# Short Paper

## Input Test Data Volume Reduction for Skewed-Load Tests by Additional Shifting of Scan-In States

Irith Pomeranz

***Abstract*—Test data compression methods reduce the input test data volume by allowing compressed tests to be stored on a tester. Additional reductions in the input test data volume can be achieved if each stored test is used for producing several different tests. Skewed-load tests create a unique opportunity to expand a stored test into several different skewed-load tests by continuing to shift the scan-in state for one or more additional clock cycles. This opportunity for test data volume reduction beyond test data compression is introduced in this paper. The paper describes a procedure that starts from a given skewed-load test set. The procedure removes tests from the test set and recovers the fault coverage by applying several tests based on every stored test.**

***Index Terms*—Input test data volume, scan-based tests, skewed-load tests, test data compression, transition faults.**

## I. Introduction

Test data compression methods reduce the input test data volume by allowing compressed tests to be stored on a tester, and reproducing the tests on-chip by using decompression logic [1]–[4]. Additional reductions in the input test data volume can be achieved if each stored test is used for producing several different tests [5]–[7]. The approaches described in [5]–[6] are designed for built-in test generation. In [7], each stored test is used for producing three tests, a broadside test and two skewed-load tests. This allows the number of stored tests to be reduced without losing fault coverage.

Skewed-load tests create a unique opportunity to expand a stored test into several different skewed-load tests by continuing to shift the scan-in state for one or more additional clock cycles. None of the previous papers in this area addressed skewed-load tests directly, or identified this opportunity for achieving test data volume reduction beyond test data compression. This is important since skewed-load tests are commonly used for detecting delay faults in standard-scan circuits. The following is the basic idea of the new input test data volume reduction method.

Let $t_i =< s_i v_i, p_i u_i >$ be a skewed-load test. The scan-in state $s_i$ and the primary input vector $v_i$ define the first pattern that the test applies to the combinational logic of the circuit. The state $p_i$ is obtained by a single shift of $s_i$. The state $p_i$ and the primary input vector $u_i$ define the second pattern that the test applies to the circuit. For a circuit with $K$ state variables, let $s_i = s_i(0)\ s_i(1) \ldots s_i(K-1)$ and $p_i = p_i(0)\ p_i(1) \ldots p_i(K-1)$.

Manuscript received June 28, 2013; revised August 25, 2013 and October 30, 2013; accepted November 4, 2013. Date of current version March 17, 2014. This paper was recommended by Associate Editor X. Wen.

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

Digital Object Identifier 10.1109/TCAD.2013.2290085

Assuming, for simplicity of discussion, a single scan chain that is shifted to the right, the fact that $p_i$ is obtained from $s_i$ by a single shift implies that $p_i(j) = s_i(j-1)$ for $0 < j < K$. The value of $p_i(0)$ is determined by the scan-in value that follows the scan-in operation of $s_i$.

To accommodate tester limitations that prevent primary input vectors from being changed at-speed, the skewed-load tests considered in this paper are such that $u_i = v_i$. Denoting the scan-in value that determines $p_i(0)$ by $\gamma_i$, a skewed-load test is determined by the triple $(s_i, v_i, \gamma_i)$. The test corresponding to this triple is $< s_i v_i, p_i v_i >$, with $p_i(0) = \gamma_i$ and $p_i(j) = s_i(j-1)$ for $0 < j < K$.

The triple $(s_i, v_i, \gamma_i)$ can be used for applying additional skewed-load tests, where the state under the first pattern is obtained by shifting $s_i$ one or more additional times. For example, with $s_i = 10000$, $v_i = 11$ and $\gamma_i = 0$, a single additional shift of $s_i$ yields the state 01000, two additional shifts of $s_i$ yield the state 00100, and so on. The corresponding skewed-load tests are <10000 11, 01000 11>, <01000 11, 00100 11>, <00100 11, 00010 11>, and so on. It is also possible to define skewed-load tests with the complement of $\gamma_i$ to obtain the tests <10000 11, 11000 11>, <11000 11, 11100 11>, <11100 11, 11110 11>, and so on.

In general, with the approach described in this paper, a triple $(s_i, v_i, \gamma_i)$ is extracted from every skewed-load test $t_i$ in a given compacted skewed-load test set. With $0 \le n_j \le K$ and $\beta = 0$ or 1, the triple $(s_i, v_i, \gamma_i)$ can be used for defining up to $2(K+1)$ tests. In the test denoted by $(i, n_j, \beta)$, $s_i$ is shifted $n_j$ additional times to define the state for the first pattern of the test. The scan-in vector during these additional shifts is $\gamma_i$ if $\beta = 0$, and $\bar{\gamma}_i$ if $\beta = 1$. The primary input vector for the test is $v_i$. With up to $2(K+1)$ tests based on every stored test $t_i$, or every triple $(s_i, v_i, \gamma_i)$, it is possible to store fewer tests without losing fault coverage. This allows the input test data volume to be reduced. In the implementation considered in this paper, information about the tests that need to be applied is stored as well. As a result, significantly fewer than $2(K+1)$ tests are applied for every stored test.

The paper describes a procedure that accepts a compact skewed-load test set $T$, and produces a reduced test set $T_s \subseteq T$ and a set $T_a$ of additional tests. The test set $T$ can be generated by any test generation procedure [8]–[11]. The tests in $T_s$ need to be stored. The tests in $T_a$ are derived from the tests in $T_s$. Together, $T_s$ and $T_a$ detect all the target faults that are detected by $T$. If the tests in $T$ satisfy the constraints of a test data compression method that allows compressed tests to be stored, these constraints are carried over to $T_s$. Thus, the reductions in input test data volume are achieved on top of those that can be achieved for $T$.

