otherwise, if $n_{i,2} \geq n_i$ and $n_{i,2} \geq n_{i,1}$, the procedure assigns $t_{i,sel} = t_{i,2}$.

In the case of $s344$ with the options shown in Table V, the procedure selects to add $t_{5,2}$ to T_{multi}.

The selected test $t_{i,sel}$ is added to T_{multi}. Fault simulation with fault dropping of F under $t_{i,sel}$ removes from F the faults that it detects.

After obtaining a test set T_{multi} that detects all the faults in F, the procedure applies forward-looking reverse order fault simulation in order to remove unnecessary tests.

The test compaction procedure is summarized next.

Procedure 1: Test compaction

1) Let T_{init} be a given two-cycle test set. Include in F_{targ} all the target faults that T_{init} detects. Assign $T_{multi} = T_{init}$.
2) For $\mu = 0$, 1 and for $\lambda = 3$, 4, ..., L:
   a) Assign $T_{init} = T_{multi}$.
   b) Reorder T_{init} such that the tests appear by order of increasing number of functional clock cycles.
   c) Assign $T_{multi} = \emptyset$ and $F = F_{targ}$.
   d) For every test $t_i =< s_i, v_i, l_i > \in T_{init}$ such that $l_i > \lambda$, perform fault simulation with fault dropping of F under t_i, and add t_i to T_{multi}.
   e) For every test $t_i =< s_i, v_i, l_i > \in T_{init}$ such that $l_i \leq \lambda$, call Procedure $add\_test(\mu, \lambda, F, t_i)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2408257, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

f) Apply forward-looking reverse order fault simulation to T_{multi}.

Procedure $add\_test(\mu, \lambda, F, t_i)$**:**

1) Simulate F under t_i and find the set D_i and the number n_i of detected faults. If $n_i = 0$, stop.
2) Define $t_{i,1} =< s_i, v_i, \lambda >$. Simulate D_i under $t_{i,1}$. If all the faults are detected, simulate $F - D_i$ under $t_{i,1}$. Let $n_{i,1}$ be the number of detected faults.
3) If $\mu = 0$, assign $n_{i,2} = 0$. Else:
   a) Assign $t_{i,2} = t_{i,1}$, $n_{i,2} = n_{i,1}$, and modify $t_{i,2}$ to increase $n_{i,2}$.
4) If $n_i \geq n_{i,1}$ and $n_i \geq n_{i,2}$, assign $t_{i,sel} = t_i$. Else, if $n_{i,1} \geq n_i$ and $n_{i,1} \geq n_{i,2}$, assign $t_{i,sel} = t_{i,1}$. Else, if $n_{i,2} \geq n_i$ and $n_{i,2} \geq n_{i,1}$, assign $t_{i,sel} = t_{i,2}$.
5) Add $t_{i,sel}$ to T_{multi}. Perform fault simulation with fault dropping of F under $t_{i,sel}$.

The worst-case computational complexity of the procedure is determined as follows. Let the number of state variables and primary inputs be K. For a given bound L on the number of functional clock cycles, the procedure performs $2(L-2)$ iterations with $\mu = 0$ or 1, and $\lambda = 3, 4, ..., L$.

In every iteration, the procedure considers all the tests in T_{init}. For every test $t_i \in T_{init}$, the procedure considers at most two options without modifying the scan-in state or primary input vector. With modification, the procedure considers $O(K)$ options iteratively. The number of iterations is equal to the constant N_B. Overall, in the worst case, the procedure performs fault simulation of F under $O(|T_{init}|K)$ tests in every iteration.

Considering all the iterations, the procedure simulates F under $O(L|T_{init}|K)$ tests.

The number of clock cycles required for applying a test set $T = \{t_0, t_1, ..., t_{m-1}\}$ to a circuit with k state variables that are included in a single scan chain is the following. Let $t_i =< s_i, v_i, l_i >$, for $0 \leq i < m$. The number of scan shift cycles required for m tests is $(m+1)k$. The number of functional clock cycles for t_i is l_i. Considering all the tests, the number of functional clock cycles is $\sum_{i=0}^{m-1} l_i$. The total number of clock cycles required for test application is $(m+1)k + \sum_{i=0}^{m-1} l_i$.

IV. Experimental Results

The test compaction procedure was applied to benchmark circuits as described in this section.

