Test and Non-Test Cubes for Diagnostic Test Generation Based on Merging of Test Cubes

Irith Pomeranz
School of Electrical & Computer Eng.
Purdue University
W. Lafayette, IN 47907, U.S.A.

Abstract - **Test generation by merging of test cubes supports test compaction and test data compression. This paper describes a new approach to the use of test cube merging for the generation of compact diagnostic test sets. For this the paper uses the new concept of non-test cubes. While a test cube for a fault f_{i0} detects the fault, a non-test cube for a fault f_{i1} prevents the fault from being detected. Merging a test cube for a fault f_{i0} and a non-test cube for a fault f_{i1} produces a diagnostic test cube that distinguishes the two faults. The paper describes a procedure for diagnostic test generation based on merging of test and non-test cubes. Experimental results demonstrate that compact diagnostic test sets are obtained.**

I. Introduction

An approach to test generation that supports test compaction as well as test data compression consists of generating partially-specified tests (test cubes) and then merging test cubes that are compatible [1]. Two test cubes t_0 and t_1 are said to be compatible if they assign the same values to all the inputs where they are both specified. When two compatible test cubes t_0 and t_1 are merged, the resulting test cube has a specified value for every input where t_0 or t_1 is specified. The number of test cubes is reduced since t_0 and t_1 are replaced by a single test cube. The merging process can be performed under the constraints of a test data compression method to ensure that the resulting test cubes can be compressed and tests can be applied through on-chip decompression logic.

This paper describes a new approach to the use of test cube merging for the generation of compact diagnostic test sets. A diagnostic test generation procedure based on test cube merging fits within a test generation flow where test cubes are generated and then merged under the constraints of a test data compression method.

Diagnostic test generation procedures were described in [2]-[9]. The goal of diagnostic test generation is to distinguish pairs of faults. Considering a fault pair (f_{i0},f_{i1}), a test that causes the circuit to produce different output vectors, depending on whether f_{i0} or f_{i1} is present in it, is said to distinguish the faults. When faults are distinguished, at most one of them needs to be included in the set of candidate faults that a fault diagnosis procedure produces. Since the computational complexity of diagnostic test generation is higher than that of test generation for fault detection, and since diagnostic test sets are significantly larger than fault detection test sets, diagnostic tests are generated only for fault pairs that are not distinguished by a fault detection test set.

In order to develop a diagnostic test generation procedure that is based on merging of test cubes, this paper uses the new concept of non-test cubes [10]. A test cube d_{i0} for a fault f_{i0} detects the fault. A non-test cube u_{i1} for a fault f_{i1} prevents the fault from being detected. Merging the test cube d_{i0} for f_{i0} and the non-test cube u_{i1} for f_{i1} produces a test cube t that detects one fault but not the other. As a result, the faults produce different output vectors, and t distinguishes the faults.

Generalizing this concept to consider the specific output on which the faults are detected or not detected provides a complete characterization of diagnostic tests. Specifically, let d_{i0}^z be a test cube that detects a fault f_{i0} on an output z. Let u_{i1}^z be a non-test cube that prevents f_{i1} from being detected on the same output z. Merging the test cube d_{i0}^z for f_{i0} and the non-test cube u_{i1}^z for f_{i1} produces a test cube t^z that detects one fault but not the other on z. As a result, it distinguished the two faults. For simplicity, this paper does not consider specific outputs when defining test and non-test cubes. Nevertheless, during diagnostic fault simulation and test generation, a fault pair (f_{i0},f_{i1}) is considered to be distinguished by a test cube even if both faults are detected, as long as they are detected on different outputs, thus producing different output vectors.

Test and non-test cubes for diagnostic test generation can be extracted from a fault detection test set. The paper describes procedures for this purpose. The paper also notes that an arbitrary test cube d_{i0} for a fault f_{i0} and an arbitrary non-test cube u_{i1} for a fault f_{i1} may conflict. In this case, they cannot be merged using a procedure such as the one described in [1]. The paper overcomes this issue by defining an initial test cube t based on d_{i0} and u_{i1}, and then modifying t in order to obtain a diagnostic test cube that distinguishes (f_{i0},f_{i1}). The procedure thus uses values from d_{i0} and u_{i1} as a basis for a diagnostic test cube even if they conflict.

Finally, the paper describes a diagnostic test generation procedure based on the use of test and non-test cubes, and presents experimental results to demonstrate its effectiveness in producing compact diagnostic tests.