Shifting the scan-in state of a test in order to obtain new tests is also possible with broadside tests. A broadside test can be described by a pair $(s_i, v_i)$. For consistency with the description

0278-0070 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

TABLE I
SKEWED-LOAD TEST SET

| $i$ | $< s_i v_i, p_i u_i >$ |
|---|---|
| 0 | <10110 111, 01011 111> |
| 1 | <00101 101, 10010 101> |
| 2 | <01010 101, 00101 101> |
| 3 | <01000 001, 00100 001> |
| 4 | <11011 101, 01101 101> |
| 5 | <10100 001, 01010 001> |
| 6 | <10010 111, 11001 111> |
| 7 | <00011 001, 00001 001> |
| 8 | <01010 001, 00101 001> |
| 9 | <01000 111, 00100 111> |
| 10 | <01101 001, 00110 001> |
| 11 | <11100 011, 01110 011> |
| 12 | <10000 011, 11000 011> |
| 13 | <00111 011, 00011 011> |
| 14 | <00010 111, 10001 111> |
| 15 | <00100 111, 00010 111> |
| 16 | <10110 101, 11011 101> |

TABLE II
SELECTED SKEWED-LOAD TESTS AND TRIPLES

| $i$ | $< s_i v_i, p_i u_i >$ | det |
|---|---|---|
| 0 | <01000 001, 00100 001> | 26 |
| 1 | <01010 001, 00101 001> | 38 |
| 2 | <01000 111, 00100 111> | 45 |
| 3 | <01101 001, 00110 001> | 52 |
| 4 | <11100 011, 01110 011> | 65 |
| 5 | <00111 011, 00011 011> | 81 |
| 6 | <00100 111, 00010 111> | 90 |
| 7 | <10110 101, 11011 101> | 102 |

| $i$ | $(s_i, v_i, \gamma_i)$ | $n_j$ | $\beta$ | $(i, n_j, \beta)$ | det |
|---|---|---|---|---|---|
| 0 | (01000,001,0) | | | | |
| | | 0 | 1 | <01000 001, 10100 001> | 103 |
| | | 1 | 0 | <00100 001, 00010 001> | 105 |
| | | 1 | 1 | <10100 001, 11010 001> | 108 |
| 1 | (01010,001,0) | | | | |
| 2 | (01000,111,0) | | | | |
| | | 1 | 1 | <10100 111, 11010 111> | 114 |
| 3 | (01101,001,0) | | | | |
| 4 | (11100,011,0) | | | | |
| | | 1 | 0 | <01110 011, 00111 011> | 115 |
| 5 | (00111,011,0) | | | | |
| | | 1 | 0 | <00011 011, 00001 011> | 119 |
| | | 1 | 1 | <10011 011, 11001 011> | 129 |
| 6 | (00100,111,0) | | | | |
| | | 1 | 0 | <00010 111, 00001 111> | 131 |
| | | 1 | 1 | <10010 111, 11001 111> | 134 |
| 7 | (10110,101,1) | | | | |
| | | 1 | 1 | <01011 101, 00101 101> | 139 |
| | | 1 | 0 | <11011 101, 11101 101> | 144 |

of a skewed-load test, it is possible to extract a scan-in vector $\gamma_i$ from the broadside test, and use a triple $(s_i, v_i, \gamma_i)$. The triple $(i, n_j, \beta)$ implies that after $s_i$ is scanned in, $\gamma_i$ or $\bar{\gamma}_i$ is shifted in during $n_j$ additional clock cycles. Next, two functional clock cycles are applied, with primary input vector $v_i$, in order to apply a broadside test. With this approach, the same triples can be used for describing both types of tests. Shifting the scan-in state to obtain new tests fits better with skewed-load tests where the scan-in vector $\gamma_i$ is needed as part of the test. In this paper only skewed-load tests are considered. The paper focuses on the input test data volume. For the output test data volume it is assumed that output compaction is used as in [2] and [3].

The paper is organized as follows. Section II presents an example to illustrate the basis for the input test data volume reduction method described in this paper. Section III describes the test application process and the storage requirements. Section IV describes the procedure for input test data volume reduction. Section V presents experimental results.

## II. EXAMPLE

This section illustrates the possibility of reducing the input test data volume for skewed-load tests by producing several tests based on every test in a given test set.

The circuit considered in this section is ITC-99 benchmark *b*01. A skewed-load test set for the circuit is shown in Table I. The test set detects 144 transition faults.

The first part of Table II shows a subset of the test set from Table I. The tests were selected by the procedure described in Section IV. They were given consecutive indices (different from the ones in Table I) for ease of reference. For every test, column *det* shows the number of transition faults that are detected after the test is simulated.

The second part of Table II shows a set of triples $(s_i, v_i, \gamma_i)$ based on the tests shown in the first part of Table II. For every triple, the table also shows the values of $n_j$ and $\beta$ for which the corresponding tests increase the number of detected faults. The tests in the first part of Table II are simulated first, followed by the tests in the second part. The number of detected faults after a test is simulated is shown in the last column.

For example, $t_0$ = <01000 001, 00100 001> from Table II is equal to $t_3$ from Table I. The corresponding triple is $(s_0, v_0, \gamma_0)$ = (01000, 001, 0). The test obtained with $n_j = 0$ and $\beta = 1$ uses $s_0$ as the initial state of the test, and shifts it with the value 1 to obtain 10100 as part of the second pattern. The test obtained with $n_j = 1$ and $\beta = 0$ shifts $s_0$ by one position with scan-in value 0 to obtain 00100 as the initial state of the test. It shifts it again with the value 0 to obtain 00010 as part of the second pattern.

Tables I and II demonstrate the following point. The (compacted) skewed-load test set for *b*01 contains 17 tests. It is sufficient to store only eight of these tests, which are shown in the first part of Table II, and use them to produce additional skewed-load tests with specific parameters $n_j$ and $\beta$, as shown in the second part of Table II. In both cases, the tests applied to the circuit detect the same 144 transition faults.

