Multicycle Broadside and Skewed-Load Tests for Test Compaction

Irith Pomeranz

***Abstract*— This paper describes a test compaction procedure that combines the advantages of using multicycle tests for test compaction with the advantages of using both broadside and skewed-load tests for increasing the fault coverage and achieving test compaction. The procedure is the first to combine these two concepts in a single procedure. The combination is made possible by a definition of a multicycle skewed-load test that is suggested in this paper, and complements the definition of a multicycle broadside test. Experimental results demonstrate the effectiveness of multicycle broadside and skewed-load tests in achieving test compaction for benchmark circuits.**

***Index Terms*— broadside tests, skewed-load tests, multicycle tests, test compaction, transition faults.**

I. Introduction

An l-cycle scan-based test has $l \geq 1$ clock cycles between a scan-in and a scan-out operation [1]-[17]. Typically, these clock cycles are functional capture cycles. Functional capture cycles allow fault effects that reach the next-state variables to be captured in the flip-flops. Such fault effects can be either propagated further through additional functional capture cycles, or scanned out if the test does not have additional functional capture cycles. Each functional capture cycle translates into an input pattern that is applied to the combinational logic of the circuit. With more functional capture cycles, a test can potentially detect more faults. This explains why multicycle tests contribute to test compaction.

Three exceptions to the exclusive use of functional capture cycles between scan operations are the following. (1) In a two-cycle skewed-load test, the first clock cycle after the scan-in operation is a scan shift cycle [1]. (2) The approach called transparent-scan uses arbitrary sequences of scan shift and functional capture cycles [7]. (3) The approach described in [9] combines broadside and skewed-load tests into three-cycle tests that include scan shift and functional capture cycles between the scan-in and scan-out operations.

Skewed-load tests require the scan enable input to change at the circuit speed when a scan shift cycle is followed by a fast functional capture cycle. This requirement is addressed in [18]-[19] by generating fast scan enable inputs locally from the global scan enable input.

A two-cycle broadside test has two functional capture cycles between its scan operations. By extension, an l-cycle broadside test has $l \geq 2$ functional capture cycles between scan operations. Between the scan operations of a skewed-load test, a scan shift cycle is followed by a functional capture cycle. A definition of an l-cycle skewed-load test is suggested in this paper. Under this definition, between scan operations, an l-cycle skewed-load test has $l - 1$ scan shift cycles followed by a functional capture cycle. The rationale for this definition is discussed later.

Even with two-cycle tests, the use of both broadside and skewed-load tests has two advantages. (1) Considering transition faults (or delay faults in general), some faults are only detectable by broadside tests and some faults are only detectable by skewed-load tests. Therefore, using both test types increases the transition fault coverage [1]-[2]. (2) The use of both test types contributes to test compaction because of the flexibility to select a test that detects more faults. A test compaction procedure for transition faults that selects between a broadside and skewed-load test the one that detects more faults is described in [11].

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

TABLE I
Format of a Multicycle Test

symbol	meaning
s_i	scan-in state
v_i	primary input vector
l_i	number of clock cycles between scan operations
e_i	test type (0 - broadside, 1 - skewed load)
c_i	scan-in values

The goal of this paper is to combine the advantages of using multicycle tests for test compaction with the advantages of using both broadside and skewed-load tests for increasing the fault coverage and achieving test compaction. The test compaction procedure described in this paper is the first to combine these two concepts in a single procedure by suggesting a definition of multicycle skewed-load tests.

The test compaction procedure has the following features. Some of these features also exist in other test compaction procedures as described below. The unique feature of the procedure described in this paper is that it uses both multicycle broadside and skewed-load tests to achieve test compaction.

(1) The procedure is applied to a compact test set T that consists of two-cycle broadside and skewed-load tests. It obtains multicycle tests by adding clock cycles and modifying the tests using a simulation-based process in order to avoid sequential test generation.

(2) To reduce the number of tests in a test set that is already compact, the procedure considers pairs of tests $t_{p,0} \in T$ and $t_{p,1} \in T$. It attempts to modify $t_{p,0}$ into a multicycle test that detects all the faults that $t_{p,0}$ and $t_{p,1}$ detect in T. When this is successful, $t_{p,1}$ is removed from T.

(3) When the procedure modifies $t_{p,0}$, it allows its type to change from broadside to skewed-load or vice versa. This is in addition to changing its scan-in state and primary input sequence. All the modifications occur as part of the same process that interleaves the various types of modifications.

