Equivalent Faults under Launch-on-Shift (LOS) Tests with Equal Primary Input Vectors

IRITH POMERANZ, Purdue University

A recent work showed that it is possible to transform a single-cycle test for stuck-at faults into a launch-on-shift (LOS) test that is guaranteed to detect the same stuck-at faults without any logic or fault simulation. The LOS test also detects transition faults. This was used for obtaining a compact LOS test set that detects both types of faults. In the scenario where LOS tests are used for both stuck-at and transition faults, this paper observes that, under certain conditions, the detection of a stuck-at fault guarantees the detection of a corresponding transition fault. This implies that the two faults are equivalent under LOS tests. Equivalence can be used for reducing the set of target faults for test generation and test compaction. The paper develops this notion of equivalence under LOS tests with equal primary input vectors, and provides an efficient procedure for identifying it. It presents experimental results to demonstrate that such equivalences exist in benchmark circuits, and shows an unexpected effect on a test compaction procedure.

CCS Concepts: •**Hardware** →**Test-pattern generation and fault simulation;**

Additional Key Words and Phrases: Equivalent faults, launch-on-shift tests, test compaction, test generation, transition faults.

ACM Reference format:
Irith Pomeranz. 2020. Equivalent Faults under Launch-on-Shift (LOS) Tests with Equal Primary Input Vectors. *ACM Trans. Des. Autom. Electron. Syst.* 11, 11, Article 11 (November 2020), **??** pages.
DOI: 0000001.0000001

1 INTRODUCTION

Test generation procedures produce different types of tests, and target different fault models. This is needed since defects exhibit different behaviors, and may be modeled by different faults, and detected by different types of tests [1]-[11]. For example, cell-aware faults consist of both static and dynamic (or delay) faults [8]-[10]. Single-cycle tests may be generated for static faults, and two-cycle launch-on-capture (LOC) or launch-on-shift (LOS) tests may be generated for dynamic faults. LOS tests were defined in [12], and LOC tests were defined in [13]. Considering transition faults, more faults are typically detectable by LOS than LOC tests, and fewer LOS tests are typically required for achieving the same fault coverage [12]-[13]. In a compact test set that consists of both test types, more LOS than LOC tests are typically included to achieve a higher fault coverage using fewer tests [6]. The requirement of a LOS test for a scan enable input that changes at-speed was addressed in [14]-[15].

Author's address: Irith Pomeranz, 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.

DOI: 0000001.0000001

A recent work showed that a single-cycle test for static faults can be transformed into a LOS test that is guaranteed to detect the same static faults [16]. This is achieved without any logic or fault simulation. The LOS test also detects delay faults, while the single-cycle test detects only static faults. The importance of delay faults results from the frequent occurrence of delay defects in state-of-the-art technologies. Delay defects are modeled by delay faults, and tested by tests for delay faults [17]-[28].

The transformation of single-cycle tests into LOS tests was used in [16] for obtaining a compact LOS test set that detects both static and dynamic faults (single stuck-at faults and transition faults were considered in [16]). The test compaction procedure from [16] starts from a compact LOS test set for transition faults, and a compact single-cycle test set for stuck-at faults that are not detected by the transition fault test set. It is able to achieve additional test compaction by eliminating single-cycle tests, and using only LOS tests to detect both types of faults. The advantage of using the same tests to detect several types of faults was also noted in [1], [2], [3] and [11].

This paper considers the scenario where LOS tests are used for both stuck-at and transition faults (or in general, for both static and delay faults) in order to support test compaction beyond that possible if both single-cycle and LOS tests are used. Throughout the discussion, only LOS tests with equal primary input vectors are considered. The paper observes that, under this scenario, and if certain conditions are satisfied, the detection of a stuck-at fault guarantees the detection of a corresponding transition fault. The fact that a LOS test that detects a transition fault also detects a corresponding stuck-at fault is known. When the reverse is true, the two faults are equivalent under LOS tests. Equivalence can be used for reducing the set of faults targeted for test generation and test compaction. This is similar to fault collapsing [29], but with the difference that it considers faults from different models, and under a specific test type (LOS tests). It should be noted in this regard that fault collapsing for single stuck-at faults reduces the number of faults significantly, but only limited fault collapsing is possible for transition faults. Specifically, for an n-input gate, the number of single stuck-at faults after fault collapsing is $n + 2$ instead of $2(n + 1)$, but no similar reduction is possible for transition faults.

The paper defines the notion of equivalence between corresponding stuck-at and transition faults under LOS tests with equal primary input vectors, and describes an efficient procedure for identifying it. It also presents two types of experimental results to demonstrate that equivalences between corresponding stuck-at and transition faults under LOS tests exist in benchmark circuits. The first experiment uses exhaustive test sets to achieve completeness in the sense that it identifies all the pairs of equivalent faults. The circuits for this experiment are small finite-state machine benchmarks. The second experiment considers larger benchmark circuits for which exhaustive test sets are prohibitive. For these circuits, an efficient procedure is used for identifying a subset of all the equivalences between corresponding stuck-at and transition faults. The paper also shows an unexpected effect on the test compaction procedure from [16] when equivalences are used for reducing the set of target faults.

Throughout the discussion, if two faults are determined to be equivalent, the determination is accurate in the sense that the faults are guaranteed to be detected by the same tests considering LOS tests with equal primary input vectors. Experimental results for benchmark circuits show that the percentage of equivalences found varies with the circuit. The effect on the test compaction procedure from [16] is visible when the percentage of equivalences is larger than 5%. Finally, the resulting LOS test set can be complemented by LOC tests to increase the fault coverage.