The tests obtained from the same triple $(s_i, v_i, \gamma_i)$ with different values of $n_j$ and $\beta$ can be substantially different. For example, in the test obtained from $(s_0, v_0, \gamma_0)$ with $n_j = 0$ and $\beta = 1$, the state during the second pattern of the test is 10100. With $n_j = 1$ and $\beta = 0$, the state during the second pattern is 00010. The states differ in three out of five bits. These differences contribute to the ability to detect faults that are detected by other tests in the test set, allowing such tests to be removed.

## III. TEST APPLICATION AND STORAGE REQUIREMENTS

The application of a test described by $(i, n_j, \beta)$, with $t_i \in T_s$, $n_j \geq 0$ and $\beta = 0$ or 1, can proceed in one of two ways.

If $s_i$ is stored in a compressed form and applied through on-chip decompression logic, application of the test can be done by scanning in $s_i$ and then scanning in $\gamma_i$ (if $\beta = 0$) or $\bar{\gamma}_i$ (if $\beta = 1$) for $n_j$ additional clock cycles. This will bring the circuit into the initial state of the skewed-load test. Scanning $\gamma_i$ or $\bar{\gamma}_i$ for one additional clock cycle, followed by a functional clock cycle, will apply the skewed-load test to the circuit. This does not require any on-chip hardware support. It requires a tester where it is possible to adjust the number of scan-in cycles for a test based on a triple $(i, n_j, \beta)$. The tester needs to store the test set $T_s$, and the triples $(i, n_j, \beta)$ that indicate how to apply the tests in $T_s$. Based on these triples, the tester applies each test with the correct number of scan clock cycles. Since testers are already able to apply multicycle tests, adding these requirements is preferred over transferring more test data to the chip, and synthesizing on-chip logic to control the number of clock cycles in a test based on the additional test data.

It is possible to use the additional scan cycles of the test for fault detection. This may increase the fault coverage compared with the fault coverage of $T$ (to the extent that multicycle tests, when applied at-speed, can detect faults that cannot be detected by two-cycle tests), and possibly reduce the number of stored tests. However, it would also require sequential fault simulation of the same number of multicycle tests to assess the effect on the set of detected faults. In this paper, only the last two clock cycles of the test are used for fault detection. They are simulated as a two-pattern test using the same fault simulation procedure as skewed-load tests. This is sufficient for detecting all the faults that can be detected by skewed-load tests, and achieving significant reductions in test data volume.

If test data compression is not used, another option is to compute on the tester the state that is obtained by shifting $s_i$ by $n_j$ positions with a scan-in vector that is determined by $\beta$. The computation requires only a shift operation. The resulting state can then be used as the scan-in state of the test.

Storage of a skewed-load test for a circuit with $K$ state variables, $N$ primary inputs and $M$ scan chains requires $K + N + M$ bits for a scan-in state, a primary input vector, and a scan-in vector for the second pattern of the test. With test data compression, let the number of bits required for a test be $b(K, N, M)$. For simplicity it is assumed that this number is only a function of $K$, $N$ and $M$. For a test set $T$ the number of bits is $|T|b(K, N, M)$.

With the approach described in this paper, a subset $T_s$ of $T$ is stored. In addition, there is a set $T_a$ that indicates which skewed-load tests need to be applied based on the tests in $T_s$ in order to achieve the fault coverage of $T$. Both $T_s$ and $T_a$ need to be stored on the tester. Each entry of $T_a$ consists of an index of a test in $T_s$, and values for $n_j$ and $\beta$. Let $n_{max}$ be the maximum value of $n_j$. The number of bits required for storing this information is as follows.

The number of bits for storing $T_s$ is $|T_s|b(K, N, M)$. The number of bits for storing an entry of $T_a$ is $\lceil log_2|T_s|\rceil$ for an index of a test in $T_s$, $\lceil log_2(n_{max} + 1)\rceil$ bits for storing $n_j$, and one bit for $\beta$. The total number of bits for $T_s$ and $T_a$ is $|T_s|b(K, N, M) + |T_a|(\lceil log_2|T_s|\rceil + \lceil log_2(n_{max} + 1)\rceil + 1)$.

In this formula, the size of $T_a$ is multiplied by parameters that increase logarithmically with $|T_s|$ and $K$. Therefore, a reduction in $|T_s|$ can translate into a reduction in the input test data volume even if compressed tests are stored. For the results reported in this paper, $b(K, N, M) = K + N + 1$.

## IV. Procedure

This section describes a procedure that accepts a compact skewed-load test set $T$, and computes sets $T_s$ and $T_a$ that satisfy the following conditions.

1) $T_s \subseteq T$ is the set of stored skewed-load tests.
2) Each test in $T_a$ is described by a triple $(i, n_j, \beta)$ such that $t_i \in T_s$, $0 \le n_j \le K$ and $\beta = 0$ or 1.
3) The test set $T_s \cup T_a$ detects all the target faults that are detected by $T$. The set of target faults that are detected by $T$ is denoted by $F$.
4) The number of bits required for storing $T_s$ and $T_a$ is as small as possible.

The procedure initially assigns $T_s = T$ and $T_a = \emptyset$. It attempts to remove tests from $T_s$ one at a time. When a test $t$ is removed, the procedure checks whether it is possible to find $T_a$, which is based on the tests that remain in $T_s$, such that $T_s \cup T_a$ detects all the faults in $F$. If $T_a$ is found, $t$ is removed permanently from $T_s$. Otherwise, $t$ is reintroduced into $T_s$.