The test compaction procedure described in [14] also considers pairs of tests when it attempts to remove a test. However, it does not consider multicycle tests with more than two clock cycles between scan operations. The procedure described in [16] also modifies tests in a given test set in order to create multicycle tests without requiring sequential test generation. However, it only considers broadside tests, and it does not include the option of changing a test type. The procedure described in [11] considers broadside and skewed-load tests for test compaction. However, it only considers two-cycle tests. In addition, it considers the two test types separately, and selects the best of the two for every test that it adds to the test set.

The paper is organized as follows. A format for representing multicycle broadside and skewed-load tests, and the assumptions related to their application, are described in Section II. Section III describes the test compaction procedure. Section IV presents experimental results.

II. Multicycle Tests

A multicycle test t_i is described by the five-tuple $\langle s_i, v_i, l_i, e_i, c_i \rangle$ as shown in Table I and described next. The scan-in state of the test is s_i. The test has a single primary input vector v_i that is held constant for the duration of the test. The number of clock cycles between the scan operations of the test is l_i. For a broadside test, all the clock cycles are functional capture cycles. For a skewed-load test, the first $l_i - 1$ clock cycles are scan shift cycles, and the last clock cycle is a functional capture cycle. The type of the test is given by e_i, with

TABLE II
MULTICYCLE SKEWED-LOAD TEST

s_i	v_i	l_i	e_i	c_i
10011001101100110101011001100110010000	01	7	1	0 0 0 1 1 1 1

u	$\psi_i(u)$	$c_i(u)$
0	10011001101100110101011001100110010000	0
1	01001100110110011010101100110011001000	0
2	00100110011011001101010110011001100100	0
3	00010011001101100110101011001100110010	1
4	10001001100110110011010101100110011001	1
5	11000100110011011001101010110011001100	1
6	11100010011001101100110101011001100110	1
7	01100010011000101100100101011001100110	

$e_i = 0$ for a broadside test, and $e_i = 1$ for a skewed-load test. For a skewed-load test, scan-in values are given by the sequence c_i. For uniformity, c_i is included in the description of both test types even though it is not needed for a broadside test, and its length is l_i even though the last clock cycle of a test is always a functional capture cycle that does not require a scan-in value.

For illustration of the definition of a multicycle skewed-load test, Table II shows a seven-cycle skewed-load test for ITC-99 benchmark $b05$, and the states that the circuit traverses under the test. The circuit has 34 state variables that are included in a single scan chain. The scan chain is shifted to the right. The circuit has two primary inputs. A clock cycle between the scan operations of the test is denoted by u. The state of the circuit at clock cycle u is denoted by $\psi_i(u)$. The state $\psi_i(0)$ is the scan-in state s_i. The state is shifted by one position to the right in clock cycles $0 \leq u \leq 5$. For ease of reference, the scan-in value at clock cycle u is repeated under column $c_i(u)$. Clock cycle $u = 6$ is a functional capture cycle. The state $\psi_i(7)$ is the scan-out state of the test.

Fault detection occurs as follows. Under a multicycle broadside test, a fault effect that reaches a next-state variable before the last functional capture cycle of the test is captured in a flip-flop. Such a fault effect continues to propagate through the circuit in the next clock cycle. The fault is detected if the fault effect reaches a next-state variable at the last clock cycle of the test. In this case, it is scanned out.

To increase the likelihood that faults will be detected by a multicycle broadside test, the primary outputs are observed at every clock cycle between scan operations. In this case, a fault is detected if a fault effect reaches a primary output. To avoid the need to observe the primary outputs at speed, output compaction can be used, and the compacted output response can be scanned out at the end of the test together with the scan-out state.

For a multicycle skewed-load test, such as the one in Table II, faults may be activated at any clock cycle between the scan operations. The use of $l_i - 1$ scan shift cycles ensures that different faults can be activated by skewed-load and broadside tests. This contributes to the effectiveness of using both test types.

When faults are activated by a multicycle skewed-load test, fault effects may reach the primary outputs or next-state variables. A fault effect that reaches a next-state variable is not captured in a flip-flop during a scan shift cycle. Only the last clock cycle of the test is a functional capture cycle, and can capture fault effects that reach the next-state variables. Thus, the last two clock cycles of the test act as a two-cycle skewed-load test.