The paper is organized as follows. Section 2 reviews the transformation of a single-cycle test into a LOS test. Section 3 provides the conditions under which a stuck-at fault and a corresponding transition fault are equivalent. Section 4 presents experimental results to demonstrate the existence

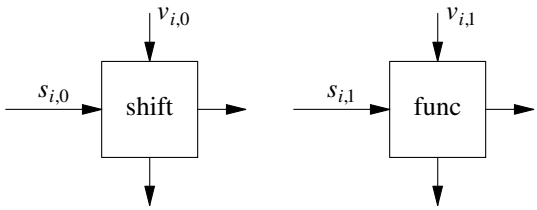

Fig. 1. LOS test

of equivalences between corresponding faults in benchmark circuits. Section 5 describes an efficient procedure for identifying equivalences, and presents experimental results of this procedure. Section 6 discusses an unexpected effect of equivalences on the test compaction procedure from [16].

2 TEST TRANSFORMATION

This section reviews the transformation of a single-cycle test into a LOS test.

A single-cycle test $\langle p_i, u_i \rangle$ consists of a scan-in state p_i and a primary input vector u_i. After p_i is scanned in, the circuit is clocked in functional mode with u_i applied to the primary inputs. The primary output vector is observed, and the next-state is latched in the flip-flops. The state is then scanned out.

An LOS test $\langle s_{i,0}, v_{i,0}; s_{i,1}, v_{i,1} \rangle$ is illustrated by Figure 1. The test has two clock cycles between a scan-in and a scan-out operation. In the first clock cycle after the scan-in operation, the circuit is in state $s_{i,0}$ and the primary input vector $v_{i,0}$ is applied. The circuit is brought to state $s_{i,0}$ by a scan-in operation, and the first clock cycle is applied in scan shift mode. In the second clock cycle the circuit is in state $s_{i,1}$ and the primary input vector $v_{i,1}$ is applied. The circuit is clocked in functional mode. The primary output vector is observed, and the next-state is latched in the flip-flops. The state is then scanned out.

The state $s_{i,1}$ is obtained by a single shift of the state $s_{i,0}$. This requires a scan-in vector to be specified for the first clock cycle of the test. The scan-in vector appears in $s_{i,1}$, and it is not specified separately. For example, considering a circuit with five state variables in a single scan chain that is shifted to the right, the states $s_{i,0} = 00111$ and $s_{i,1} = 10011$ imply that the scan-in vector 1 is used.

An important property of a LOS test for faults in the combinational logic is the following. Even if a fault is activated in the first clock cycle of the test, and propagated to the next-state variables, the first clock cycle does not latch the values coming from the combinational logic. Therefore, the fault effects are not propagated to the second clock cycle. Consequently, the circuit is guaranteed to be in state $s_{i,1}$ during the second clock cycle even in the presence of faults.

Based on this discussion, suppose that a single-cycle test $\langle p_i, u_i \rangle$ is embedded in the second cycle of a LOS test to create a test of the form $\langle s_{i,0}, v_{i,0}; p_i, u_i \rangle$. In this test, $s_{i,0}$ is selected such that p_i is obtained after a single shift of the scan chains. For consistency with current practices, let $v_{i,0} = u_i$. In this case, the LOS test is guaranteed to detect every single stuck-at fault that the single-cycle test detects. In addition, the LOS test detects transition faults that are not detected by the single-cycle test.

For example, considering the same circuit as before, let the single-cycle test be $\langle 10011, 000 \rangle$. The corresponding LOS test is either $\langle 00111, 000; 10011, 000 \rangle$ or $\langle 00110, 000; 10011, 000 \rangle$. The first option is illustrated by Figure 2.

For ease of discussion, all the circuits considered in this paper are assumed to have a single scan chain that is shifted to the right as in the example above.

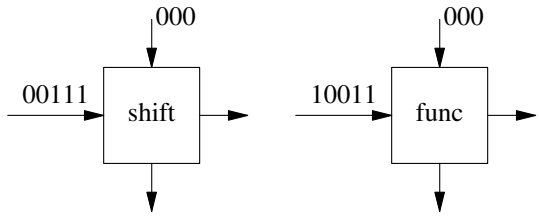

Fig. 2. Transformed LOS test

3 EQUIVALENCE UNDER LOS TESTS

This section defines equivalence between stuck-at and transition faults under LOS tests. Although equivalence can exist between any two faults, the paper focuses on equivalence between corresponding stuck-at and transition faults as discussed next. This limits the number of fault pairs that need to be considered, and simplifies the procedure for identifying equivalences.

The stuck-at fault where line g is stuck-at the value a is denoted by g/a. The transition fault that delays the $a \rightarrow a'$ transition on line g is denoted by $g/a \rightarrow a'$. The faults g/a and $g/a \rightarrow a'$ are referred to as corresponding since a test that detects $g/a \rightarrow a'$ is guaranteed to detect the fault g/a. This is a known relationship, and it can be explained as follows.

Let $\langle s_{i,0}, v_{i,0}; s_{i,1}, v_{i,1} \rangle$ be a two-cycle test that detects the transition fault $g/a \rightarrow a'$. This implies that the test assigns $g = a$ in the first clock cycle and $g = a'$ in the second clock cycle. This creates the fault effect $g = a'/a$ in the second clock cycle. The fault effect is propagated to an observable output in the second clock cycle for the fault to be detected.

With $g = a$ in the first clock cycle, the test does not activate the stuck-at fault g/a. It activates the fault in the second clock cycle by assigning $g = a'$. In the presence of the stuck-at fault this creates the fault effect $g = a'/a$. As in the case of the transition fault, the fault effect is propagated to an observable output in the second clock cycle, and the fault is detected.

The goal of this paper is to identify cases where every LOS test that detects the stuck-at fault g/a also detects the corresponding transition fault $g/a \rightarrow a'$. When this occurs, the two faults are said to be equivalent. Equivalence in this case has the following implications.

(1) It only applies to LOS tests with equal primary input vectors. The faults are not equivalent if single-cycle tests are used for stuck-at faults, if LOC tests are considered, or if the primary input vectors of a LOS test are allowed to be different. The latter assumption can be removed by allowing $v_{i,0} \neq u_i$ to be selected during the transformation of a single-cycle test into a LOS test.

