

On Test Data Compression and n -Detection Test Sets

Irith Pomeranz¹
 School of Electrical & Computer Eng.
 Purdue University
 W. Lafayette, IN 47907
 pomeranz@ecn.purdue.edu

Sudhakar M. Reddy²
 Electrical & Computer Eng. Dept.
 University of Iowa
 Iowa City, IA 52242
 reddy@engineering.uiowa.edu

Abstract

We consider the relationship between test data compression and the ability to perform comprehensive testing of a circuit under an n -detection test set. The size of an n -detection test set grows approximately linearly with n . Therefore, one may expect a decompressor that can decompress a compressed n -detection test set to be larger than a decompressor required for a compact conventional test set. The results presented in this work demonstrate that it is possible to use a decompressor designed based on a compact one-detection test set in order to apply an n -detection test set. Thus, the design of the decompressor does not have to be changed as n is increased. We describe a procedure that generates an n -detection test set to achieve this result.

Categories and Subject Descriptors

B.8.1 Reliability, Testing and Fault-Tolerance.

General Terms

Reliability.

Keywords

n -detection test sets, test data compression, test generation.

1. Introduction

On-chip decompression of compressed test data stored on a tester was considered in [1]-[7] as a method to reduce test data volume and test application time for scan circuits. A basic decompression scheme based on a combinational decompressor is shown in Figure 1. In the figure, C is a scan circuit with N_S scan chains. Test data is loaded to the circuit through N_I inputs that drive a decompressor circuit D . The decompressor expands every N_I -bit vector into an N_S -bit vector, which is applied to C through its scan chains. If the length of the longest scan chain in C is L , it takes L clock cycles to apply a test t to C . In every clock cycle, an N_I -bit vector r_k is applied to the decompressor,

the decompressor produces an N_S -bit vector v_k , and the scan chains are shifted by one position with v_k as the new scan-in vector. After L clock cycles, the vectors v_0, \dots, v_{L-1} define an input pattern of C . For a given test t , the vectors r_0, \dots, r_{L-1} , and the corresponding vectors v_0, \dots, v_{L-1} are selected such that t is applied to C . Using a decompressor reduces the test data volume since only $N_I < N_S$ bit vectors need to be supplied by the tester for each N_S -bit vector applied to the circuit. Additionally, test application time is also reduced since the number of internal scan chains can be chosen to be high enough to obtain the desired test application time.

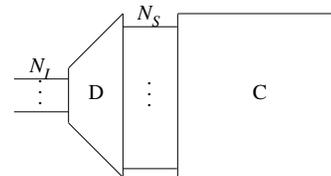


Figure 1: A basic decompression scheme

Given a test set T that detects a set of faults F in a circuit C , in order to design the decompressor D , it is first necessary to find a set of vectors V that the decompressor will be able to apply to C , and satisfies the following condition. Using sequences of the form $v_{j_0} \dots v_{j_{(L-1)}}$, where $v_{j_k} \in V$ for $0 \leq k < L$, it should be possible to form tests that detect all the faults in F . The size of V determines the minimum number of decompressor inputs, which is given by the equation $N_I = \lceil \log_2 |V| \rceil$. Thus, it is important to minimize the size of V in order to minimize N_I and thus minimize the test data volume by reducing the number of bits per vector applied to the decompressor. The reduction of test data volume allows a reduction of external tester storage. This has been one of the primary goals of using on-chip decompressors for test application [1]-[7].

Our goal in this work is to consider the relationship between test data compression and the ability to perform comprehensive testing of a circuit under an n -detection test set. An n -detection test set is one where each modeled fault is detected either by n different tests, or by the maximum number of different tests that can detect the fault if this number is smaller than n . In various types of experiments reported in [8]-[12], n -detection test sets were shown to be useful in achieving high defect coverage and high coverage of unmodeled faults. The advantage of using n -detection test sets over other approaches for generating test sets with high defect coverage is that an existing test generation procedure for a gate-level fault model (such as stuck-at faults) can be used with simple extensions for generating n -detection test sets, thus avoiding the need to develop new test generation procedures for new fault or defect models.

1. Research supported in part by NSF Grant No. CCR-0098091 and in part by SRC Grant No. 2001-TJ-950.

2. Research supported in part by NSF Grant No. CCR-0097905 and in part by SRC Grant No. 2001-TJ-949.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 2003, June 2-6, 2003, Anaheim, California, USA.
 Copyright 2003 ACM 1-58113-688-9/03/0006...\$5.00.