To ensure that multicycle skewed-load tests with more than two clock cycles are effective, it is important to observe the primary outputs at every clock cycle between scan operations. In this case, a fault whose fault effect reaches a primary output during a scan shift cycle is detected.

Observation of primary output values is assumed in this paper for all the clock cycles between the scan operations of a test. The circuits are assumed to have a single scan chain that is shifted to the right. The only difference between this case and the case of multiple scan chains is that c_i is a sequence of scan-in vectors when the circuit has multiple scan chains.

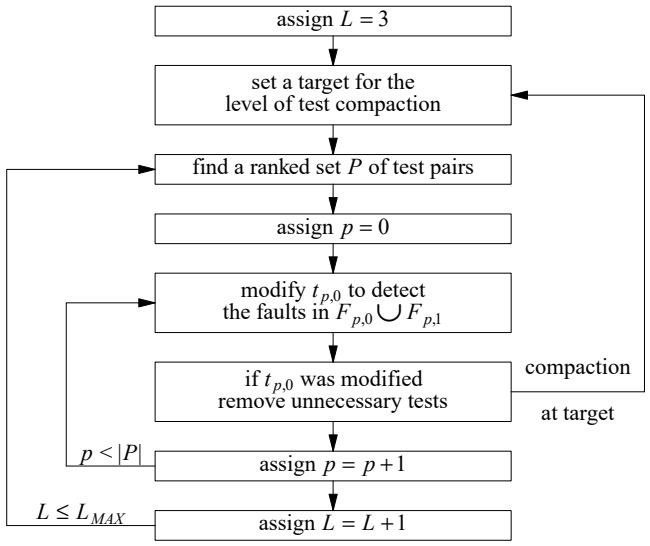

Fig. 1. Test compaction procedure

As in [1]-[2], the tests in this paper use a slow clock for the first clock cycle after the scan-in operation. A fast clock is used for the remaining clock cycles. The target faults are transition faults with an extra delay of a single clock cycle.

III. TEST COMPACTION PROCEDURE

This section describes the test compaction procedure, which is outlined in Figure 1.

A. Procedure Outline

The test set that the test compaction procedure considers is denoted by T. This is initially a compact two-cycle test set that consists of broadside and skewed-load tests. The set of target faults that are detected by T is denoted by F. Fault simulation with fault dropping of F under T yields the first test that detects every fault in F. For a fault $f \in F$, the index of this test is denoted by $ind(f)$. A test $t_i \in T$ is associated with a set of faults $F_i = \{f \in F : ind(f) = i\}$ that are detected by t_i. The procedure may move a fault $f \in F$ between different tests that detect it by changing the value of $ind(f)$.

The procedure uses a parameter that is denoted by L to limit the number of clock cycles in a multicycle test. With a given value of L, an l_i-cycle test that it generates is such that $l_i \leq L$. The procedure increases L gradually up to a limit that is denoted by L_{MAX}. For the experiments in this paper, $L_{MAX} = 8$. The gradual increase of L ensures that the procedure does not consider multicycle tests with more clock cycles than necessary for achieving test compaction.

The procedure defines a set P of test pairs that it will consider for test compaction. The test pair with index p is $\pi_p = (t_{p,0}, t_{p,1})$. When the procedure considers π_p, it attempts to modify $t_{p,0}$ into a multicycle test that detects all the faults in $F_{p,0}$ and $F_{p,1}$. If this is successful, $t_{p,1}$ is removed from T without losing fault coverage.

The test pairs in P are ranked by increasing number of detected faults. This is based on the expectation that, if a pair $\pi_p = (t_{p,0}, t_{p,1})$ detects fewer faults, then $t_{p,0}$ is easier to modify such that $t_{p,1}$ can be removed. The procedure considers the pairs from P one by one in

this order. If the consideration of a test pair $\pi_p = (t_{p,0}, t_{p,1})$ results in the modification of $t_{p,0}$ and the removal of $t_{p,1}$, pairs that include the modified test $t_{p,0}$ or the removed test $t_{p,1}$ are not considered.

To ensure that modified tests are considered as well, and that the ranking of the test pairs in P is up-to-date, the procedure starts a new iteration when the level of test compaction reaches a target. The target used in this paper is a reduction of 5% in the number of clock cycles required for applying the test set. Specifically, let C_{init} be the number of clock cycles required for applying the initial test set. Let C_{comp} be the number of clock cycles required for applying the compacted test set. The first target that the procedure uses is $C_{comp} \leq 0.95C_{init}$. Once this target is achieved, the procedure changes the target to $C_{comp} \leq 0.90C_{init}$, then to $0.85C_{init}$, and so on. If the target is not reached after considering all the test pairs in P, the procedure terminates for the current value of L.