Throughout this paper, single stuck-at faults are used as target faults. The same concepts are applicable to other fault models. Diagnostic test generation procedures

978-3-9815370-2-4/DATE14/©2014 EDAA

typically target single stuck-at faults since these faults are also used by fault diagnosis procedures to define sets of candidate faults. In addition, tests that distinguish single stuck-at faults are effective in distinguishing other types of faults as well.

The paper is organized as follows. Section II reviews the concept of non-test cubes. Section III describes the generation of diagnostic tests based on test and non-test cubes. Section IV presents experimental results.

II. Non-Test Cubes

This section reviews the definition and computation of non-test cubes. Non-test cubes were defined for the purpose of test generation targeting hard-to-detect faults [10]. They are used here for diagnostic test generation.

A non-test cube u_i for a fault f_i is a partially-specified input vector such that its specified input values are sufficient for preventing f_i from being detected. As a result, none of the fully-specified input vectors, which are covered by u_i, detect f_i.

Diagnostic test generation, which is considered in this paper, starts from a fault detection test set. For a given detectable fault f_i, a fault detection test set contains at least one test that detects f_i. In addition, it typically contains tests that do not detect f_i. One of these tests can be used for deriving a non-test cube for f_i. A procedure for unspecifying a test t to obtain a non-test cube u_i for a fault f_i is described next.

Let $t = t(0)t(1) \cdots t(n-1)$ be a test for a circuit with n primary inputs, where $t(j)$ is the value of input j, for $0 \le j < n$. Suppose that t does not detect f_i. To obtain a non-test cube u_i for f_i, the procedure initially assigns $u_i = t$. It then considers the input values of u_i one at a time in a random order. When $u_i(j)$ is considered, the procedure carries out the following steps.

The procedure assigns $u_i^j = u_i$ and then unspecifies the value of input j under u_i^j by assigning $u_i^j(j) =$ x. Using implications, the procedure computes a fault-free/faulty value for every line. The faulty value is the one obtained in the presence of f_i. The procedure then searches for a propagation path from the site of f_i to an output. A line g on a propagation path cannot be assigned a 0/0 or 1/1 value. The other seven values, 0/1, 0/x, 1/0, 1/x, x/0, x/1 and x/x, are acceptable. The search for a propagation path can be done in time that is linear in the size of the circuit. If a propagation path exists, it is possible that one of the tests that covers u_i^j will detect f_i. In this case, the unspecified value of $u_i^j(j)$ is not accepted. If no propagation path exists, u_i^j does not allow f_i to be detected, and the procedure accepts the unspecified value of $u_i^j(j)$ by assigning $u_i = u_i^j$.

III. Diagnostic Test Generation

This section discusses the generation of diagnostic tests by test cube merging using test and non-test cubes. The basic step of the procedure is the generation of a partially-specified diagnostic test for a given fault pair. This step is described in Subsection III.A. Subsection III.B describes the overall test generation process that is used in the experiments reported in this paper.

Similar to other diagnostic test generation procedures, the procedure described in this section starts from a fault detection test set. The fault detection test set is denoted by T_0, and it detects all the detectable target faults. The procedure uses this test set as a source for test and non-test cubes.

The procedure is described without considering the constraints of a particular test data compression method. Such constraints can be accommodated when test cubes are merged and modified.

A. Generation of a Diagnostic Test for a Fault Pair

Let (f_{i0}, f_{i1}) be a fault pair such that f_{i0} and f_{i1} are detected but not distinguished by T_0.

If T_0 was generated by merging of test cubes, a test cube for f_{i0} was obtained during the generation of T_0. If this test cube is available it can be used for the generation of a diagnostic test for (f_{i0}, f_{i1}). If a test cube for f_{i0} is not available, it can be extracted from a test in T_0 that detects f_{i0}. The extraction process used in this paper selects a test for f_{i0} by simulating tests from T_0 in a random order until it finds one that detects f_{i0}. The process then unspecifies the input values of this test one at a time. An unspecified value is accepted if f_{i0} continues to be detected. Otherwise, the specified value is restored. The resulting test cube is denoted by d_{i0}.

A non-test cube for f_{i1} is extracted from a test in T_0 that does not detect f_{i1} as described in Section II. The test is selected by simulating tests from T_0 in a random order until one is found, which does not detect f_{i1}. The non-test cube for f_{i1} is denoted by u_{i1}.

If d_{i0} and u_{i1} do not conflict in any input value, they can be merged to obtain a diagnostic test cube for (f_{i0}, f_{i1}).

If d_{i0} and u_{i1} conflict, the procedure defines an initial test cube t that includes all the specified values of u_{i1}, and all the specified values of d_{i0} that do not conflict with the corresponding values of u_{i1}. The rationale for this initialization of t is as follows. Non-test cubes typically contain significantly fewer specified values than test cubes, and there are significantly fewer options for modifying them and still ensure that a fault is not detected. Therefore, u_{i1} is given a higher priority in determining t.