(2) Although the output values that the faults produce are not considered explicitly, the output values in the second clock cycle of a LOS test are equal when the faults are equivalent.

The conditions under which every LOS test that detects the stuck-at fault g/a also detects the corresponding transition fault $g/a \rightarrow a'$ are discussed next.

Let $\langle p_i, u_i \rangle$ be an arbitrary single-cycle test that detects the fault g/a. When the test is transformed into a LOS test, one of two tests are obtained, $\langle s^0_{i,0}, u_i; p_i, u_i \rangle$, where the last bit of the scan-in state is 0, or $\langle s^1_{i,0}, u_i; p_i, u_i \rangle$, where the last bit of the scan-in state is 1. Suppose that in both cases, $g = a$ is obtained in the first cycle of the test. Moreover, suppose that this applies to every single-cycle test $\langle p_i, u_i \rangle$ that detects g/a. In this case, every LOS test for g/a also detects $g/a \rightarrow a'$, establishing the equivalence between the faults.

A direct application of this condition is considered in Section 4. An efficient procedure for identifying equivalences is described in Section 5.

Table 1. Equivalences based on an Exhaustive Test Set

circuit	pi	sv	pairs	equiv	%equiv
lion	2	2	44	0	0.00
dk15	3	2	185	2	1.08
train4	2	2	47	1	2.13
tav	4	2	58	2	3.45
lion9	2	3	76	3	3.95
modulo12	1	4	98	4	4.08
train11	2	4	127	6	4.72
bbtas	2	3	83	4	4.82
mc	3	2	103	9	8.74
bbara	4	4	160	15	9.38
beecount	3	3	126	12	9.52
ex3	2	4	228	25	10.96
dk17	2	3	188	22	11.70
s8	3	3	76	9	11.84
shiftreg	1	3	41	5	12.20
dk14	3	3	267	34	12.73
ex2	2	5	476	70	14.71
ex6	5	3	343	52	15.16
donfile	2	5	420	68	16.19
ex7	2	4	218	36	16.51
ex5	2	3	217	37	17.05
opus	5	4	279	49	17.56
dk512	1	4	185	35	18.92
bbsse	7	4	319	62	19.44
sse	7	4	319	62	19.44
s1a	8	5	881	177	20.09
dk16	2	5	856	182	21.26
dk27	1	3	94	20	21.28
firstex	2	4	90	21	23.33
keyb	7	5	631	149	23.61
ex4	5	4	271	68	25.09
mark1	4	4	277	77	27.80
cse	7	4	537	154	28.68
dvram	8	6	754	236	31.30
fetch	9	5	567	179	31.57
rie	9	5	968	335	34.61
log	9	5	500	188	37.60
nucpwr	13	5	745	295	39.60

4 EXISTENCE OF EQUIVALENCES

This section describes a complete experiment that is meant to demonstrate the existence of equivalences in benchmark circuits for which exhaustive test sets can be derived. Completeness here implies that all the equivalences are found. A similar experiment finds potentially-equivalent faults

Table 2. Potential-Equivalences based on a Fault Detection Test Set

circuit	pi	sv	pairs	p.equiv	%p.equiv
s208	11	8	324	87	26.85
s298	3	14	450	136	30.22
s344	9	15	608	134	22.04
s382	3	21	605	105	17.36
s420	19	16	669	169	25.26
s526	3	21	838	207	24.70
s641	35	19	939	102	10.86
s820	18	5	926	299	32.29
s953	16	29	1261	480	38.07
s1196	14	18	553	23	4.16
s1423	17	74	2366	253	10.69
s5378	35	179	8429	1211	14.37
s9234	19	228	14868	2247	15.11
s13207	31	669	23465	4039	17.21
s15850	14	597	28157	3159	11.22
s35932	35	1728	52814	1474	2.79
s38417	28	1636	74251	4314	5.81
s38584	12	1452	60908	8356	13.72
b03	5	30	682	94	13.78
b04	12	66	1843	91	4.94
b05	2	34	2774	291	10.49
b07	2	51	1778	242	13.61
b08	10	21	714	144	20.17
b09	2	28	614	45	7.33
b10	12	17	728	92	12.64
b11	8	30	1708	298	17.45
b14	33	247	15126	1894	12.52
b15	36	447	33147	7177	21.65
b20	33	494	34372	4191	12.19
aes_core	258	530	89744	17061	19.01
des_area	239	128	7678	634	8.26
i2c	17	128	3457	467	13.51
pci_spoci_ctrl	23	60	2520	776	30.79
sasc	15	117	2861	334	11.67
simple_spi	15	131	3384	431	12.74
spi	45	229	10830	1595	14.73
steppermotordrive	3	25	708	85	12.01
systemcaes	258	670	37914	3899	10.28
systemcdes	130	190	11626	1107	9.52
usb_phy	14	98	2234	263	11.77
tv80	13	359	27159	5673	20.89
wb_dma	215	523	13649	1973	14.46
average				1801	15.69

in larger circuits for which exhaustive test sets are prohibitive. Potentially-equivalent faults are used only for comparison with the accurate results in Section 5.

4.1 Equivalent Faults

For the first experiment in this section, T_{EXH} is the exhaustive set of all the LOS tests. With S states and V primary input vectors, the number of LOS tests in T_{EXH} is $2 \cdot S \cdot V$.

The experiment considers every pair of corresponding stuck-at and transition faults g/a and $g/a \rightarrow a'$. The faults are simulated under all the tests in T_{EXH}. Fault simulation yields values of two fault detection flags for every test $t_i \in T_{EXH}$, The flags are denoted by $d_i(g/a)$ and $d_i(g/a \rightarrow a')$. We have that $d_i(g/a) = 1$ if t_i detects g/a, and $d_i(g/a) = 0$ otherwise. Similarly, $d_i(g/a \rightarrow a') = 1$ if t_i detects $g/a \rightarrow a'$, and $d_i(g/a \rightarrow a') = 0$ otherwise. The procedure stops as soon as it finds a test t_i for which $d_i(g/a) \neq d_i(g/a \rightarrow a')$. If this does not occur for any test, the faults are detected by the same tests. Therefore, they are equivalent.