The number of clock cycles is used for measuring the level of test compaction, and not the number of tests, since tests may have increased numbers of clock cycles that are accounted for in C_{comp}.

Details related to the set of test pairs, the removal of tests, and the modification of a test are given next.

B. Test Pairs

Initially, the set P consists of every test pair $\pi_p = (t_{p,0}, t_{p,1})$ for every $t_{p,0} \in T$ and $t_{p,1} \in T$ such that $t_{p,0} \neq t_{p,1}$. This includes cases where $t_{p,0}$ and $t_{p,1}$ have different types. Thus, a broadside test may be modified to detect all the faults that are detected by a skewed-load test, and vice versa. It also includes cases where $t_{p,0}$ and $t_{p,1}$ detect faults with conflicting detection conditions, such as the $0 \rightarrow 1$ and $1 \rightarrow 0$ transition faults on the same line. Such faults can be detected by a multicycle test using different clock cycles for activation and propagation. For every test pair $t_{p,0} \in T$ and $t_{p,1} \in T$, both $(t_{p,0}, t_{p,1})$ and $(t_{p,1}, t_{p,0})$ are included in P.

A test pair $\pi_p = (t_{p,0}, t_{p,1})$ is associated with the number of faults that the pair detects together. This number is denoted by $n_{det}(\pi_p)$. This number can be computed by using the subsets of faults that the tests detect, $F_{p,0}$ and $F_{p,1}$, after fault simulation with fault dropping of F under T. Using these subsets, $n_{det}(\pi_p) = |F_{p,0}| + |F_{p,1}|$.

However, after fault simulation with fault dropping, some of the faults in $F_{p,0}$ and $F_{p,1}$ may also be detected by other tests in T. Before the procedure considers the pair π_p for modification and removal, it moves as many faults as possible from $F_{p,0}$ and $F_{p,1}$ to other tests. It is thus important to also move faults from $F_{p,0}$ and $F_{p,1}$ to other tests for the computation of $n_{det}(\pi_p)$.

To find the minimum subsets $F_{p,0}$ and $F_{p,1}$, the faults in $F_{p,0} \cup F_{p,1}$ are simulated under the tests in $T \setminus \{t_{p,0}, t_{p,1}\}$. If a fault $f \in F_{p,0} \cup F_{p,1}$ is detected by a test $t_i \in T \setminus \{t_{p,0}, t_{p,1}\}$, $det(f) = i$ is assigned in order to move f to F_i.

This computation is carried out before π_p is considered for modification and removal. For the ranking of the test pairs in P, an estimate of $n_{det}(\pi_p)$ is used, whose computation requires a lower computational effort, but whose accuracy is better than using fault simulation with fault dropping. For the estimate, every test $t_i \in T$ is considered individually, instead of considering test pairs. For t_i, the procedure moves faults from F_i to other tests by simulating F_i under $T \setminus \{t_i\}$, and updating $ind(f)$ if a test t_j other than t_i detects it. Let the number of faults that remain in F_i be $n_{det}(t_i)$. For every $\pi_p \in P$, the procedures uses $n_{det}(\pi_p) = n_{det}(t_{p,0}) + n_{det}(t_{p,1})$ as an estimate of the number of faults that π_p detects.

Experimental results indicate that most of the test pairs that contribute to test compaction have low values of $n_{det}(\pi_p)$, typically $2 \leq n_{det}(\pi_p) \leq 8$. Therefore, a constant bound N_{DET} is introduced, and only test pairs with $n_{det}(\pi_p) \leq N_{DET}$ are included in P. In addition, the number of test pairs in P is limited to a constant that is denoted by N_{PAIRS}. If P contains more than N_{PAIRS} test pairs, only N_{PAIRS} test pairs are kept in P, with the lowest numbers of detected faults. For the experiments in this paper, $N_{DET} = 16$ and $N_{PAIRS} = 10000$.