The size of an n -detection test set grows approximately linearly with n [9]. Thus, the use of on-chip decompressors to reduce test application time and test data volume may become essential in reducing the costs of using n -detection test sets [6]. When an n -detection test set is applied through a decompressor, one may expect the number of vectors that must be included in the set V to be increased as n is increased, thus increasing the number of decompressor inputs and its complexity. However, we observe that even if V is computed based on a test set T that detects each target fault only once, it may be possible to construct additional tests using only vectors in V . An increase in the numbers of detections is then achieved without increasing the size of V , and thus the decompressor design is not changed. This observation leads us to study the following problem.

Constrained n -detection test generation: Given a test set T comprised of a minimal set of vectors V , generate a test set \hat{T} such that (1) every test $t \in \hat{T}$ is comprised only of vectors out of V ; and (2) every target fault is detected n times by \hat{T} , or the maximum possible number of times.

An alternative approach to allow the application of an n -detection test set through a decompressor is to start from an n -detection test set T^n and find a minimal set of vectors V^n sufficient for generating tests to detect all the target faults the same numbers of times as they are detected by T^n . The decompressor can then be designed based on V^n . In this approach, it is not guaranteed that $V^n = V$ would be found. We avoid this difficulty by starting from a decompressor based on a compact one-detection test set T , and designing T^n based on the same decompressor. We ensure in this way that the size of the decompressor would not increase due to the need to apply an n -detection test set.

In Section 2 we describe the decompression scheme we use. In Section 3 we describe the generation of an n -detection test set using only vectors out of a given set V derived from a one-detection test set T . Experimental results of the procedure described in Section 3 are given in Section 4.

2. The decompression scheme

In this section, we describe briefly the decompression scheme we use and the derivation of the set of vectors V that the decompressor can apply to the circuit.

2.1. Preliminaries

The set V is computed based on a one-detection test set T that detects a set of faults F . Given the test set T , we first partition T into scan vectors. We pad certain vectors with unspecified values if some of the scan chains are shorter than others. A test t_j is thus divided into vectors $v_{j0}, v_{j1}, \dots, v_{j(L-1)}$.

The set of vectors comprising all the tests in T defines the initial contents of the set V based on which the decompressor D will be designed. The set V is sufficient for creating every test in T by applying one or more vectors out of V consecutively. The decompressor circuit applies the appropriate vectors out of V based on the encoded vectors applied to its inputs.

To allow compression of every vector $v \in V$ into a decompressor input vector r , the decompressor must have at least $\lceil \log_2 |V| \rceil$ inputs. Thus, we have $N_I \geq \lceil \log_2 |V| \rceil$ and the size of V determines the number of decompressor inputs. Our goal, similar to [4], is to minimize the size of V in order to minimize N_I and thus reduce the number of bits in each vector applied to the decompressor. This will reduce the size of the compressed test data that needs to be stored externally. It also reduces the size of the decompressor due to two effects. (1) If V is reduced such that N_I can be reduced, the decompressor has

fewer inputs, which typically implies a smaller size. (2) Even for the same value of N_I , a smaller set V implies more don't-cares on the outputs of D corresponding to vectors that do not need to be compressed.

The size of V is largely determined by the test set T . It is possible to minimize V by using the techniques discussed next.

2.2. Compatible vectors

Some of the vectors in V have unspecified values. As a result, V may contain compatible vectors. For example, if V contains the vectors $00x0$ and $000x$, it is possible to use 0000 instead of both vectors. This reduces the size of V by one. To achieve this reduction, T is modified such that $00x0$ and $000x$ are replaced by 0000 .

To construct a minimal set of vectors V based on a given test set T , we add the vectors comprising the tests of T into V one at a time. Initially, $V = \emptyset$. When a new vector v_{jk} is inserted into V , we find the first vector v already in V , which is compatible with v_{jk} . If v does not exist, v_{jk} is added to V as a new vector. Otherwise, the vector obtained by merging v_{jk} and v is added to V to replace v . T is modified to reflect the merger.

2.3. Modifying T to minimize V

In [4], as many values in T as possible are unspecified before V is defined. This maximizes the number of compatible vector pairs in V , thus allowing its size to be reduced. However, the process of unspecified values is applied before V is constructed and values are unspecified even if it does not help merge any vectors in V . We use the following two-phase solution instead.

We first attempt to replace scan vectors that appear only once in T by other vectors out of T . Thus, if a test $t_j \in T$ consists of scan vectors $v_{j0}, v_{j1}, \dots, v_{j(L-1)}$ and v_{jk} appears only once in T (i.e., v_{jk} appears only in t_j and only once in t_j), we attempt to replace v_{jk} in t_j by a different vector \hat{v} that appears in T . To accept the replacement of a vector $v \in V$ by a vector $\hat{v} \in V$ in a test $t \in T$, we check that all the faults detected by the original test t are also detected by the modified test t . Every time we replace a vector v by a vector \hat{v} , we reduce the number of vectors in V by one.