The results of this experiment for finite-state machine benchmarks are shown in Table 1. After the circuit name, column *pi* shows the number of primary inputs. Column *sv* shows the number of state variables. Column *pairs* shows the number of fault pairs where each pair contains a single stuck-at fault and the corresponding transition fault. Fault collapsing is not used in order to ensure that all the pairs of corresponding faults can be considered. Only pairs where both faults are detectable are counted.

Column *equiv* shows the number of equivalent pairs found. Column *%equiv* shows the percentage of equivalent pairs out of the total number of fault pairs.

From Table 1 it can be seen that the percentage of equivalent pairs of corresponding stuck-at and transition faults varies significantly with the circuit. In many circuits the percentage of equivalent fault pairs is high.

4.2 Potentially-Equivalent Faults

Suppose that T_{EXH} is replaced with a non-exhaustive fault detection LOS test set T_{DET} for stuck-at and transition faults. Such a test set is available from [16]. The test set T_{DET} from [16] consists of a compact single-cycle test set for single stuck-at faults, which is transformed into a LOS test set, and a compact LOS test set for transition faults. Using T_{DET}, the experiment from Section 4.1 is repeated. In this case, the following results are obtained.

If $d_i(g/a) \neq d_i(g/a \rightarrow a')$ is obtained for any test $t_i \in T_{DET}$, the faults g/a and $g/a \rightarrow a'$ are not equivalent.

If $d_i(g/a) = d_i(g/a \rightarrow a')$ is obtained for every test $t_i \in T_{DET}$, it is possible that the faults g/a and $g/a \rightarrow a'$ are equivalent if no test t_j exists outside of T_{DET} for which $d_j(g/a) \neq d_j(g/a \rightarrow a')$. Since it is not known whether such a test exists, the faults are referred to as potentially-equivalent. The number of potentially-equivalent fault pairs is an upper bound on the number of equivalent fault pairs.

This experiment is applicable to larger benchmark circuits for which an exhaustive test set is prohibitive. It is important since it yields an upper bound on the number of equivalent fault pairs that may be found by accurate analysis. The results for larger benchmark circuits are shown in Table 2.

Table 2 shows significant percentages of potentially-equivalent fault pairs, similar to the percentages of equivalent fault pairs in Table 1.

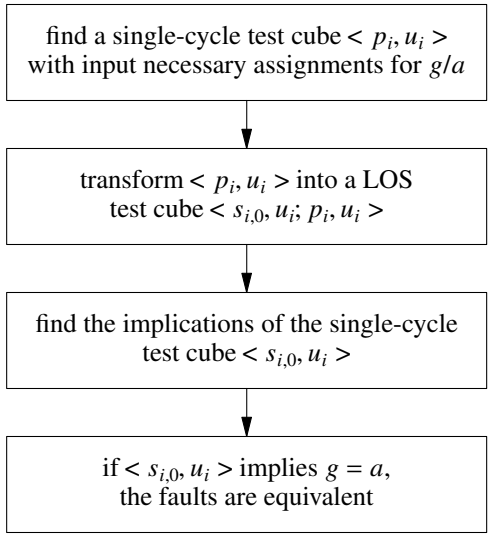

Fig. 3. Overall description of the procedure

5 EFFICIENT IDENTIFICATION PROCEDURE

This section describes an efficient procedure for identifying equivalences between stuck-at faults and the corresponding transition faults under LOS tests with equal primary input vectors. The overall flow of the procedure is described first followed by its details and experimental results.

5.1 Overall Description

The procedure considers a pair of corresponding stuck-at and transition faults, g/a and $g/a \rightarrow a'$. The steps of the procedure are illustrated by Figure 3.

The procedure first finds input necessary assignments for g/a. These are values that must be assigned to present-state variables and primary inputs under a single-cycle test to ensure that g/a is detected. The input necessary assignments are included in a single-cycle test cube $\langle p_i, u_i \rangle$.

The single-cycle test cube is transformed into a LOS test cube $\langle s_{i,0}, u_i; p_i, u_i \rangle$, where p_i is obtained by a single shift of $s_{i,0}$. This LOS test cube represents the input necessary assignments for the detection of g/a by a LOS test with equal primary input vectors.

The first cycle of the LOS test cube yields a single-cycle test cube $\langle s_{i,0}, u_i \rangle$. The implications of the specified input values in this test cube are computed. Suppose that the implications include the assignment $g = a$. In this case, a LOS test for g/a is guaranteed to assign $g = a$ in the first clock cycle. This is the only additional requirement for the detection of $g/a \rightarrow a'$ on top of the detection of g/a. Therefore, a LOS test for the stuck-at fault g/a is guaranteed to detect the transition fault $g/a \rightarrow a'$. Since the reverse is always true, the two faults are equivalent.

For illustration, the next example considers benchmark circuit $b01$. The circuit has five state variables and three primary inputs. The faults for which equivalence is checked are the stuck-at fault $g_{28}/0$ and the corresponding transition fault $g_{28}/0 \rightarrow 1$. The input necessary assignments for $g_{28}/0$ are included in the single-cycle test cube $\langle x010x, xx1 \rangle$. The single-cycle test cube is transformed into the LOS test cube $\langle 010xx, xx1; x010x, xx1 \rangle$. This is shown in Figure 4. Computing the implications of the single-cycle test cube $\langle 010xx, xx1 \rangle$, it turns out that $g_{28} = 0$ is one of

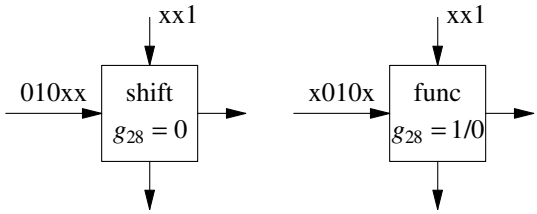

Fig. 4. Example of transformed LOS test cube

the implications. Therefore, a LOS test for $g_{28}/0$, which must be covered by the LOS test cube $\langle 010xx, xx1; x010x, xx1\rangle$, also detects the corresponding transition fault $g_{28}/0 \rightarrow 1$.