TABLE III
MODIFICATION OF A TEST

i	s_i	v_i	l_i	e_i	c_i
61	1000011001101100111101001011111101	00	2	1	00
53	0000111111010001110111110011001011	10	2	0	00
mod	0000111111010001110111110011001011	10	3	0	000
mod	0000111011010001110111110001001011	10	3	1	000

C. Removing Unnecessary Tests

After the procedure modifies a test $t_{p,0}$ successfully such that a test $t_{p,1}$ can be removed from T, it checks whether the modification of $t_{p,0}$ allows other tests to be removed as well. For this purpose, the procedure places $t_{p,0}$ at the beginning of the test set, and $t_{p,1}$ at the end of the test set. In this order, it performs fault simulation with fault dropping of F under T. The procedure removes every test $t_i \in T$ for which $F_i = \emptyset$ is obtained. In addition to $t_{p,1}$, this condition may be satisfied for other tests. All such tests are removed from T.

D. Modifying a Test

The main part of the procedure is the modification of a test $t_{p,0}$ such that it would detect all the faults in the set $F_{p,0} \cup F_{p,1}$. Let $t_{p,0} = \langle s_{p,0}, v_{p,0}, l_{p,0}, e_{p,0}, c_{p,0} \rangle$. As the procedure modifies $t_{p,0}$, it stops as soon as it finds that $t_{p,0}$ detects all the faults in $F_{p,0} \cup F_{p,1}$.

The test is first modified into an L-cycle test by adding $L - l_{p,0}$ random values at the end of $c_{p,0}$, and assigning $l_{p,0} = L$.

With $l_{p,0} = L$, the candidates for modification are the bits of $s_{p,0}$, $v_{p,0}$, and $c_{p,0}$, and the test type $e_{p,0}$. All the entries are considered in a random order a constant number of times. The constant four is used for the experiments in this paper. When an entry is considered, its value is complemented. Let $t_{p,0}^{mod}$ be the modified test. The procedure simulates $F_{p,0} \cup F_{p,1}$ under $t_{p,0}^{mod}$. If the modification does not reduce the number of detected faults, it is accepted by assigning $t_{p,0} = t_{p,0}^{mod}$. Otherwise, $t_{p,0}^{mod}$ is discarded.

Table III shows an example where a test pair of ITC-99 benchmark $b05$ is considered for modification and removal. The pair is (t_{53}, t_{61}) in a test set that initially consists of 67 tests. The test t_{53}, which is shown in the second row of Table III, is a broadside test, while the test t_{61}, which is shown in the first row of Table III, is a skewed-load test. The set $F_{53} \cup F_{61}$ consists of two faults after faults are moved to other tests in the test set. Initially, t_{53} detects one of the two faults in $F_{53} \cup F_{61}$. The test t_{53} is extended into a three-cycle broadside test as shown in the third row of Table III. Several entries of the test are complemented next before the number of detected faults is increased to two, and the modification terminates. The resulting test is shown in the fourth row of Table III. As part of the modification, the type of the test was changed from broadside to skewed-load by changing the value of e_i. The procedure accepts to replace t_{53} with the modified test in order to remove t_{61}.

The worst-case computational complexity of the test compaction procedure is determined by the modification procedure as follows. With a constant number of test pairs in P, the procedure modifies a constant number of tests in every iteration. For B bits in a test, the modification of a test requires fault simulation of at most N_{DET} faults under $O(B)$ modified tests. The number of iterations is $O(|T|)$. The worst case is obtained if the procedure removes a single test in every iteration. Considering all the iterations, the procedure simulates