The implementation of the procedure does not use any commercial tools. The fault simulation process that it is based on considers one fault and one test at a time.

No resynthesis was applied to the benchmark circuits that may affect their sets of faults or the numbers of tests required for detecting them.

The test compaction procedure was applied to compact two-cycle broadside test sets with constant primary input vectors that were generated for transition faults in benchmark circuits. Only benchmark circuits where the transition fault coverage is at least 70% were considered. For other circuits, the requirement to use broadside tests with constant primary input vectors causes high fault coverage loss.

The procedure was applied with the following parameter values. The procedure generates multi-cycle tests with at most $L = 8$ functional clock cycles. Experimental results showed that a higher value of L typically does not increase significantly the level of test compaction that can be achieved.

The limit on the number of times the procedure considers all the bits of a test for complementation is $N_B = 4$. Experimental results indicated that the first tests that the procedure considers in an iteration can benefit from a higher limit. However, $N_B = 4$ is sufficient for obtaining effective tests.

The results are shown in Tables VI and VII as follows. There are several rows for every circuit. The row with $\mu = 0$ and $\lambda = 2$ corresponds to the initial two-cycle test set. It represents the level of test compaction that can be achieved by a two-cycle test set with constant primary input vectors.

The row with $\mu = 0$ and $\lambda > 2$ corresponds to the final test set obtained with $\mu = 0$. This test set may be associated with $\lambda < 8$ if increasing λ further does not reduce the number of clock cycles required for test application. The test set shows the level of test compaction that can be achieved by using multi-cycle tests with scan-in states and primary input vectors from a two-cycle test set.

The next rows correspond to test sets that are obtained with $\mu = 1$. There is a row for every 10% reduction in the number of clock cycles, and for the final test set obtained with $\mu = 1$. These test sets show the levels of test compaction that can be achieved by constructing a multi-cycle test set that is based on a two-cycle test set, while modifying scan-in states and primary input vectors.

For each test set, columns μ and λ show the values of the respective parameters.

Column $tests$ subcolumn tot shows the number of tests in the test set. Subcolumn $frac$ shows the number of tests as a fraction of the number of tests in the initial two-cycle test set. Subcolumn $2cyc$ shows the fraction of two-cycle tests in the test set.

Column $select$ shows the fractions of tests that are added to T_{multi} based on the options used for selecting them. Subcolumn $opt0$ corresponds to tests that are added unmodified. Subcolumn $opt1$ corresponds to tests that are added with modified lengths but the same scan-in states and primary input vectors. Subcolumn $opt2$ corresponds to tests that are added with modified lengths, scan-in states and primary input vectors.

Column $func$ subcolumn max shows the maximum number of functional clock cycles in a test. Subcolumn ave shows the average number of functional clock cycles in a test.

Column $cycles$ subcolumn tot shows the number of clock cycles required for applying the test set. Subcolumn $frac$ shows the number of cycles as a fraction of the number of cycles required for the initial two-cycle test set.

Column $f.c.$ shows the transition fault coverage, which is not affected by test compaction.

Column $n.time$ shows the following run time information. Let rt_0 be the run time for transition *fault simulation* of the initial two-cycle test set. The value of rt_0 in seconds on a Linux machine with a 3GHz processor is shown in parentheses in the row for the initial test set. This run time is unnecessarily

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2408257, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE VI