5.2 Input Necessary Assignments

To compute input necessary assignments for a stuck-at fault g/a, the procedure described in this section initializes a single-cycle test cube $\langle p_i, u_i\rangle$ to the all-x cube. For every input b of the combinational logic (present-state variable or primary input), the procedure checks whether $b = 0$ or $b = 1$ prevents the fault g/a from being detected when it is added to $\langle p_i, u_i\rangle$. In general, if $b = 0$ ($b = 1$) prevents the fault from being detected when it is added to $\langle p_i, u_i\rangle$, the assignment $b = 1$ ($b = 0$) is necessary, and it is added to $\langle p_i, u_i\rangle$. The details are discussed next.

For every possible input assignment $b = \beta$, where $\beta = 0$ or 1, the procedure needs to check whether the partially-specified single-cycle test cube $\langle p_i, u_i\rangle$ allows the fault g/a to be detected after the assignment $b = \beta$ is added to it. To perform the check, the procedure first performs the implications of p_i and u_i, including $b = \beta$. If the implications result in $g = a$, the fault g/a cannot be detected. Otherwise, the procedure performs the implications of $g = a$ in the faulty circuit. It then looks for a path through which fault effects can be propagated from g to an output of the combinational logic. This requires the circuit to be traced from the outputs toward g using lines with values other than 0/0 and 1/1. If no such path exists, the fault cannot be detected. Otherwise, the fault can potentially be detected.

For an input b of the combinational logic, the procedure performs the following computations.

It adds the assignment $b = 0$ to $\langle p_i, u_i\rangle$ temporarily, and checks whether g/a can be detected. It assigns $d_0 = 1$ if the fault can be detected, and $d_0 = 0$ otherwise.

Independently, the procedure adds the assignment $b = 1$ to $\langle p_i, u_i\rangle$ temporarily, and checks whether g/a can be detected. It assigns $d_1 = 1$ if the fault can be detected, and $d_1 = 0$ otherwise.

Based on d_0 and d_1, the procedure makes one of the following decisions.

(1) If $d_0 = 0$ and $d_1 = 0$, the fault g/a is undetectable. Such faults are not considered in this paper.

(2) If $d_0 = 1$ and $d_1 = 0$, the assignment $b = 0$ is necessary for the detection of g/a. The procedure adds the assignment $b = 0$ to $\langle p_i, u_i\rangle$ permanently.

(3) Similarly, if $d_0 = 0$ and $d_1 = 1$, the assignment $b = 1$ is necessary for the detection of g/a. The procedure adds the assignment $b = 1$ to $\langle p_i, u_i\rangle$ permanently.

(4) If $d_0 = 1$ and $d_1 = 1$, both assignments to b are possible, and no input necessary assignment is added to $\langle p_i, u_i\rangle$.

The procedure considers all the inputs of the combinational logic repeatedly as long as it finds additional input necessary assignments. This ensures that the results are independent of the order of the inputs. Specifically, the first input necessary assignment is found with the all-x cube. This assignment (and others that can be found with the all-x-cube) will be found independent of the order. Once these assignments are found, there are additional assignments that can be found. This

is also independent of the order. After a sufficient number of passes over the inputs, all the possible assignments will be found.

To speed up this process, the procedure uses the following observations.

(1) Not all the inputs of the combinational logic affect the detection of a fault. To find the inputs that do, the procedure finds the output cone of g by tracing the circuit from g to the outputs. It then traces the circuit from all the outputs that are driven by g to the inputs. This yields the input cone of g. Only inputs of the combinational logic that are in the input cone of g are considered during the computation of input necessary assignments for g/a.

(2) The implications of $\langle p_i, u_i \rangle$ are kept up-to-date. When a new temporary assignment needs to be checked, only new implications are computed, and then canceled. The implications are updated when a new assignment becomes permanent.

Further improvements in efficiency can be obtained by keeping a list of changes that occur when a new assignment is added temporarily, and using the list to cancel the assignment or make it permanent. This would avoid the need to copy the values of all the circuit lines every time an assignment is considered.

5.3 Computational Complexity

To analyze the worst-case computational complexity for a pair of faults g/a and $g/a \rightarrow a'$, suppose that the input cone of g/a has B inputs and L lines. To find input necessary assignments for g/a, the procedure considers $O(B)$ inputs in every one of $O(B)$ iterations. The worst-case for the number of iterations occurs if a single additional input necessary assignment is obtained in every iteration. Overall, the procedure evaluates $O(B^2)$ input necessary assignments.

To evaluate an input necessary assignment $b = \beta$, the procedure performs implications for $b = \beta$. In addition, it performs implications for $g = a$ in the faulty circuit. Finally, it traces the circuit to check whether g/a has a propagation path. In the worst case, this requires $O(L)$ operations.

The total is $O(B^2L)$ operations. In effect, a small number of iterations is typically required before no additional input necessary assignments can be found. In addition, fewer than L lines are considered for implications and propagation paths.

5.4 Experimental Results

The results of the efficient procedure for identifying equivalent pairs of corresponding stuck-at and transition faults are given in Table 3.

Table 3 is organized similar to Table 1. After the circuit name, column *pi* shows the number of primary inputs. Column *sv* shows the number of state variables. Column *pairs* shows the number of fault pairs where each pair contains a single stuck-at fault and the corresponding transition fault. Column *equiv* shows the number of equivalent pairs found. Column *%equiv* shows the percentage of equivalent pairs out of the total number of fault pairs.