TABLE IV
EXPERIMENTAL RESULTS

circuit	sv	len	iter	pair	det	tests	2brd	2skw	mbrd	mskw	mix	mod	ave	cycles	ratio	f.c.	ntime
s38417	1636	-	0	-	-	663	337	326	0	0	-	-	2.000	1087630	1.000	98.254	-
s38417	1636	4	1	7392	2	629	289	307	32	1	17	0	2.070	1031982	0.949	98.254	357.25
s38417	1636	6	1	8972	2	596	241	289	63	3	31	0	2.227	978019	0.899	98.255	1121.64
b21	494	-	0	-	-	360	137	223	0	0	-	-	2.000	179054	1.000	84.688	-
b21	494	4	1	647	2	341	111	212	18	0	11	0	2.059	169650	0.947	84.688	69.06
b21	494	8	1	1608	3	329	100	201	28	0	22	0	2.161	163731	0.914	84.688	509.33
b20	494	-	0	-	-	288	106	182	0	0	-	-	2.000	143342	1.000	88.544	-
b20	494	3	1	8241	4	273	83	175	15	0	7	0	2.055	135917	0.948	88.544	67.97
b20	494	8	1	7337	5	262	69	168	25	0	14	0	2.164	130489	0.910	88.544	1096.82
b15	447	-	0	-	-	440	121	319	0	0	-	-	2.000	198007	1.000	93.715	-
b15	447	4	1	7699	3	417	94	304	15	4	11	0	2.070	187709	0.948	93.718	135.79
b15	447	8	1	8877	5	397	73	292	26	6	19	0	2.244	178797	0.903	93.721	792.98
s9234	228	-	0	-	-	275	119	156	0	0	-	-	2.000	63478	1.000	85.229	-
s9234	228	4	1	2860	4	261	100	148	13	0	8	1	2.061	60274	0.950	85.229	122.58
s9234	228	6	1	6584	7	250	83	146	20	1	8	1	2.164	57769	0.910	85.229	638.22
s38584	1452	-	0	-	-	473	206	267	0	0	-	-	2.000	689194	1.000	82.192	-
s38584	1452	3	1	8405	2	449	166	261	20	2	2	0	2.049	654320	0.949	82.192	68.72
s38584	1452	4	1	7704	3	425	138	246	34	7	4	0	2.125	619455	0.899	82.202	212.51
s38584	1452	5	1	8458	4	412	123	244	37	8	4	0	2.182	600575	0.871	82.205	381.01
tv80	359	-	0	-	-	706	166	540	0	0	-	-	2.000	255225	1.000	96.330	-
tv80	359	8	1	2303	2	687	137	532	18	0	8	0	2.079	248420	0.973	96.330	391.92
b22	709	-	0	-	-	335	89	246	0	0	-	-	2.000	238894	1.000	87.716	-
b22	709	8	1	1763	2	318	68	233	17	0	13	0	2.094	226837	0.950	87.731	424.11
systemcaes	670	-	0	-	-	160	40	120	0	0	-	-	2.000	108190	1.000	95.602	-
systemcaes	670	3	1	3132	7	151	28	114	9	0	6	0	2.060	102151	0.944	95.602	134.71
systemcaes	670	5	1	1530	8	143	22	108	13	0	12	0	2.140	96786	0.895	95.602	1100.46
systemcaes	670	8	1	2359	14	137	17	105	15	0	15	0	2.299	92775	0.858	95.602	3483.54
aes_core	530	-	0	-	-	275	88	187	0	0	-	-	2.000	146830	1.000	97.452	-
aes_core	530	4	1	31	6	261	77	174	10	0	7	4	2.050	139395	0.949	97.452	48.04
aes_core	530	5	1	4	7	260	77	173	10	0	8	4	2.058	138865	0.946	97.452	75.95
b14	247	-	0	-	-	219	75	144	0	0	-	-	2.000	54778	1.000	84.162	-
b14	247	3	1	7133	5	210	60	141	9	0	3	0	2.043	52546	0.959	84.162	61.47
i2c	128	-	0	-	-	72	15	57	0	0	-	-	2.000	9488	1.000	81.841	-
i2c	128	3	1	2297	13	68	11	53	4	0	4	1	2.059	8972	0.946	81.841	226.05
i2c	128	6	1	1849	14	64	6	50	7	1	5	1	2.250	8464	0.892	82.005	1869.01