EXPERIMENTAL RESULTS FOR CIRCUITS WITH $C_{0,fin}/C_{0,2} \geq 0.9$

			tests			select			func		cycles			
circuit	μ	λ	tot	frac	2cyc	opt0	opt1	opt2	max	ave	tot	frac	f.c.	n.time
usb_phy	0	2	52	1.00	1.00	-	-	-	2	2.00	5298	1.00	88.96	(1.73)
usb_phy	1	3	46	0.88	0.74	0.74	0.00	0.26	3	2.26	4710	0.89	88.96	159.89
usb_phy	1	4	40	0.77	0.65	0.80	0.00	0.20	4	2.55	4120	0.78	88.96	285.53
s38417	0	2	648	1.00	1.00	-	-	-	2	2.00	1063060	1.00	97.11	(19650.60)
s38417	0	4	644	0.99	0.90	0.93	0.07	0.00	4	2.16	1056614	0.99	97.11	8.59
s38417	1	3	518	0.80	0.62	0.69	0.00	0.31	4	2.46	850356	0.80	97.11	324.92
s5378	0	2	180	1.00	1.00	-	-	-	2	2.00	32759	1.00	77.78	(239.08)
s5378	0	3	177	0.98	0.94	0.94	0.06	0.00	3	2.06	32226	0.98	77.78	2.58
s5378	1	4	153	0.85	0.46	0.71	0.00	0.29	4	2.83	27999	0.85	77.78	147.55
s5378	1	8	140	0.78	0.42	0.84	0.00	0.16	8	4.52	25872	0.79	77.78	468.78
s9234	0	2	355	1.00	1.00	-	-	-	2	2.00	81878	1.00	76.54	(1709.65)
s9234	0	5	348	0.98	0.84	0.95	0.05	0.00	5	2.33	80382	0.98	76.54	9.66
s9234	1	4	303	0.85	0.65	0.80	0.00	0.20	5	2.67	70122	0.86	76.54	139.40
s9234	1	8	277	0.78	0.69	0.86	0.00	0.14	8	3.55	64368	0.79	76.54	411.67
b20	0	2	306	1.00	1.00	-	-	-	2	2.00	152270	1.00	79.65	(5896.20)
b20	0	8	299	0.98	0.47	0.96	0.04	0.00	8	3.19	149154	0.98	79.65	19.03
b20	1	3	260	0.85	0.09	0.22	0.00	0.78	8	3.67	129888	0.85	79.65	121.80
b20	1	5	228	0.75	0.09	0.66	0.00	0.34	8	4.25	114094	0.75	79.65	604.83
b20	1	8	205	0.67	0.08	0.75	0.01	0.24	8	5.23	102837	0.68	79.65	1161.86
sasc	0	2	47	1.00	1.00	-	-	-	2	2.00	5710	1.00	85.48	(8.75)
sasc	0	3	46	0.98	0.85	0.85	0.15	0.00	3	2.15	5598	0.98	85.48	4.82
sasc	1	3	37	0.79	0.54	0.59	0.00	0.41	3	2.46	4537	0.79	85.48	66.30
sasc	1	5	32	0.68	0.53	0.70	0.00	0.30	5	3.22	3964	0.69	85.48	128.42
sasc	1	8	31	0.66	0.55	0.65	0.00	0.35	8	4.39	3880	0.68	85.48	315.24
wb_dma	0	2	175	1.00	1.00	-	-	-	2	2.00	92398	1.00	75.04	(1061.76)
wb_dma	0	4	172	0.98	0.90	0.95	0.05	0.00	4	2.15	90848	0.98	75.04	5.40
wb_dma	1	4	146	0.83	0.56	0.70	0.01	0.29	4	2.78	77287	0.84	75.04	605.10
wb_dma	1	7	129	0.74	0.50	0.78	0.00	0.22	7	3.98	68503	0.74	75.04	1583.86
wb_dma	1	8	125	0.71	0.51	0.78	0.00	0.22	8	4.39	66447	0.72	75.04	2006.81
s13207	0	2	349	1.00	1.00	-	-	-	2	2.00	234848	1.00	79.98	(1800.08)
s13207	0	5	339	0.97	0.80	0.93	0.07	0.00	5	2.40	228272	0.97	79.98	5.79
s13207	1	3	285	0.82	0.49	0.63	0.00	0.37	5	2.74	192114	0.82	79.98	156.73
s13207	1	5	234	0.67	0.18	0.58	0.00	0.42	5	4.08	158170	0.67	79.98	948.84
s13207	1	8	207	0.59	0.18	0.58	0.00	0.42	8	6.17	140429	0.60	79.98	2689.29
spi	0	2	865	1.00	1.00	-	-	-	2	2.00	200044	1.00	82.59	(1455.29)
spi	0	7	838	0.97	0.84	0.96	0.04	0.00	7	2.50	194225	0.97	82.59	27.29
spi	1	5	749	0.87	0.35	0.65	0.00	0.35	7	3.84	174625	0.87	82.59	2196.32
tv80	0	2	658	1.00	1.00	-	-	-	2	2.00	237897	1.00	82.02	(3192.37)
tv80	0	7	632	0.96	0.74	0.97	0.03	0.00	7	2.66	228927	0.96	82.02	16.28
tv80	1	5	545	0.83	0.41	0.73	0.00	0.27	7	3.58	197965	0.83	82.02	535.65
tv80	1	8	496	0.75	0.40	0.78	0.00	0.22	8	4.68	180745	0.76	82.02	1270.26
s1423	0	2	69	1.00	1.00	-	-	-	2	2.00	5318	1.00	73.96	(4.92)
s1423	0	8	62	0.90	0.29	0.94	0.06	0.00	8	4.32	4930	0.93	73.96	18.58
s1423	1	4	54	0.78	0.07	0.59	0.00	0.41	8	4.83	4331	0.81	73.96	50.66
s1423	1	7	43	0.62	0.09	0.53	0.00	0.47	8	6.33	3528	0.66	73.96	169.18
s1423	1	8	42	0.61	0.10	0.38	0.00	0.62	8	7.19	3484	0.66	73.96	282.58
b14	0	2	207	1.00	1.00	-	-	-	2	2.00	51790	1.00	72.04	(698.38)
b14	0	7	192	0.93	0.57	0.89	0.11	0.00	7	3.04	48254	0.93	72.04	14.33
b14	1	6	170	0.82	0.15	0.91	0.00	0.09	7	4.06	42927	0.83	72.04	374.23
b14	1	8	164	0.79	0.15	0.88	0.00	0.12	8	4.51	41495	0.80	72.04	518.10
systemcaes	0	2	202	1.00	1.00	-	-	-	2	2.00	136414	1.00	88.74	(2342.70)
systemcaes	0	8	180	0.89	0.21	0.83	0.17	0.00	8	5.51	122262	0.90	88.74	33.45
systemcaes	1	6	150	0.74	0.03	0.39	0.00	0.61	8	6.51	102146	0.75	88.74	479.29
systemcaes	1	8	119	0.59	0.04	0.38	0.00	0.62	8	7.39	81279	0.60	88.74	4033.61