Column *cone* shows the average number of inputs in the input cone for a pair of equivalent faults (the input cone for the pair is defined based on the stuck-at fault). Column *rtime* shows the average runtime for a pair of equivalent faults. The runtime is measured in seconds on a Linux machine with a 3GHz processor. Dashes are entered in the last two columns when the circuit does not have any pairs of equivalent faults.

The following points can be seen from Table 3. Not all the potentially-equivalent fault pairs from Table 2 are proved to be equivalent. Considering the averages in Tables 2 and 3, an average of 1801 potentially-equivalent fault pairs are found in Table 2. This is an upper bound on the number of truly equivalent fault pairs. An average of 588 equivalent fault pairs are found in Table 3. These

Table 3. Efficient Identification of Equivalences

circuit	pi	sv	pairs	equiv	%equiv	cone	rtime
s208	11	8	324	66	20.37	12.65	0.000
s298	3	14	450	116	25.78	7.68	0.000
s344	9	15	608	54	8.88	9.85	0.000
s382	3	21	605	76	12.56	9.07	0.000
s420	19	16	669	138	20.63	21.53	0.000
s526	3	21	838	185	22.08	10.21	0.000
s641	35	19	939	30	3.19	19.73	0.000
s820	18	5	926	287	30.99	13.45	0.000
s953	16	29	1261	427	33.86	15.76	0.000
s1196	14	18	553	2	0.36	26.50	0.005
s1423	17	74	2366	115	4.86	49.98	0.002
s5378	35	179	8429	977	11.59	34.84	0.006
s9234	19	228	14868	1341	9.02	63.43	0.028
s13207	31	669	23465	3018	12.86	114.32	0.153
s15850	14	597	28157	2230	7.92	151.32	0.385
s35932	35	1728	52814	0	0.00	-	-
s38417	28	1636	74251	1680	2.26	25.45	0.111
s38584	12	1452	60908	6291	10.33	35.82	0.176
b03	5	30	682	19	2.79	17.89	0.000
b04	12	66	1843	4	0.22	22.50	0.000
b05	2	34	2774	12	0.43	23.67	0.002
b07	2	51	1778	28	1.57	26.04	0.002
b08	10	21	714	46	6.44	16.37	0.001
b09	2	28	614	6	0.98	22.00	0.000
b10	12	17	728	20	2.75	13.45	0.000
b11	8	30	1708	16	0.94	14.00	0.001
b14	33	247	15126	12	0.08	150.67	0.244
b15	36	447	33147	16	0.05	367.88	2.118
b20	33	494	34372	13	0.04	258.31	1.001
aes_core	258	530	89744	300	0.33	35.51	0.153
des_area	239	128	7678	0	0.00	-	-
i2c	17	128	3457	195	5.64	21.29	0.003
pci_spoci_ctrl	23	60	2520	540	21.43	38.30	0.004
sasc	15	117	2861	27	0.94	6.85	0.001
simple_spi	15	131	3384	198	5.85	17.93	0.002
spi	45	229	10830	120	1.11	89.81	0.051
steppermotordrive	3	25	708	39	5.51	19.46	0.001
systemcaes	258	670	37914	2170	5.72	156.96	0.268
systemcdes	130	190	11626	11	0.09	109.82	0.070
tv80	13	359	27159	2985	10.99	155.65	0.205
usb_phy	14	98	2234	116	5.19	9.24	0.001
wb_dma	215	523	13649	760	5.57	38.39	0.028
average				588	7.67		

pairs are truly equivalent. Overall, the procedure finds significant numbers of equivalent pairs consisting of stuck-at and corresponding transition faults.

The average number of cone inputs that the procedure needs to consider is significantly lower than the number of inputs to the combinational logic. This limits its runtime.

5.5 Fault Collapsing

The procedure for identifying equivalences was described for pairs of corresponding stuck-at and transition faults assuming no fault collapsing for single stuck-at faults. However, it can also be applied if fault collapsing is used for stuck-at faults. Stuck-at fault collapsing can also be used for speeding up the procedure as discussed next.

For a transition fault $g/a \rightarrow a'$, the corresponding stuck-at fault is g/a. Suppose that g/a does not exist in the set of target faults because it was removed by fault collapsing. In this case, the set of target faults contains a stuck-at fault that is equivalent to g/a. Let the fault $\hat{g}/\hat{a}$ be the stuck-at fault in the set of target faults that is equivalent to g/a. The fault $\hat{g}/\hat{a}$ can be found during or after fault collapsing, and associated with $g/a \rightarrow a'$.

The procedure for finding equivalences is applied only to the stuck-at faults in the set of target faults. When a fault h/c is considered, the procedure finds a single-cycle test cube followed by a LOS test cube of input necessary assignments, $\langle s_{i,0}, u_i; p_i, u_i \rangle$. The procedure performs logic simulation of the single-cycle test cube $\langle s_{i,0}, u_i \rangle$. For every fault $g/a \rightarrow a'$ whose associated stuck-at fault is $\hat{g}/\hat{a} = h/c$, the procedure checks whether $g = a$ is an implication of $\langle s_{i,0}, u_i \rangle$. If this is the case, $g/a \rightarrow a'$ and h/c are equivalent.

6 TEST COMPACTION

This section discusses the effects of the equivalences between corresponding stuck-at and transition faults on a test compaction procedure. The test compaction procedure is the one from [16]. This is the only available test compaction procedure that generates LOS tests for stuck-at and transition faults. The procedure accepts a compact single-cycle test set for stuck-at faults, and a compact LOS test set for transition faults. It transforms single-cycle tests into LOS tests, combines the two test sets, and then achieves further test compaction by reordering the test set, modifying tests, and removing tests that become unnecessary. The procedure requires extensive fault simulation to compact a test set made up of two subsets that are already compact.