The procedure considers an increasing bound on the values that it may use for $n_j$. The bound is denoted by $n_{max}$, and its values are $n_{max} = 0, 1, \ldots, K$. For every value of $n_{max}$ the procedure attempts to remove every test from $T_s$. With $n_{max}$, there are $2|T_s|(n_{max} + 1)$ tests of the form $(i, n_j, \beta)$ that can be included in $T_a$. Increasing $n_{max}$ gradually serves two purposes: 1) the number of tests that can potentially be included in $T_a$ is increased as $n_{max}$ is increased. However, at the same time, the number of tests in $T_s$ is decreased. As a result, the increase in the number of tests that need to be considered is moderated and 2) with low values of $n_{max}$ there are fewer options for a test $t \in T_s$ to be removed and replaced by tests in $T_a$. A test that can be removed even when $n_{max}$ is low is considered easy-to-remove, while a test that can be removed only when $n_{max}$ is high is considered hard-to-remove. The removal of a hard-to-remove test may imply that other tests will not be removed, while the removal of an easy-to-remove test may leave the flexibility to remove other tests. Increasing $n_{max}$ gradually prevents hard-to-remove tests from being removed early in the process, and potentially allows more tests to be removed.

The procedure computes $T_a$ for a given set $T_s$ as follows. Initially, $T_a = \emptyset$. The procedure performs fault simulation with fault dropping of $F$ under $T_s$. It then considers every $t_i \in T_s$, $0 \le n_j \le n_{max}$ and $\beta \in \{0, 1\}$. It performs fault simulation with fault dropping of $F$ under the test corresponding to $(i, n_j, \beta)$. If the test detects any faults, the triple $(i, n_j, \beta)$ is added to $T_a$.

In the example of $b01$, using $n_{max} = 0$ does not allow any test to be removed from $T_s$. Using $n_{max} = 1$, the tests $t_0$, $t_1$, $t_2$, $t_4$, $t_5$, $t_6$, $t_7$, $t_{12}$ and $t_{14}$ are removed from $T_s$. The resulting set $T_s$ is the one shown in the first part of Table II, and $T_a$ is shown in the second part of Table II.

TABLE III

EXPERIMENTAL RESULTS (ISCAS-89)