If t does not distinguish (f_{i0}, f_{i1}), it is modified by a test elimination process [5] that complements its specified values one at a time. This procedure does not affect the unspecified values of t in order not to lose the possibility of compressing t. When the (specified) value of input j is considered, the procedure assigns $t^j = t$ and then complements $t^j(j)$. Fault simulation is carried out to determine whether or not to accept the complemented value. The conditions for accepting a complemented value are the following.

(1) If t^j distinguishes (f_{i0}, f_{i1}), the procedure assigns $t = t^j$ and terminates with t as a test cube that distinguishes (f_{i0}, f_{i1}).

(2) If t^j does not detect f_{i1}, the procedure accepts the

complemented value of input j by assigning $t = t^j$. Thus, the procedure requires that f_{i1} would continue to not be detected. This is based on the same observation that caused the procedure to give a higher priority to u_{i1} when initializing t. Since non-test cubes have small numbers of specified values, it is important to keep f_{i1} from being detected as the test cube is being modified.

The procedure performs a constant number of iterations over all the specified input values of t. Compared with the test elimination procedure from [5], the procedure considers significantly fewer input values since most of the inputs are typically unspecified under d_{i0} and u_{i1}. In addition, the procedure uses the requirement of not detecting f_{i1} to further reduce the computational effort.

B. Overall Diagnostic Test Generation Process

The diagnostic test generation procedure implemented in this paper extends the fault detection test set T_0 into a diagnostic test set by performing several iterations. For $I > 0$, iteration I targets a subset of fault pairs that is denoted by P_I, and it produces a diagnostic test set that is denoted by T_I. For a variable denoted by s_I, P_I consists of the fault pairs from the equivalence classes of size s_I that are obtained for T_{I-1}. The use of equivalence classes is motivated by the following considerations.

Equivalence classes are obtained by diagnostic fault simulation. Two faults are included in the same equivalence class if and only if they are not distinguished. Faults in the same equivalence class have to be included together in the set of candidate faults that a fault diagnosis procedure produces. If T_I distinguishes pairs of faults from the equivalence classes of size s_I that are obtained for T_{I-1}, a fault diagnosis procedure can potentially obtain smaller sets of candidate faults by using T_I.

Let C_{I-1} be the set of equivalence classes obtained for T_{I-1}. The value of s_I is determined so as to target the largest possible equivalence classes, as follows.

For $I = 1$, s_1 is the size of the largest equivalence classes obtained for T_0. For $I > 1$, s_I is initially assigned the value $s_{I-1}-1$. If C_{I-1} does not contain any equivalence class of size s_I, the procedure reduces s_I down to the size of the next largest equivalence class in C_{I-1}. The procedure terminates when s_I is smaller than the size of the smallest equivalence class in C_{I-1}. In all the cases, $s_I = 2$ is obtained in the last iteration.

After s_I is determined, P_I is defined as follows. For every $C_{I-1,k} \in C_{I-1}$, if $|C_{I-1,k}| = s_I$, P_I contains every pair of faults from $C_{I-1,k}$. For completeness, if f_{i0} and f_{i1} are included in $C_{I-1,k}$, both (f_{i0},f_{i1}) and (f_{i1},f_{i0}) are included in P_I. This is needed since the diagnostic test generation procedure treats the two faults of a pair differently as discussed earlier.

In iteration I the procedure attempts to generate a diagnostic test cube for every fault pair $(f_{i0},f_{i1}) \in P_I$. If a test cube t is generated, it is added to a set denoted by A_I. All the fault pairs in P_I that are distinguished by t are marked, and they are not considered further. The procedure considers all the unmarked fault pairs in P_I repeatedly until no new diagnostic test cube is generated. Repetition is useful since the procedure may use different test and non-test cubes at different times, and the test elimination process may also produce different results.

Next, the procedure reduces the number of test cubes in A_I by merging every pair of test cubes that do not conflict in any input value. It then adds the tests to T_I. Diagnostic fault simulation of the new tests added to T_I is used for updating the set of equivalence classes for T_I.

IV. Experimental Results

The procedure from Section III was applied to single stuck-at faults in ISCAS-89, ITC-99 and IWLS-05 benchmark circuits. The test set T_0 is a compacted fault detection test set for single stuck-at faults.

The results are shown in Table I as follows. There is a row for the fault detection test set T_0, and for the first and last values of s_I such that the diagnostic test generation procedure produced new diagnostic tests.