high because the fault simulation procedure does not use any of the commonly used speedup techniques of parallel simulation. In the rows for the multi-cycle test sets, the run time for producing the multi-cycle test set is divided by rt_0 in order to provide an indication of the computational effort in terms of transition fault simulation time. This is referred to as the normalized run time. The normalized run time is cumulative and includes all the iterations of the procedure up to the one reported.

The circuits are arranged as follows. Let $C_{0,2}$ be the number of clock cycles required for the initial two-cycle test set. Let $C_{0,fin}$ be the number of clock cycles required for the final multi-cycle test set obtained with $\mu = 0$. The circuits are arranged by decreasing value of $C_{0,fin}/C_{0,2}$. The test compaction procedure is designed for circuits where the ratio $C_{0,fin}/C_{0,2}$ is higher, and the scan-in states of a two-cycle test set need to be modified for a multi-cycle test set. Such circuits appear earlier in Tables VI and VII.

The following points can be seen from Tables VI and VII. Without modifying scan-in states ($\mu = 0$), there are circuits where the test compaction procedure is able to reduce the number of clock cycles significantly. However, there are also circuits where the reductions are small or moderate. For usb_phy, no reduction is obtained when $\mu = 0$.

In most of the cases, modifying scan-in states allows the test compaction procedure to reduce the number of clock cycles significantly. This includes circuits where the reduction in the number of clock cycles without modifying scan-in states is small.

The procedure typically selects tests with modified scan-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2408257, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE VII
EXPERIMENTAL RESULTS FOR CIRCUITS WITH $C_{0,fin}/C_{0,2} < 0.9$