The test compaction procedure from [16] uses a collapsed set of single stuck-at faults, and the set of all the transition faults (only limited fault collapsing is possible for transition faults, and its effects are negligible). The baseline for comparison is the procedure when it is run with these sets of target faults. To check the effects of the equivalences suggested in this paper, the procedure from Section 5.5 is used for associating transition faults with stuck-at faults, and finding equivalent faults. A transition fault found to have an equivalent stuck-at fault is removed from the set of target faults.

The expected effect of removing equivalent transition faults from the set of target faults is to speed up the fault simulations that the test compaction procedure performs. An unexpected effect that occurs is that removing equivalent transition faults changes the flow of the test compaction procedure, causing it to produce smaller test sets. This can be explained as follows.

One of the subprocedures of the test compaction procedure from [16] performs fault simulation of the test set with fault dropping to compute the number of faults detected by each test. It then reorders the test set such that the tests appear by decreasing number of detected faults. In this order it performs fault simulation with fault dropping again. If tests at the end of the test set do not detect any faults, they are removed from the test set.

Table 4. Test Compaction

		with			without		
circuit	%equiv	tests	frac	ntime	tests	frac	ntime
s953	33.86	103	0.912	132.33	102	0.903	135.33
tv80	10.99	652	0.894	5584.05	642	0.881	4647.88
wb_dma	5.57	147	0.821	6016.42	146	0.816	5215.73
s820	30.99	112	0.903	45.33	114	0.919	34.83
s5378	11.59	183	0.750	1899.81	186	0.762	1202.98
s9234	9.02	252	0.768	343.65	260	0.793	400.42
s13207	12.86	298	0.730	6496.22	353	0.865	2234.89
s15850	7.92	227	0.703	1000.30	236	0.731	643.92
s38584	10.33	309	0.771	1279.08	320	0.798	1106.93
i2c	5.64	86	0.827	408.72	90	0.865	155.22
pci_spoci_ctrl	21.43	172	0.815	1025.06	180	0.853	472.33
systemcaes	5.72	153	0.968	1579.94	154	0.975	562.76
average		224	0.822		232	0.847	

Another subprocedure from [16] has a basic step where it considers a test t_i that detects more than one fault, and a test t_j that detects only one fault. It attempts to modify t_i so as to detect the fault detected by t_j. When this is successful, t_j is removed from the test set. This subprocedure is influenced by the order of the tests since this order determines how many faults are detected by each test (fault simulation in [16] is always carried out with fault dropping).

When equivalences are used for removing transition faults from the set of target faults, the numbers of detected faults become more accurate in capturing the numbers of truly different faults that are detected by every test. As a result, reordering becomes more effective. In addition, a test t_i that detects a stuck-at fault and an equivalent transition fault detects only one fault when equivalences are used. In this case, it is considered for removal.

The effects of equivalences on the flow of the test compaction procedure can be eliminated by keeping the equivalent faults in the set of target faults, but avoiding their simulation. When a stuck-at fault is detected, if an associated transition fault is known to be equivalent, the transition fault can be counted as detected without being simulated. In this case, numbers of detected faults will remain the same, and the flow of the procedure will not change. The runtime is expected to decrease because fewer faults are simulated. In this paper, the effect of equivalences on the flow of the procedure, and its ability to achieve test compaction, are investigated by removing equivalent faults from the set of target faults. It should be noted that the runtime may increase when the flow changes.

The results of the test compaction procedure from [16] with and without identifying equivalences are shown in Table 4 as follows. The circuits are ones where the numbers of tests in the compact transition and stuck-at test sets are large enough to allow additional test compaction to be achieved. In addition, the percentage of equivalences is at least 5% to ensure that the effects of identifying them is noticeable.

Table 4 is organized as follows. Column *%equiv* repeats the percentage of equivalences found by the efficient procedure. Column *with* (*without*) shows the results with (without) identification of equivalences. Subcolumn *tests* shows the number of tests in the final compact test set. Subcolumn *frac* shows the fraction of tests that remain in the final test set relative to the initial test set. Subcolumn *ntime* shows the normalized runtime, where the total runtime is divided by the runtime

for fault simulation with fault dropping of the initial test set. The runtime for finding equivalences is included in the total runtime when they are used.

For the circuits in the top part of Table 4, the use of equivalences results in a larger number of tests in the final test set. For the circuits in the bottom part of Table 4, the use of equivalences results in a smaller number of tests.

From Table 4 it can be seen that the use of equivalences allows the test compaction procedure to produce smaller final test sets for most of the circuits considered. In addition, the differences in numbers of tests are higher for the circuits in the bottom part of Table 4. The averages given in the last row of Table 4 further demonstrate the advantages of using equivalences. Overall, test compaction is based on heuristics that influence the flow of the procedure and its final results in complex ways, which may prevent a particular method (such as the use of equivalences) from producing universally better results. Nevertheless, on the average, the use of equivalences improves the ability of the procedure from [16] to achieve test compaction.

7 CONCLUDING REMARKS

This paper considered the scenario where LOS tests with equal primary input vectors are used for both stuck-at and transition faults. This scenario may be used for enhancing the ability to achieve test compaction for the two fault models together. A known observation is that a LOS test that detects a transition fault also detects a corresponding stuck-at fault. The paper showed that, under certain conditions, a LOS test that detects a stuck-at fault also detects a corresponding transition fault. When this occurs, the two faults are equivalent under LOS tests. Equivalence can be used for reducing the set of target faults for test generation and test compaction. The paper developed this notion of equivalence, provided an efficient procedure for identifying equivalences, and demonstrated the existence of equivalences in benchmark circuits using two types of experiments. It also studied the effects of equivalences on a test compaction procedure.