| circuit | nmax | stor | appl | incr | bits | frac | f.c. | n.time |
|---|---|---|---|---|---|---|---|---|
| s5378 | init | 196 | 196 | 1.00 | 42140 | 1.00 | 79.59 | 1.00 |
| s5378 | 1 | 182 | 197 | 1.01 | 39280 | 0.93 | 79.59 | 616.43 |
| s5378 | 2 | 173 | 204 | 1.04 | 37536 | 0.89 | 79.59 | 1019.43 |
| s5378 | 3 | 168 | 210 | 1.07 | 36582 | 0.87 | 79.59 | 1518.43 |
| s5378 | 4 | 160 | 218 | 1.11 | 35096 | 0.83 | 79.59 | 2090.14 |
| s5378 | 5 | 155 | 221 | 1.13 | 34117 | 0.81 | 79.59 | 2699.57 |
| s5378 | 6 | 149 | 228 | 1.16 | 32983 | 0.78 | 79.59 | 3335.86 |
| s5378 | 7 | 142 | 233 | 1.19 | 31622 | 0.75 | 79.59 | 4005.71 |
| s5378 | 8 | 140 | 238 | 1.21 | 31374 | 0.74 | 79.59 | 4705.71 |
| s5378 | 9 | 136 | 245 | 1.25 | 30657 | 0.73 | 79.59 | 5489.00 |
| s5378 | 10 | 134 | 245 | 1.25 | 30253 | 0.72 | 79.59 | 6290.57 |
| s5378 | 11 | 133 | 246 | 1.26 | 30064 | 0.71 | 79.59 | 7129.00 |
| s5378 | 12 | 131 | 244 | 1.24 | 29634 | 0.70 | 79.59 | 8065.71 |
| s5378 | 13 | 129 | 243 | 1.24 | 29217 | 0.69 | 79.59 | 9016.14 |
| s5378 | 14 | 128 | 241 | 1.23 | 28876 | 0.69 | 79.59 | 9940.00 |
| s5378 | 15 | 127 | 245 | 1.25 | 28721 | 0.68 | 79.59 | 10933.71 |
| s9234 | init | 239 | 239 | 1.00 | 59272 | 1.00 | 79.02 | 1.00 |
| s9234 | 1 | 215 | 247 | 1.03 | 53640 | 0.90 | 79.13 | 1073.39 |
| s9234 | 2 | 205 | 258 | 1.08 | 51423 | 0.87 | 79.14 | 1865.48 |
| s9234 | 3 | 199 | 263 | 1.10 | 50056 | 0.84 | 79.15 | 2808.87 |
| s9234 | 4 | 197 | 266 | 1.11 | 49684 | 0.84 | 79.15 | 4343.74 |
| s9234 | 5 | 191 | 268 | 1.12 | 48292 | 0.81 | 79.15 | 5673.00 |
| s9234 | 6 | 188 | 272 | 1.14 | 47632 | 0.80 | 79.15 | 7297.22 |
| s9234 | 7 | 182 | 275 | 1.15 | 46252 | 0.78 | 79.15 | 8911.00 |
| s9234 | 8 | 180 | 279 | 1.17 | 45927 | 0.77 | 79.19 | 10486.96 |
| s13207 | init | 396 | 396 | 1.00 | 277596 | 1.00 | 89.02 | 1.00 |
| s13207 | 1 | 357 | 399 | 1.01 | 250719 | 0.90 | 89.02 | 1654.81 |
| s13207 | 2 | 343 | 402 | 1.02 | 241151 | 0.87 | 89.02 | 2868.10 |
| s13207 | 3 | 329 | 405 | 1.02 | 231541 | 0.83 | 89.02 | 4374.87 |
| s13207 | 4 | 320 | 409 | 1.03 | 225477 | 0.81 | 89.02 | 6066.16 |
| s13207 | 5 | 315 | 410 | 1.04 | 222050 | 0.80 | 89.02 | 7989.35 |
| s13207 | 6 | 313 | 412 | 1.04 | 220700 | 0.80 | 89.02 | 10165.29 |
| s15850 | init | 243 | 243 | 1.00 | 148716 | 1.00 | 87.75 | 1.00 |
| s15850 | 1 | 219 | 261 | 1.07 | 134448 | 0.90 | 87.94 | 1053.59 |
| s15850 | 2 | 210 | 264 | 1.09 | 129114 | 0.87 | 87.95 | 1843.57 |
| s15850 | 3 | 205 | 272 | 1.12 | 126197 | 0.85 | 87.95 | 2727.09 |
| s15850 | 4 | 204 | 271 | 1.12 | 125652 | 0.84 | 87.95 | 3861.74 |
| s15850 | 5 | 198 | 277 | 1.14 | 122124 | 0.82 | 87.95 | 5413.13 |
| s15850 | 6 | 193 | 285 | 1.17 | 119220 | 0.80 | 87.95 | 6820.89 |
| s15850 | 7 | 191 | 290 | 1.19 | 118080 | 0.79 | 87.95 | 8461.41 |
| s15850 | 8 | 186 | 298 | 1.23 | 115288 | 0.78 | 87.95 | 10222.96 |
| s35932 | init | 29 | 29 | 1.00 | 51156 | 1.00 | 73.49 | 1.00 |
| s35932 | 1 | 23 | 44 | 1.52 | 40719 | 0.80 | 73.49 | 94.32 |
| s35932 | 2 | 19 | 50 | 1.72 | 33764 | 0.66 | 73.49 | 147.65 |
| s35932 | 3 | 17 | 53 | 1.83 | 30276 | 0.59 | 73.49 | 195.80 |
| s35932 | 4 | 15 | 53 | 1.83 | 26764 | 0.52 | 73.49 | 239.64 |
| s35932 | 5 | 12 | 58 | 2.00 | 21536 | 0.42 | 73.49 | 287.59 |
| s35932 | 7 | 11 | 60 | 2.07 | 19796 | 0.39 | 73.49 | 372.43 |
| s35932 | 8 | 10 | 64 | 2.21 | 18126 | 0.35 | 73.49 | 406.17 |
| s35932 | 10 | 9 | 67 | 2.31 | 16398 | 0.32 | 73.49 | 482.38 |
| s35932 | 11 | 9 | 64 | 2.21 | 16371 | 0.32 | 73.49 | 531.30 |
| s35932 | 13 | 8 | 67 | 2.31 | 14584 | 0.29 | 73.49 | 608.20 |
| s35932 | 14 | 8 | 66 | 2.28 | 14576 | 0.28 | 73.49 | 655.61 |
| s35932 | 16 | 7 | 72 | 2.48 | 12933 | 0.25 | 73.49 | 741.12 |
| s35932 | 17 | 7 | 71 | 2.45 | 12924 | 0.25 | 73.49 | 779.01 |
| s35932 | 18 | 7 | 70 | 2.41 | 12915 | 0.25 | 73.49 | 814.28 |
| s35932 | 22 | 6 | 73 | 2.52 | 11187 | 0.22 | 73.49 | 971.93 |
| s35932 | 23 | 6 | 72 | 2.48 | 11178 | 0.22 | 73.49 | 1013.38 |
| s35932 | 25 | 6 | 71 | 2.45 | 11169 | 0.22 | 73.49 | 1122.13 |
| s35932 | 33 | 5 | 71 | 2.45 | 9480 | 0.19 | 73.49 | 1470.22 |
| s35932 | 63 | 5 | 68 | 2.34 | 9450 | 0.18 | 73.49 | 3081.81 |
| s38584 | init | 344 | 344 | 1.00 | 503960 | 1.00 | 79.18 | 1.00 |
| s38584 | 1 | 315 | 356 | 1.03 | 461926 | 0.92 | 79.18 | 1250.66 |
| s38584 | 2 | 307 | 367 | 1.07 | 450475 | 0.89 | 79.18 | 2178.41 |
| s38584 | 3 | 298 | 387 | 1.12 | 437638 | 0.87 | 79.18 | 3396.44 |
| s38584 | 4 | 290 | 409 | 1.19 | 426397 | 0.85 | 79.18 | 4883.94 |
| s38584 | 5 | 285 | 417 | 1.21 | 419241 | 0.83 | 79.18 | 6706.18 |
| s38584 | 6 | 276 | 436 | 1.27 | 406420 | 0.81 | 79.19 | 9024.01 |
| s38584 | 7 | 270 | 455 | 1.32 | 397955 | 0.79 | 79.19 | 11442.04 |

TABLE IV

EXPERIMENTAL RESULTS (ITC-99)