			tests			select			func		cycles			
circuit	μ	λ	tot	frac	2cyc	opt0	opt1	opt2	max	ave	tot	frac	f.c.	n.time
b15	0	2	488	1.00	1.00	-	-	-	2	2.00	219559	1.00	81.12	(12295.59)
b15	0	8	422	0.86	0.17	0.84	0.16	0.00	8	5.13	191247	0.87	81.12	31.86
b15	1	5	369	0.76	0.02	0.51	0.01	0.48	8	5.85	167549	0.76	81.12	269.02
simple_spi	0	2	78	1.00	1.00	-	-	-	2	2.00	10505	1.00	76.70	(23.38)
simple_spi	0	8	65	0.83	0.28	0.85	0.15	0.00	8	4.74	8954	0.85	76.70	22.61
simple_spi	1	7	57	0.73	0.23	0.62	0.00	0.38	8	5.68	7922	0.75	76.70	190.77
simple_spi	1	8	54	0.69	0.24	0.63	0.00	0.37	8	6.06	7532	0.72	76.70	299.38
b08	0	2	45	1.00	1.00	-	-	-	2	2.00	1056	1.00	73.22	(0.15)
b08	0	8	34	0.76	0.29	0.83	0.17	0.00	8	3.97	870	0.82	73.22	31.53
b08	1	5	28	0.62	0.04	0.42	0.00	0.58	8	5.11	752	0.71	73.22	118.93
b08	1	6	26	0.58	0.00	0.48	0.14	0.38	8	5.85	719	0.68	73.22	164.27
b05	0	2	104	1.00	1.00	-	-	-	2	2.00	3778	1.00	76.34	(4.23)
b05	0	8	73	0.70	0.40	0.81	0.19	0.00	8	4.64	2855	0.76	76.34	20.17
b05	1	4	62	0.60	0.13	0.67	0.00	0.33	8	5.29	2470	0.65	76.34	28.48
b05	1	8	48	0.46	0.06	0.45	0.06	0.49	8	6.94	1999	0.53	76.34	87.81
b07	0	2	64	1.00	1.00	-	-	-	2	2.00	3443	1.00	70.30	(1.11)
b07	0	8	42	0.66	0.33	0.90	0.10	0.00	8	4.21	2370	0.69	70.30	19.33
b07	1	7	35	0.55	0.29	0.81	0.00	0.19	8	5.00	2011	0.58	70.30	79.53
b07	1	8	34	0.53	0.29	0.71	0.06	0.24	8	5.29	1965	0.57	70.30	128.20
s35932	0	2	30	1.00	1.00	-	-	-	2	2.00	53628	1.00	71.80	(11382.08)
s35932	0	8	20	0.67	0.10	0.50	0.50	0.00	8	6.60	36420	0.68	71.80	14.42
s35932	1	6	14	0.47	0.00	0.00	0.00	1.00	8	7.43	26024	0.49	71.80	22.63
aes_core	0	2	311	1.00	1.00	-	-	-	2	2.00	165982	1.00	96.18	(33437.81)
aes_core	0	8	204	0.66	0.00	0.34	0.66	0.00	8	7.57	110195	0.66	96.18	45.84
aes_core	1	7	173	0.56	0.00	0.21	0.00	0.79	8	7.76	93562	0.56	96.18	119.20
b11	0	2	73	1.00	1.00	-	-	-	2	2.00	2366	1.00	81.58	(0.94)
b11	0	8	42	0.58	0.21	0.82	0.18	0.00	8	5.21	1509	0.64	81.58	29.16
b11	1	8	35	0.48	0.06	0.51	0.09	0.40	8	7.23	1333	0.56	81.58	133.51
systemcdes	0	2	91	1.00	1.00	-	-	-	2	2.00	17662	1.00	96.09	(91.48)
systemcdes	0	8	55	0.60	0.05	0.40	0.60	0.00	8	7.25	11039	0.63	96.09	40.51
systemcdes	1	7	45	0.49	0.00	0.31	0.00	0.69	8	7.71	9087	0.51	96.09	82.53
systemcdes	1	8	42	0.46	0.00	0.47	0.16	0.37	8	8.00	8506	0.48	96.09	997.70

in states and primary input vectors only in a small fraction of the cases. Nevertheless, these tests are important for test compaction.

The run time of the test compaction procedure depends on the values of the parameters L and N_B. At the cost of a lower reduction in the number of clock cycles, it is possible to reduce the run time of the procedure by reducing the values of these parameters. The discussion below focuses on the value of L.

By increasing the value of λ gradually, the procedure matches the numbers of clock cycles in the multi-cycle tests it produces to the circuit. However, this also implies that the procedure may consider the same test repeatedly with every value of λ in order to transform it into a three-cycle test, then a four-cycle test, and so on. A test that can be replaced with an L-cycle test will be considered only once if $\lambda = L$ is considered first, and the value of λ is reduced gradually. This is possible if L is known in advance. After the test is replaced with an L-cycle test, it will not be considered again with lower values of λ. Considering $\lambda < L$ is important for a test that cannot be replaced with an L-cycle test, but can be replaced with an l_i-cycle test for $2 < l_i < L$.