REFERENCES

[1] L. N. Reddy, I. Pomeranz and S. M. Reddy, "COMPACTEST-II: A Method to Generate Compact Two-pattern Test Sets for Combinational Logic Circuits", in Proc. Intl. Conf. on Computer-Aided Design, 1992, pp. 568-574.
[2] R. Desineni, K. N. Dwarkanath and R. D. Blanton, "Universal Test Generation using Fault Tuples", in Proc. Intl. Test Conf., 2000, pp. 812-819.
[3] S. M. Reddy, G. Chen, J. Rajski, I. Pomeranz, P. Engelke and B. Becker, "A Unified Fault Model and Test Generation Procedure for Interconnect Opens and Bridges", in Proc. Europ. Test Symp., 2005, pp. 22-27.
[4] S. Goel and R. A. Parekhji, "Choosing the Right Mix of At-Speed Structural Test Patterns: Comparisons in Pattern Volume Reduction and Fault Detection Efficiency", in Proc. Asian Test Symp., 2005, pp. 330-336.
[5] D. Kim, M. E. Amyeen, S. Venkataraman, I. Pomeranz, S. Basumallick and B. Landau, "Testing for Systematic Defects Based on DFM Guidelines", in Proc. Intl. Test Conf., Oct. 2007, pp. 1-10.
[6] 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.
[7] S. Alampally, R. T. Venkatesh, P. Shanmugasundaram, R. A. Parekhji and V. D. Agrawal, "An Efficient Test Data Reduction Technique through Dynamic Pattern Mixing Across Multiple Fault Models", in Proc. VLSI Test Symp., 2011, pp. 285-290.
[8] F. Hapke and J. Schloeffel, "Introduction to the Defect-oriented Cell-aware Test Methodology for Significant Reduction of DPPM Rates", in Proc. European Test Symp., 2012, pp. 1-6.
[9] F. Yang, S. Chakravarty, A. Gunda, N. Wu and J. Ning, "Silicon Evaluation of Cell-Aware ATPG Tests and Small Delay Tests", in Proc. Asian Test Symp., 2014, pp. 101-106.
[10] A. D. Singh, "Cell Aware and Stuck-open Tests", in Proc. European Test Symp., 2016, pp. 1-6.
[11] C.-H. Wu and K.-J. Lee, "Transformation of Multiple Fault Models to a Unified Model for ATPG Efficiency Enhancement", in Proc. Intl. Test Conf., 2016, pp. 1-10.
[12] J. Savir and S. Patil, "Scan-Based Transition Test", IEEE Trans. on Computer-Aided Design, Aug. 1993, pp. 1232-1241.
[13] J. Savir and S. Patil, "Broad-Side Delay Test", IEEE Trans. on Computer-Aided Design, Aug. 1994, pp. 1057-1064.

[14] 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.
[15] 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.
[16] I. Pomeranz, "Skewed-Load Tests for Transition and Stuck-at Faults", IEEE Trans. on Computer-Aided Design, Vol 38, No. 10, Oct. 2019, pp. 1969-1973.
[17] A. Sreedhar, A. Sanyal and S. Kundu, "On Modeling and Testing of Lithography Related Open Faults in Nano-CMOS Circuits", in Proc. Design, Autom. and Test in Europe Conf., 2008, pp. 616-621.
[18] M. O. Simsir, A. Bhoj and N. K. Jha, "Fault Modeling for FinFET Circuits", in Proc. Intl. Symp. on Nanoscale Architectures, 2010, pp. 41-46.
[19] J. Zha, X. Cui and C. L. Lee, "Modeling and Testing of Interference Faults in the Nano NAND Flash Memory", in Proc. Design, Automation & Test in Europe Conf., 2012, pp. 527-531.
[20] D. Xiang, Z. Chen and L.-T. Wang, "Scan Flip-Flop Grouping to Compress Test Data and Compact Test Responses for Launch-on-Capture Delay Testing", ACM Trans. on Design Automation, Vol. 17, No. 2, April 2012, Article No. 18.
[21] S.-Y. Huang, Y.-H. Lin, K.-H. Tsai, W.-T. Cheng, S. Sunter, Y.-F Chou and D.-M. Kwai, "Small Delay Testing for TSVs in 3-D ICs", in Proc. Design Automation Conf., 2012, pp. 1031-1036.
[22] A. Mondal, P. P. Chakrabarti and P. Dasgupta, "Symbolic-Event-Propagation-Based Minimal Test Set Generation for Robust Path Delay Faults", ACM Trans. on Design Automation, Vol. 17 No. 4, Oct. 2012, Article No. 47.
[23] M. Sauer, A. Czutro, I. Polian and B. Becker, "Small-delay-fault ATPG with Waveform Accuracy", in Proc. Intl. Conf. on Computer-Aided Design, 2012, pp. 30-36.
[24] W. Zhao, J. Ma, M. Tehranipoor and S. Chakravarty, "Power-safe Application of TDF Patterns to Flip-chip Designs during Wafer Test", ACM Trans. on Design Automation, Vol. 18, No. 3, July 2013, Article No. 43.
[25] S. Di Carlo, G. Gambardella, P. Prinetto, D. Rolfo and P. Trotta, "SATTA: A Self-Adaptive Temperature-Based TDF Awareness Methodology for Dynamically Reconfigurable FPGAs", ACM Trans. on Reconfigurable Technology and Systems, Vol. 8, No. 1, Feb. 2015, Article No. 1.
[26] Y. Zhang, Z. Peng, J. Jiang, H. Li and M. Fujita, "Temperature-aware Software-based Self-testing for Delay Faults", in Proc. Design, Automation & Test in Europe Conf., 2015, pp. 423-428.
[27] H. G. Mohammadi, P.-E. Gaillardon and G. De Micheli, "Fault Modeling in Controllable Polarity Silicon Nanowire Circuits", in Proc. Design, Automation & Test in Europe Conf., 2015, pp. 453-458.
[28] A. S. Trinadh, S. Potluri, S. B. Ch., V. Kamakoti and S. G. Singh, "Optimal Don't Care Filling for Minimizing Peak Toggles During At-Speed Stuck-At Testing", ACM Trans. on Design Automation, Vol. 23, No. 1, Oct. 2017, Article No. 5.
[29] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1995.