| circuit | nmax | stor | appl | incr | bits | frac | f.c. | n.time |
|---|---|---|---|---|---|---|---|---|
| b14 | init | 214 | 214 | 1.00 | 60134 | 1.00 | 83.17 | 1.00 |
| b14 | 1 | 194 | 229 | 1.07 | 54864 | 0.91 | 83.18 | 964.93 |
| b14 | 2 | 188 | 239 | 1.12 | 53389 | 0.89 | 83.18 | 1645.73 |
| b14 | 3 | 184 | 244 | 1.14 | 52364 | 0.87 | 83.18 | 2418.37 |
| b14 | 4 | 181 | 250 | 1.17 | 51689 | 0.86 | 83.18 | 3407.07 |
| b14 | 5 | 176 | 254 | 1.19 | 50392 | 0.84 | 83.18 | 4495.10 |
| b14 | 6 | 169 | 260 | 1.21 | 48581 | 0.81 | 83.18 | 5707.73 |
| b14 | 7 | 164 | 267 | 1.25 | 47320 | 0.79 | 83.18 | 6964.60 |
| b14 | 8 | 160 | 273 | 1.28 | 46429 | 0.77 | 83.18 | 8408.37 |
| b14 | 9 | 152 | 280 | 1.31 | 44376 | 0.74 | 83.18 | 9712.20 |
| b14 | 10 | 148 | 284 | 1.33 | 43356 | 0.72 | 83.18 | 11128.53 |
| b20 | init | 207 | 207 | 1.00 | 109296 | 1.00 | 85.91 | 1.00 |
| b20 | 1 | 200 | 214 | 1.03 | 105740 | 0.97 | 85.96 | 919.52 |
| b20 | 2 | 194 | 216 | 1.04 | 102674 | 0.94 | 85.96 | 1483.21 |
| b20 | 3 | 188 | 224 | 1.08 | 99660 | 0.91 | 85.96 | 2260.35 |
| b20 | 4 | 184 | 231 | 1.12 | 97716 | 0.89 | 85.96 | 3104.53 |
| b20 | 5 | 182 | 231 | 1.12 | 96684 | 0.88 | 85.96 | 4044.78 |
| b20 | 6 | 176 | 260 | 1.26 | 93936 | 0.86 | 85.96 | 4915.22 |
| b20 | 7 | 174 | 277 | 1.34 | 93108 | 0.85 | 85.96 | 5822.58 |
| b20 | 8 | 170 | 285 | 1.38 | 91255 | 0.83 | 85.96 | 6930.07 |
| b20 | 9 | 165 | 308 | 1.49 | 88979 | 0.81 | 85.96 | 8050.94 |
| b20 | 10 | 162 | 306 | 1.48 | 87408 | 0.80 | 85.96 | 9273.76 |
| b20 | 11 | 159 | 311 | 1.50 | 85928 | 0.79 | 85.96 | 10558.11 |

After obtaining $T_a$, the procedure applies forward-looking reverse order fault simulation to $T_a$. This process removes from $T_a$ tests that are not necessary for detecting any fault.

The computational complexity of the procedure is determined by the need to compute $T_a$ every time an attempt is made to remove a test from $T_s$. This requires fault simulation with fault dropping of at most $2|T|(n_{max}+1)$ tests, which is repeated at most $|T|$ times. However, not all the faults need to be simulated when an attempt is made to remove a test. Faults that were detected earlier based on different tests are still detected and do not need to be simulated. The corresponding tests need to be included in $T_a$ even if they are not simulated (the procedure was implemented without using this speedup technique, and without using any of the speedup techniques that are typically used for fault simulation such as parallel fault or parallel pattern simulation). Consideration of $0 \leq n_{max} \leq K$ results in fault simulation of at most $2|T|(K+1)^2$ tests. When $K$ is large, the procedure can be run with a constant bound on $n_{max}$, or with a bound on the run time. Let $RT_{init}$ be the run time for fault simulation with fault dropping of the test set $T$. Let $RT(n)$ be the run time of the procedure for $0 \leq n_{max} \leq n$. The ratio $RT(n)/RT_{init}$ can be used for measuring the computational effort of the procedure. In the next section, the procedure is terminated based on this ratio, which is referred to as its normalized run time.

## V. EXPERIMENTAL RESULTS

This section presents the results of the procedure from Section IV when applied to transition faults in ISCAS-89, ITC-99, and IWLS-05 benchmark circuits.

The test set $T$ to which the procedure was applied is a compacted skewed-load test set for transition faults. The test set achieves the highest or close to the highest transition fault coverage that can be achieved by skewed-load tests with the constraint of constant primary input vectors. The procedure described in this paper can be applied without this constraint by replacing $v_i$ with a pair of primary input vectors $(v_i, u_i)$.

The results are shown in Tables III–V as follows. There are several rows for every circuit. The first row corresponds to the test set $T$. As discussed in Section IV, the procedure increases $n_{max}$ gradually. The next rows correspond to increasing values of $n_{max}$ for which reductions in the storage requirements were achieved. The procedure was terminated if its normalized run time (the ratio $RT(n)/RT_{init}$) exceeded 10 000.

Each row contains the following information. Column *nmax* shows the value of $n_{max}$, or the entry *init* in the case of the test set $T$. Column *stor* shows the number of tests that need to be stored. This is the number of tests in $T$ or $T_s$. Column *appl* shows the number of tests that need to be applied. This is the number of tests in $T$ or $T_s \cup T_a$. Column *incr* shows the increase in the number of applied tests relative to the number of tests in $T$.

Column *bits* shows the number of bits required for storing $T$, or $T_s$ and $T_a$. Column *frac* shows the number of bits as a fraction of the number of bits required for storing $T$.

TABLE V
EXPERIMENTAL RESULTS (IWLS-05)