In addition, the procedure attempts to modify a test t_i for a given value of λ if $l_i \leq \lambda$. In order to reduce the run time, it is possible to consider a test for modification only if $l_i < \lambda$. In this case, tests with $l_i = \lambda$ will not be modified.

To check the effects of these observations on the run time and the number of clock cycles, the test compaction procedure was applied to several circuits with $L = 5$, and decreasing values of λ. In addition, the procedure attempts to modify a test t_i only if $l_i < \lambda$. The value $L = 5$ was selected based on the results in Tables VI and VII, which show that this value is effective for most of the circuits considered. Table VIII shows the results of the last iteration where a reduction in the number of clock cycles was achieved.

Compared with the results in Tables VI and VII, Table VIII shows reductions in the numbers of clock cycles with significantly reduced normalized run times for most of the circuits. In several cases, the last iteration that reduces the number of clock cycles has $\lambda = 5$, and lower values of λ are not needed.

Finally, it is interesting to check the effect of modifying a test set, in order to increase its effectiveness as a source for multi-cycle tests, on the design-for-testability procedure from [12]. This procedure increases the transition fault coverage by holding the values of selected state variables constant during multi-cycle tests. The procedure increases the number of tests while increasing the fault coverage.

The procedure from [12] was applied starting from two test sets: a test set where T_{init} is followed by T_{multi}, and a test set where T_{multi} is followed by T_{init}. With $T_{multi} \neq T_{init}$, the procedure from [12] may achieve a different transition fault coverage depending on the test set given to it. To compare the number of tests for the same transition fault coverage, both test sets are used. The procedure from [12] was modified to prefer tests from the first test set given to it. It uses tests based on the second test set only as necessary to increase the fault coverage further.

The results for several circuits are shown in Table IX. For every case, Table IX shows the number of tests and the transition fault coverage that the procedure from [12] achieves

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2015.2408257, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE VIII

EXPERIMENTAL RESULTS WITH REDUCED PARAMETER VALUES

			tests			select			func		cycles			
circuit	μ	λ	tot	frac	2cyc	opt0	opt1	opt2	max	ave	tot	frac	f.c.	n.time
usb_phy	1	3	42	0.81	0.62	0.84	0.00	0.16	5	2.86	4334	0.82	88.96	280.97
s5378	1	3	156	0.87	0.68	0.90	0.00	0.10	5	2.75	28532	0.87	77.78	89.24
sasc	1	3	37	0.79	0.51	0.86	0.00	0.14	5	3.30	4568	0.80	85.48	79.99
wb_dma	1	3	153	0.87	0.73	0.95	0.00	0.05	5	2.71	80957	0.88	75.04	347.35
s13207	1	3	258	0.74	0.44	0.90	0.00	0.10	5	3.49	174172	0.74	79.98	143.15
s1423	1	3	52	0.75	0.42	0.92	0.00	0.08	5	3.65	4112	0.77	73.96	40.47
b14	1	3	176	0.85	0.44	0.78	0.00	0.22	5	3.28	44296	0.86	72.04	165.89
simple_spi	1	3	67	0.86	0.55	0.93	0.00	0.07	5	3.25	9126	0.87	76.70	150.19
b08	1	5	32	0.71	0.50	0.74	0.00	0.26	5	3.50	805	0.76	73.22	12.42
b05	1	4	68	0.65	0.37	0.93	0.00	0.07	5	3.85	2608	0.69	76.34	22.04
b07	1	3	43	0.67	0.49	0.95	0.00	0.05	5	3.42	2391	0.69	70.30	27.25
b11	1	5	46	0.63	0.41	0.69	0.00	0.31	5	3.74	1582	0.67	81.58	20.14

TABLE IX

RESULTS OF THE PROCEDURE FROM [12]

	$T_{init} + T_{multi}$		$T_{multi} + T_{init}$	
circuit	tests	f.c.	tests	f.c.
usb_phy	81	93.01	76	93.01
s5378	227	80.36	200	80.36
sasc	75	92.00	72	92.00
wb_dma	244	77.92	209	77.92
s1423	85	80.64	77	80.64
b14	351	82.70	346	82.70
simple_spi	96	85.29	95	85.29
b08	51	86.73	44	86.73
b05	104	88.41	97	88.41
b07	71	89.15	67	89.15
b11	65	89.02	62	89.02

starting from the test set given to it.