Column *sI* shows the value of s_I. Column *tests* shows the number of tests in T_I. Column *indist* shows the number of fault pairs that are not distinguished by T_I. Column 3 *larg cls* shows the sizes of the three largest equivalence classes obtained with respect to T_I. For $I > 0$, column *n.time* shows the normalized run time for generating T_I. The run time is normalized through division by the run time for diagnostic *fault simulation* of T_0. This provides an indication of the computational effort. Column *spec* shows the average percentage of specified values in test cubes and non-test cubes that are generated for faults in P_I.

The following points can be seen from Table I. The diagnostic test generation procedure is able to reduce significantly the sizes of the largest equivalence classes. After the largest equivalence classes cannot be reduced any further, the procedure also reduces the sizes of other equivalence classes. The overall effect can be seen from the reduction in the number of indistinguished fault pairs. With smaller numbers of indistinguished fault pairs, smaller candidate fault sets can be obtained.

Relatively small numbers of tests are sufficient for distinguishing large numbers of fault pairs. This is due to the use of test cube merging to reduce the number of tests that are added in every iteration.

The percentage of specified input values for non-test cubes is significantly lower than for test cubes. This contributes to the feasibility of performing diagnostic test generation by merging of test and non-test cubes.

Table II compares the number of tests and the number of indistinguished fault pairs obtained by the procedure described in this paper (column *cube merg*) with those obtained by the procedure from [7] (column [7]). The procedure from [7] considers fully-specified tests. It produces small diagnostic test sets compared with other procedures, and it distinguishes all or most of the fault pairs that are known to be distinguishable in benchmark circuits. Table II also shows the number of tests and the number of indistinguished fault pairs for the test set obtained by adding tests from the test set of [7] to the test set produced by the procedure described in this paper

Table I. Experimental Results

				3 larg				spec	
circuit	sI	tests	indist	cls			n.time	det	non
s5378	0	100	596	5	4	3			
s5378	5	102	587	4	3	3	2.45	4.72	0.30
s5378	2	127	534	4	3	3	135.27	4.46	0.90
s9234	0	111	1951	7	7	6			
s9234	7	114	1894	6	6	6	1.60	1.00	0.26
s9234	2	236	1266	5	4	4	526.40	5.52	0.87
s13207	0	235	2562	15	10	10			
s13207	15	239	2351	10	9	6	2.89	0.80	0.04
s13207	2	274	2062	9	5	5	498.95	1.40	0.24
s15850	0	97	3559	28	21	14			
s15850	14	98	3453	28	18	12	38.80	2.96	0.61
s15850	2	150	2943	28	15	10	624.16	1.81	0.41
s38417	0	87	5862	17	12	12			
s38417	17	95	5183	12	12	11	3.10	0.51	0.04
s38417	2	180	3459	11	11	11	1974.20	1.13	0.19
s38584	0	142	3317	5	5	5			
s38584	5	144	3292	5	5	4	3.91	0.72	0.11
s38584	2	187	2745	5	5	4	1337.26	0.51	0.11
b14	0	329	1523	27	17	17			
b14	27	339	1188	17	17	14	3.33	2.69	0.33
b14	2	456	420	14	6	5	98.76	11.81	0.83
b15	0	383	610	4	4	4			
b15	3	386	604	4	4	4	6.96	4.63	1.03
b15	2	413	548	4	4	4	122.70	3.92	0.76
b20	0	307	3441	47	27	27			
b20	47	334	2489	27	27	18	76.36	9.40	0.45
b20	2	444	874	13	12	7	279.90	7.29	0.50
aes_core	0	658	2098	2	2	2			
aes_core	2	661	2072	2	2	2	120.04	2.34	0.53
des_area	0	121	218	3	2	2			
des_area	3	123	212	2	2	2	1.13	4.27	1.23
des_area	2	136	189	2	2	2	54.71	4.66	1.46
pci_spoci_ctrl	0	148	76	4	3	3			
pci_spoci_ctrl	4	150	70	3	3	3	2.00	25.00	1.10
pci_spoci_ctrl	2	162	46	3	3	3	16.00	24.31	2.18
spi	0	414	312	5	4	4			
spi	5	415	304	4	4	3	1.33	3.32	0.36
spi	2	524	116	3	3	2	84.81	6.01	1.31
systemcaes	0	145	1519	4	4	3			
systemcaes	4	147	1510	4	3	3	1.61	0.99	0.19
systemcaes	2	169	1451	4	3	2	302.42	1.52	0.38
tv80	0	551	662	5	4	4			
tv80	4	553	654	5	4	4	2.55	4.75	0.72
tv80	2	597	567	5	4	4	96.41	4.98	0.58

(column *cube merg* + [7]). Only tests that distinguish additional fault pairs are included in the combined test set.