i2c	128	6	2	1767	16	63	6	49	7	1	5	1	2.254	8334	0.878	82.005	2315.96
wb_dma	523	-	0	-	-	177	53	124	0	0	-	-	2.000	93448	1.000	83.233	-
wb_dma	523	5	1	4965	3	168	43	118	6	1	5	1	2.065	88734	0.950	83.233	899.53
wb_dma	523	8	1	5578	4	160	33	117	8	2	6	2	2.244	84562	0.905	83.233	3325.36
systemcdes	190	-	0	-	-	81	24	57	0	0	-	-	2.000	15742	1.000	96.404	-
systemcdes	190	5	1	23	7	76	18	54	4	0	3	0	2.092	14789	0.939	96.404	210.38
systemcdes	190	8	1	44	9	75	16	54	5	0	3	0	2.173	14603	0.928	96.404	578.60
s1423	74	-	0	-	-	58	16	42	0	0	-	-	2.000	4482	1.000	85.278	-
s1423	74	3	1	1062	7	55	11	41	3	0	1	0	2.055	4257	0.950	85.278	79.79
s1423	74	4	1	180	4	52	8	39	4	1	1	0	2.115	4032	0.900	85.278	384.10
s1423	74	7	1	66	3	48	5	35	6	2	3	0	2.396	3741	0.835	85.278	1714.78
s1423	74	8	1	0	2	46	5	32	6	3	4	0	2.630	3599	0.803	85.278	2201.60
s15850	597	-	0	-	-	328	101	227	0	0	-	-	2.000	197069	1.000	90.437	-
s15850	597	4	1	6028	3	311	81	214	12	4	8	0	2.058	186904	0.948	90.437	199.23
s15850	597	8	1	9195	4	300	74	203	13	10	9	0	2.197	180356	0.915	90.446	1326.33
spi	229	-	0	-	-	494	69	425	0	0	-	-	2.000	114343	1.000	91.009	-
spi	229	8	1	8266	3	477	56	408	8	5	10	1	2.075	110452	0.966	91.009	663.62
simple_spi	131	-	0	-	-	63	14	49	0	0	-	-	2.000	8510	1.000	89.215	-
simple_spi	131	3	1	953	5	59	9	46	4	0	3	0	2.068	7982	0.938	89.215	75.28
simple_spi	131	3	2	716	6	56	8	43	3	2	4	1	2.089	7584	0.891	89.267	149.50
simple_spi	131	6	1	415	6	54	5	43	4	2	4	1	2.185	7323	0.861	89.267	1326.87
sasc	117	-	0	-	-	36	8	28	0	0	-	-	2.000	4401	1.000	93.636	-
sasc	117	3	1	40	2	34	6	26	1	1	0	0	2.059	4165	0.946	93.636	12.97
sasc	117	5	1	18	3	32	5	24	1	2	0	0	2.188	3931	0.893	93.636	295.50
sasc	117	7	1	69	5	31	3	24	2	2	0	0	2.355	3817	0.867	93.636	716.62
s13207	669	-	0	-	-	445	167	278	0	0	-	-	2.000	299264	1.000	94.859	-
s13207	669	3	1	9716	2	422	144	255	11	12	5	0	2.055	283854	0.949	94.859	166.52
s13207	669	4	1	1565	2	400	127	228	21	24	10	0	2.120	269117	0.899	94.859	367.68
s13207	669	5	1	3223	2	377	113	200	27	37	16	0	2.268	253737	0.848	94.859	660.27
s13207	669	7	1	4877	3	355	98	180	35	42	21	0	2.470	239041	0.799	94.859	1367.23
s13207	669	8	1	7475	3	343	92	169	36	46	24	0	2.627	231037	0.772	94.859	2042.50
s5378	179	-	0	-	-	233	96	137	0	0	-	-	2.000	42352	1.000	83.513	-
s5378	179	3	1	2484	2	221	82	127	7	5	2	0	2.054	40192	0.949	83.513	45.41
s5378	179	3	2	6303	3	209	72	115	11	11	3	0	2.105	38030	0.898	83.513	122.43
s5378	179	4	1	583	2	197	64	101	15	17	3	0	2.168	35869	0.847	83.513	267.02
s5378	179	5	1	6447	6	185	60	86	15	24	5	0	2.303	33720	0.796	83.513	608.04
s5378	179	7	1	9415	7	174	58	72	15	29	5	0	2.529	31765	0.750	83.513	1524.56
s5378	179	8	1	9059	8	172	57	69	16	30	6	0	2.599	31414	0.742	83.513	2303.99