From Table IX it can be observed that the use of T_{multi} as the first test set results in a lower number of tests for all the circuits considered. Thus, this test set is also effective for the computation of a multi-cycle test set with an increased transition fault coverage.

V. CONCLUDING REMARKS

The first part of this paper studied the extent to which scan-in states of a compact two-cycle test set are suitable as scan-in states of a compact multi-cycle test set. This issue is important for avoiding sequential test generation when constructing multi-cycle tests. The study was performed by considering exhaustive two-cycle and multi-cycle test sets for finite-state machine benchmarks. Based on the results of this study, the second part of the paper described an efficient test compaction procedure that uses two-cycle tests in a given test set as a basis for the computation of multi-cycle tests. The procedure includes the option of modifying scan-in states, and the corresponding primary input vectors, in order to make them more suitable for multi-cycle tests. This step was applied selectively to tests whose numbers of functional clock cycles were lower than a target. Experimental results were presented to demonstrate the importance of this step to the ability to achieve test compaction using multi-cycle tests without performing sequential test generation.

REFERENCES

[1] 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.

[2] 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.

[3] P. C. Maxwell, R. C. Aitken, K. R. Kollitz and A. C. Brown, "IDDQ and AC Scan: The War Against Unmodelled Defects", in Proc. Intl. Test Conf., 1996, pp. 250-258.

[4] J. Rearick, "Too Much Delay Fault Coverage is a Bad Thing", in Proc. Intl. Test Conf., 2001, pp. 624-633.

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

[6] 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.

[7] I. Park and E. J. McCluskey, "Launch-on-Shift-Capture Transition Tests", in Proc. Intl. Test Conf., 2008, pp. 1-9.

[8] E. K. Moghaddam, J. Rajski, S. M. Reddy and M. Kassab, "At-Speed Scan Test with Low Switching Activity", in Proc. VLSI Test Symp., 2010, pp. 177-182.

[9] I. Pomeranz, "Generation of Multi-Cycle Broadside Tests", IEEE Trans. on Computer-Aided Design, Aug. 2011, pp. 1253-1257.

[10] I. Pomeranz, "Multi-Cycle Tests with Constant Primary Input Vectors for Increased Fault Coverage", IEEE Trans. on Computer-Aided Design, Sep. 2012, pp. 1428-1438.

[11] I. Pomeranz, "Multi-Cycle Broadside Tests with Runs of Constant Primary Input Vectors", IET Computers & Digital Techniques, March 2014, pp. 90-96.

[12] I. Pomeranz, "Design-for-Testability for Multi-Cycle Broadside Tests By Holding of State Variables", ACM Trans. on Design Automation, Vol. 19, No. 2, March 2014, pp. 19:1-19:20.

[13] K. Lee, J. Chen and C. Huang, "Using a Single Input to Support Multiple Scan Chains", in Proc. Intl. Conf. Computer-Aided Design, 1998, pp. 74-78.

[14] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller and B. Koenemann, "OPMISR: The Foundation for Compressed ATPG Vectors", in Proc. Intl. Test Conf., 2001, pp. 748-757.

[15] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K.-H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide and J. Qian, "Embedded Deterministic Test for Low Cost Manufacturing Test", in Proc. Intl. Test Conf., 2002, pp. 301-310.

[16] N. A. Touba, "Survey of Test Vector Compression Techniques", IEEE Design & Test, Apr. 2006, pp. 294-303.

[17] I. Pomeranz and S. M. Reddy, "At-Speed Delay Testing of Synchronous Sequential Circuits", in Proc. Design Autom. Conf., 1992, pp. 177-181.

[18] K.-T. Cheng, "Transition Fault Testing for Sequential Circuits", IEEE Trans. on Computer-Aided Design, 1993, pp. 1971-1983.

[19] T. J. Snethen, "Simulation-Oriented Fault Test Generator", in Proc. Design Autom. Conf., 1977, pp. 88-93.

[20] P. Girard, C. Landrault, V. Moreda and S. Pravossoudovitch, "An Optimized BIST Test Pattern Generator for Delay Testing", in Proc. VLSI Test Symp., 1997, pp. 94-100.

[21] K.-H. Tsai, J. Rajski and M. Marek-Sadowska, "Scan Encoded Test Pattern Generation for BIST", in Proc. Intl. Test Conf., 1997, pp. 548-556.