Table II shows that the use of partially-specified diagnostic tests, with limited numbers of specified values, sometimes prevents the procedure from distinguishing all the fault pairs that can be distinguished with fully-specified tests. However, the use of test cubes also allows the procedure described in this paper to produce smaller diagnostic test sets. For example, in the case of $s5378$, even if nine diagnostic tests are needed to distinguish nine additional fault pairs and reach the number from [7], the number of tests would be smaller, 136, for the procedure described in this paper. Considering the combined test set, 10 tests from [7] distinguish 10 additional fault pairs. Overall, for all the circuits considered, the combined test set is smaller than the one from [7], and it distinguishes the same or a larger number of fault pairs.

V. Concluding Remarks

This paper used the new concept of non-test cubes to support a diagnostic test generation procedure that is based on merging of test cubes. Such a procedure fits with a test generation flow that uses test cubes to satisfy the constraints of a test data compression method. A non-test cube for a fault f_{i1} prevents the fault from being detected. Merging a test cube d_{i0} for a fault f_{i0} and a non-test cube u_{i1} for a fault f_{i1} produces a test cube that distinguishes the two faults. The paper described a procedure for diagnostic test generation based on merging of test and non-test cubes. The merging process addressed the case where conflicting test and non-test cubes need to be merged. Experimental results demonstrated that compact diagnostic test sets are obtained.

Table II. Comparison

	[7]		cube merg		cube merg + [7]	
circuit	tests	indist	tests	indist	tests	indist
s5378	153	525	127	534	137	524
s9234	381	1249	236	1266	259	1235
s13207	322	2043	274	2062	286	2043
s15850	239	2789	150	2943	176	2789
s38417	469	3362	180	3459	249	3361
s38584	752	2696	187	2745	232	2696
b14	465	867	456	420	510	305
b15	493	421	413	548	489	419
b20	554	818	444	874	550	565
aes_core	681	2062	661	2072	668	2062
des_area	139	188	136	189	137	188
pci_spoci_ctrl	173	32	162	46	172	32
spi	610	57	524	116	586	51
systemcaes	178	1450	169	1451	170	1450
tv80	651	426	597	567	645	416

References

[1] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing* and *Testable Design*, IEEE Press, 1995.

[2] P. Camurati, D. Medina, P. Prinetto and M. Sonza Reorda, "A Diagnostic Test Pattern Generation Algorithm", in Proc. Intl. Test Conf., 1990, pp. 52-58.

[3] T. Gruning, U. Mahlstedt and H. Koopmeiners, "DIATEST: A Fast Diagnostic Test Pattern Generator for Combinational Circuits", in Proc. Intl. Conf. on Computer-Aided Design, 1991, pp. 194-197.

[4] F. Corno, P. Prinetto, M. Rebaudengo and M. Sonza Reorda, "GARDA: A Diagnostic ATPG for Large Synchronous Sequential Circuits", in Proc. Europ. Design & Test Conf., 1995, pp. 267-271.

[5] I. Pomeranz and S. M. Reddy, "A Diagnostic Test Generation Procedure for Synchronous Sequential Circuits Based on Test Elimination", in Proc. Intl. Test Conf., 1998, pp. 1074-1083.

[6] A. Veneris, R. Chang, M. S. Abadir and S. Seyedi, "Functional Fault Equivalence and Diagnostic Test Generation in Combinational Logic Circuits Using Conventional ATPG", Journal of Electronic Testing: Theory and Applications (JETTA), vol. 21, no. 5, Oct. 2005, pp. 495-502.

[7] I. Pomeranz and S. M. Reddy, "Output-Dependent Diagnostic Test Generation", in Proc. VLSI Design Conf., 2010, pp. 3-8.

[8] S. Prabhu, M. S. Hsiao, L. Lingappan and V. Gangaram, "A SMT-based Diagnostic Test Generation method for Combinational Circuits", in Proc. VLSI Test Symp., 2012 pp. 215-220.

[9] Y.-H. Chen, C.-L. Chang and C.H.-P. Wen, "Diagnostic Test-pattern Generation Targeting Open-segment Defects and its Diagnosis Flow", IET Computers & Digital Techniques, Vol. 6, No. 3, 2012, pp. 186-193.

[10] I. Pomeranz, "Non-Test Cubes for Test Generation Targeting Hard-to-Detect Faults", IEEE Trans. on Computer-Aided Design, 2013.