$O(|T|B)$ faults and tests. In practice, the procedure does not consider all the test pairs in every iteration, and it terminates after a small number of iterations with every value of L. These effects are captured by the normalized run time defined in Section IV.

IV. EXPERIMENTAL RESULTS

The test compaction procedure is applied to transition faults. The initial test set consists of two-cycle broadside and skewed-load tests, and it is compacted by the procedure from [11].

The results are shown in Table IV. Considering the multicycle tests with three or more clock cycles in the final compacted test set, let T_{mbrd} be the subset of broadside tests, and let T_{mskw} be the subset of skewed-load tests. The circuits are arranged by decreasing order of $|T_{mbrd}| - |T_{mskw}|$.

The first row for every circuit describes the initial two-cycle test set. The next rows describe the test set obtained by the test compaction procedure for every 5% reduction in the number of clock cycles for test application, as well as the final test set. Reductions of 5% may be obtained for different values of L in different circuits. For example, when $L = 3$ is effective for a circuit, the first 5% reduction may be obtained with this value of L. For other circuits, the first 5% reduction may be obtained for a larger value of L.

After the circuit name, column sv shows the number of state variables. Column len shows the value of L. Column $iter$ shows the iteration of the procedure, where a zero stands for the initial test set. Column $pair$ shows the index of the test pair from P for which the test set is obtained. Column det shows the estimate for the number of faults that this pair detects.

Column $tests$ shows the number of tests in the test set. Columns $2brd$ and $2skw$ show the numbers of two-cycle broadside and skewed-load tests, respectively. Columns $mbrd$ and $mskw$ show the numbers of multicycle broadside and skewed-load tests with three or more clock cycles, respectively.

Column mix shows the number of test pairs of different types that resulted in the removal of a test. Column mod shows the number of tests that were modified to remove another test, and the modification included a type change.

Column ave shows the average number of clock cycles between the scan operations of a test. Column $cycles$ shows the total number of clock cycles required for applying the test set. Column $ratio$ shows the fraction of clock cycles required for the test set produced by the test compaction procedure relative to the initial two-cycle test set. The ratio measures the level of test compaction achieved by the procedure as discussed earlier.

Colmn $f.c.$ shows the transition fault coverage. The procedure does not allow the fault coverage to decrease as it compacts the test set.

Column $ntime$ shows the normalized run time of the procedure. Since the procedure is based on fault simulation, the run time for fault simulation with fault dropping of the initial test set divides the cumulative run time of the test compaction procedure to obtain the normalized run time.

The following points can be seen from Table IV. The test compaction procedure reduces the number of tests, and the number of clock cycles required for test application, for all the circuits considered. The level of test compaction is different for different circuits because multiple clock cycles are effective to different extents in detecting additional faults. This applies to both broadside and skewed-load tests.

Test pairs consisting of broadside and skewed-load tests are used effectively by the test compaction procedure, such that the modification of a test of one type results in the removal of a test of the other type. In addition, the type of a modified test is changed in several cases to achieve test compaction. Overall, the number of two-cycle tests of both types is reduced.

The normalized run time does not increase with the size of the circuit. Thus, the procedure scales similar to a fault simulation procedure. Moreover, there is typically a point where the run time increases abruptly before an additional test is removed. It is possible to terminate the procedure for the current value of L when the removal of another test does not occur after a preselected increase in the run time.

V. Concluding Remarks

This paper suggested a definition of a multicycle skewed-load test that complements the definition of a multicycle broadside test, and described a test compaction procedure that produces compact multicycle test sets consisting of broadside and skewed-load tests. The procedure combines the advantages of using multicycle tests for test compaction with the advantages of using both broadside and skewed-load tests for increasing the fault coverage and achieving test compaction. Experimental results for benchmark circuits demonstrated the effectiveness of multicycle broadside and skewed-load tests in achieving test compaction.

References

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

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

[3] 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.

[4] 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.

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

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

[7] 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.

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

[10] 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.

[11] I. Pomeranz, "Static Test Compaction for Delay Fault Test Sets Consisting of Broadside and Skewed-Load Tests", in Proc. VLSI Test Symp., 2011, pp. 84-89.

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

[13] Y. Sato, H. Yamaguchi, M. Matsuzono and S. Kajihara, "Multi-Cycle Test with Partial Observation on Scan-Based BIST Structure", in Proc. Asian Test Symp., 2011, pp. 54-59.

[14] I. Pomeranz, "Static Test Compaction for Scan Circuits by Using Restoration to Modify and Remove Tests", IEEE Trans. on Computer-Aided Design, Dec. 2014, pp. 1955-1964.

[15] 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.

[16] 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.

[17] 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.

[18] N. Ahmed, M. Tehranipoor, C. P. Ravikumar and K. M. Butler, "Local At-Speed Scan Enable Generation for Transition Fault Testing Using Low-Cost Testers", IEEE Trans. on Computer-Aided Design, May 2007, pp. 896-905.

[19] G. Xu and A. D. Singh, "Scan Cell Design for Launch-on-Shift Delay Tests with Slow Scan Enable", IET Computers & Digital Techniques, May 2007, pp. 213-219.