| circuit | nmax | stor | appl | incr | bits | frac | f.c. | n.time |
|---|---|---|---|---|---|---|---|---|
| spi | init | 500 | 500 | 1.00 | 137500 | 1.00 | 90.45 | 1.00 |
| spi | 1 | 442 | 539 | 1.08 | 122617 | 0.89 | 90.46 | 1058.70 |
| spi | 2 | 386 | 588 | 1.18 | 108574 | 0.79 | 90.46 | 1710.83 |
| spi | 3 | 340 | 646 | 1.29 | 97172 | 0.71 | 90.47 | 2316.22 |
| spi | 4 | 308 | 655 | 1.31 | 89211 | 0.65 | 90.47 | 2952.09 |
| spi | 5 | 285 | 685 | 1.37 | 83575 | 0.61 | 90.47 | 3577.74 |
| spi | 6 | 269 | 690 | 1.38 | 79448 | 0.58 | 90.47 | 4245.35 |
| spi | 7 | 259 | 717 | 1.43 | 77179 | 0.56 | 90.47 | 4886.87 |
| spi | 8 | 235 | 740 | 1.48 | 71190 | 0.52 | 90.47 | 5596.26 |
| spi | 9 | 225 | 747 | 1.49 | 68661 | 0.50 | 90.47 | 6258.13 |
| spi | 10 | 216 | 749 | 1.50 | 66329 | 0.48 | 90.47 | 6893.57 |
| spi | 11 | 205 | 753 | 1.51 | 63499 | 0.46 | 90.47 | 7536.57 |
| spi | 12 | 197 | 763 | 1.53 | 61533 | 0.45 | 90.47 | 8224.13 |
| spi | 13 | 189 | 751 | 1.50 | 59281 | 0.43 | 90.47 | 8956.17 |
| spi | 14 | 185 | 752 | 1.50 | 58246 | 0.42 | 90.47 | 9644.83 |
| spi | 15 | 176 | 752 | 1.50 | 55888 | 0.41 | 90.47 | 10345.00 |
| systemcaes | init | 156 | 156 | 1.00 | 144924 | 1.00 | 95.40 | 1.00 |
| systemcaes | 1 | 147 | 162 | 1.04 | 136713 | 0.94 | 95.40 | 323.34 |
| systemcaes | 2 | 138 | 176 | 1.13 | 128620 | 0.89 | 95.40 | 487.13 |
| systemcaes | 3 | 128 | 210 | 1.35 | 119732 | 0.83 | 95.40 | 668.06 |
| systemcaes | 4 | 122 | 213 | 1.37 | 114339 | 0.79 | 95.40 | 839.66 |
| systemcaes | 5 | 115 | 223 | 1.43 | 108023 | 0.75 | 95.40 | 1016.19 |
| systemcaes | 9 | 91 | 278 | 1.78 | 86783 | 0.60 | 95.40 | 2021.45 |
| systemcaes | 14 | 76 | 304 | 1.95 | 73340 | 0.51 | 95.40 | 3062.49 |
| systemcaes | 19 | 67 | 339 | 2.17 | 65779 | 0.45 | 95.40 | 4056.55 |
| systemcaes | 23 | 64 | 335 | 2.15 | 62708 | 0.43 | 95.40 | 5027.96 |
| systemcaes | 29 | 57 | 351 | 2.25 | 56481 | 0.39 | 95.40 | 6264.13 |
| systemcaes | 33 | 55 | 349 | 2.24 | 54917 | 0.38 | 95.40 | 7135.28 |
| systemcaes | 38 | 51 | 351 | 2.25 | 51279 | 0.35 | 95.40 | 8117.28 |
| systemcaes | 45 | 47 | 356 | 2.28 | 47680 | 0.33 | 95.40 | 9587.19 |
| systemcaes | 47 | 46 | 360 | 2.31 | 46816 | 0.32 | 95.40 | 10070.23 |
| systemcdes | init | 79 | 79 | 1.00 | 25359 | 1.00 | 96.32 | 1.00 |
| systemcdes | 1 | 75 | 85 | 1.08 | 24165 | 0.95 | 96.32 | 188.83 |
| systemcdes | 2 | 67 | 121 | 1.53 | 22047 | 0.87 | 96.32 | 276.00 |
| systemcdes | 3 | 60 | 142 | 1.80 | 19998 | 0.79 | 96.32 | 358.83 |
| systemcdes | 4 | 53 | 157 | 1.99 | 18053 | 0.71 | 96.32 | 443.00 |
| systemcdes | 5 | 46 | 166 | 2.10 | 15966 | 0.63 | 96.32 | 528.67 |
| systemcdes | 6 | 41 | 174 | 2.20 | 14491 | 0.57 | 96.32 | 609.17 |
| systemcdes | 7 | 34 | 179 | 2.27 | 12364 | 0.49 | 96.32 | 679.00 |
| systemcdes | 8 | 32 | 177 | 2.24 | 11722 | 0.46 | 96.32 | 738.17 |
| systemcdes | 9 | 30 | 182 | 2.30 | 11150 | 0.44 | 96.32 | 792.83 |
| systemcdes | 10 | 27 | 193 | 2.44 | 10327 | 0.41 | 96.32 | 844.83 |
| systemcdes | 11 | 26 | 192 | 2.43 | 10006 | 0.39 | 96.32 | 929.33 |
| systemcdes | 12 | 24 | 189 | 2.39 | 9354 | 0.37 | 96.32 | 983.50 |
| systemcdes | 13 | 23 | 194 | 2.46 | 9093 | 0.36 | 96.32 | 1039.83 |
| systemcdes | 50 | 11 | 199 | 2.52 | 5599 | 0.22 | 96.32 | 2286.00 |
| systemcdes | 61 | 9 | 203 | 2.57 | 5023 | 0.20 | 96.32 | 2649.67 |
| wb_dma | init | 160 | 160 | 1.00 | 118240 | 1.00 | 82.30 | 1.00 |
| wb_dma | 1 | 139 | 168 | 1.05 | 103011 | 0.87 | 82.31 | 424.62 |
| wb_dma | 2 | 133 | 171 | 1.07 | 98705 | 0.83 | 82.31 | 675.50 |
| wb_dma | 3 | 130 | 173 | 1.08 | 96543 | 0.82 | 82.31 | 938.62 |
| wb_dma | 4 | 127 | 183 | 1.14 | 94469 | 0.80 | 82.31 | 1224.75 |
| wb_dma | 5 | 124 | 185 | 1.16 | 92307 | 0.78 | 82.31 | 1552.75 |
| wb_dma | 6 | 123 | 188 | 1.18 | 91612 | 0.77 | 82.31 | 1906.50 |
| wb_dma | 7 | 122 | 189 | 1.18 | 90895 | 0.77 | 82.31 | 2271.62 |
| wb_dma | 8 | 120 | 188 | 1.18 | 89496 | 0.76 | 82.31 | 2686.88 |
| wb_dma | 9 | 118 | 188 | 1.18 | 88042 | 0.74 | 82.31 | 3120.25 |
| wb_dma | 10 | 118 | 185 | 1.16 | 88006 | 0.74 | 82.31 | 3549.38 |
| wb_dma | 11 | 114 | 190 | 1.19 | 85158 | 0.72 | 82.31 | 4002.50 |
| wb_dma | 12 | 113 | 189 | 1.18 | 84419 | 0.71 | 82.31 | 4558.25 |
| wb_dma | 13 | 109 | 196 | 1.23 | 81595 | 0.69 | 82.31 | 5042.50 |
| wb_dma | 14 | 106 | 198 | 1.24 | 79438 | 0.67 | 82.31 | 5612.75 |
| wb_dma | 15 | 105 | 198 | 1.24 | 78711 | 0.67 | 82.31 | 6171.75 |
| wb_dma | 16 | 104 | 199 | 1.24 | 78091 | 0.66 | 82.31 | 6789.25 |
| wb_dma | 17 | 102 | 207 | 1.29 | 76743 | 0.65 | 82.31 | 7372.00 |
| wb_dma | 19 | 100 | 208 | 1.30 | 75304 | 0.64 | 82.31 | 8646.25 |
| wb_dma | 20 | 99 | 207 | 1.29 | 74565 | 0.63 | 82.31 | 9268.50 |
| wb_dma | 21 | 99 | 206 | 1.29 | 74552 | 0.63 | 82.31 | 9843.25 |
| wb_dma | 22 | 97 | 212 | 1.32 | 73178 | 0.62 | 82.31 | 10457.00 |

Column *f.c.* shows the transition fault coverage. In some cases, where $T$ does not detect all the detectable transition faults, the transition fault coverage is increased as $n_{max}$ is increased. This occurs accidentally due to the use of $T_a$. The procedure is guaranteed not to reduce the fault coverage.

Column *n.time* shows the normalized run time.

The following points can be seen from Tables III–V. The procedure from Section IV reduces the number of tests in $T_s$, and the storage requirements, as $n_{max}$ is increased. The reduction is significant for most of the circuits considered. This reduction is achieved on top of any reductions that are achieved by storing compressed tests.

The first reduction in storage requirements, which is achieved with $n_{max} = 0$ or 1, may be low. This is because $T$ is highly compacted. A higher value of $n_{max}$ is needed to achieve a significant reduction in storage requirements. However, even this value is significantly smaller than the number of state variables, $K$.

The number of applied tests increases as $n_{max}$ is increased. By increasing $n_{max}$ gradually it is possible to select a solution with a bounded number of applied tests and reduced storage requirements.

Finally, a solution with lower storage requirements may also require a smaller number of applied tests. This occurs because of the use of fault simulation with fault dropping to define $T_a$ (instead of using a set covering procedure, for example).

## VI. CONCLUSION

This paper introduced the possibility of expanding a stored skewed-load test into several different skewed-load tests by shifting the scan-in state for one or more additional clock cycles. Based on this expansion, the paper described a procedure for input test data volume reduction that starts from a given skewed-load test set. The procedure removed tests from the test set and recovered the fault coverage by applying several tests based on every stored test. Experimental results demonstrated significant reductions in the input test data volume when the procedure was applied to compact skewed-load test sets for transition faults in benchmark circuits. These reductions are obtained on top of the reductions that can be achieved with test data compression methods that use compressed scan-in states. The method does not require any on-chip hardware support. It requires a tester where it is possible to adjust the number of scan-in cycles for a test based on the stored data.

## REFERENCES


[1] K.-J. Lee, J. J. Chen, and C. H. Huang, "Using a single input to support multiple scan chains," in *Proc. Int. Conf. Computer-Aided Design*, 1998, pp. 74–78.

[2] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller, and B. Koenemann, "OPMISR: The foundation for compressed ATPG vectors," in *Proc. Int. Test Conf.*, 2001, pp. 748–757.

[3] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K.-H. Tsai, *et al.*, "Embedded deterministic test for low cost manufacturing test," in *Proc. Intl. Test Conf.*, 2002, pp. 301–310.

[4] N. A. Touba, "Survey of test vector compression techniques," in *Proc. IEEE Design Test*, Apr. 2006, pp. 294–303.

[5] R. Dandapani, J. H. Patel, and J. A. Abraham, "Design of test pattern generation for built-in test," in *Proc. Int. Test Conf.*, 1984, pp. 315–319.

[6] K.-H. Tsai, J. Rajski, and M. Marek-Sadowska, "Scan encoded test pattern generation for BIST," in *Proc. Int. Test Conf.*, 1997, pp. 548–556.

[7] I. Pomeranz, "On the computation of common test data for broadside and skewed-load tests," *IEEE Trans. Comput*, vol. 61, no. 4, pp. 578–583, Apr. 2012.

[8] N. Tendolkar, R. Raina, R. Woltenberg, X. Lin, B. Swanson, and G. Aldrich, "Novel techniques for achieving high at-speed transition fault test coverage for Motorola's microprocessors based on PowerPC instruction set architecture," in *Proc. VLSI Test Symp.*, 2002, pp. 3–8.

[9] Y. Shao, I. Pomeranz, and S. M. Reddy, "On generating high quality tests for transition faults," in *Proc. Asian Test Symp.*, 2002, pp. 1–8.

[10] W. Qiu, J. Wang, D. M. H. Walker, D. Reddy, X. Lu, Z. Li, *et al.*, "K longest paths per gate (KLPG) test generation for scan-based sequential circuits," in *Proc. Int. Test Conf.*, 2004, pp. 223–231.

[11] Z. Chen, D. Xiang, and B. Yin, "The ATPG conflict-driven scheme for high transition fault coverage and low test cost," in *Proc. VLSI Test Symp.*, 2009, pp. 146–151.