As replacement vectors \hat{v} , we first consider vectors that appear more than once in T . In this way, we do not increase the number of appearances of vectors that appear only once in T , and we maximize our ability to remove them. Only after all the possible replacements using such vectors \hat{v} have been made, we consider replacement vectors that appear only once in T .

In the second phase of the procedure, we consider pairs of vectors v_1 and v_2 , each appearing only once in T . We attempt to replace both v_1 and v_2 together by a new vector \hat{v} , which may or may not appear in V . Every time we replace a pair of vectors by a new vector, we remove two vectors from V and add at most one. Thus, the size of V is reduced.

For a given pair of vectors v_1, v_2 , we generate replacement vectors as follows. First, the *intersection* vector v_c of v_1 and v_2 is computed. The intersection vector is equal to v_1 and v_2 in positions where they are equal, and it is x in other positions. The number of x values in v_c is the *distance* between v_1 and v_2 , and we denote it by δ . If δ does not exceed a preselected constant Δ_{EXH} , we define 2^δ vectors \hat{v} based on v_c , one for every specification of the x values in v_c . If δ exceeds Δ_{EXH} but does not exceed another constant, Δ , we define $2^{\Delta_{EXH}}$ vectors \hat{v} based on v_c by randomly filling the x values in v_c $2^{\Delta_{EXH}}$ times.

3. n -detection test sets

In this section and the next one, we study the ability of a decompressor based on a one-detection test set to apply an n -detection test set for $n > 1$. For this purpose, we describe in this section the derivation of an n -detection test set \hat{T} using only vectors out of a given set V .

The set V is the one obtained in Section 2 from a compacted one-detection test set T . The constant n is assumed to be given, as well as the set V and the modified test set T that corresponds to V . The n -detection test set \hat{T} will be applicable to the circuit using the same decompressor used for applying the modified test set T . If an n -detection test set cannot be derived, our goal is to come as close to such a test set as possible, i.e., our goal is for \hat{T} to detect each fault a number of times which is as close to n as possible.

We associate with every fault $f \in F$ (i.e., every fault detected by T) the number of times f is detected by \hat{T} . This number is denoted by $n_{det}(f)$. Initially, $\hat{T} = \emptyset$ and $n_{det}(f) = 0$ for every $f \in F$. We also associate with f the set of tests in \hat{T} that detect f , denoted by $T_{det}(f)$. Initially, $T_{det}(f) = \emptyset$.

Since T can be generated using only vectors from V , and since T already detects every fault in F at least once, we use T as part of the test set \hat{T} . We add every $t \in T$ to \hat{T} , and we fault simulate t using the n -detection fault simulation process given below as Procedure 1 in order to update $n_{det}(f)$ and $T_{det}(f)$. Note that Procedure 1 simulates a fault f only until it is detected n times by \hat{T} . It thus provides the accurate number of detections of a fault f which is detected fewer than n times by \hat{T} . If f is detected n times or more, $n_{det}(f) = n$. This is consistent with our goal of generating an n -detection test set.

Procedure 1: n -detection fault simulation of a test t

For every fault $f \in F$ such that $n_{det}(f) < n$:

Fault simulate f under t . If f is detected by t , set $n_{det}(f) = n_{det}(f) + 1$ and $T_{det}(f) = T_{det}(f) \cup \{t\}$.

In order to complete \hat{T} into an n -detection test set (or come as close to such a test set as possible), we first consider every fault $f \in F$ such that $n_{det}(f) = 1$. For every such fault, we attempt to generate a new test $t_w \notin \hat{T}$ using only vectors from V . If a test t_w is generated, it is added to \hat{T} and the numbers of detections of all the faults are updated by calling Procedure 1 with t_w as input. After considering all the faults with $n_{det}(f) = 1$, we consider all the faults with $n_{det}(f) = 2$ and apply the same process. We continue to consider faults with increasing numbers of detections until we have considered all the faults with $n_{det}(f) \leq n - 1$.

At an arbitrary step of this process, we consider a fault f with $n_{det}(f) = m$ and $T_{det}(f) = \{t_1(f), t_2(f), \dots, t_m(f)\}$. It is possible to use a deterministic test generation procedure to generate a test $t \notin T_{det}(f)$ for f by constraining the test generation procedure to use only vectors out of V and avoid tests that are already in $T_{det}(f)$. We use a simpler simulation-based procedure, described next.

We make several attempts to generate a new test t_w for f until a test is obtained or until all the options have been exhausted. The various options are the following.

We try every one of $t_1(f), t_2(f), \dots, t_m(f)$ as a basis for computing t_w . When $t_i(f)$ is considered, we initially set $t_w = t_i(f)$. We then try to modify t_w in order to obtain a new test that detects f . The motivation for using tests out of $T_{det}(f)$ is that, since these tests already detect f , it is likely that we will be able to obtain new tests that detect f by making modifications to these tests.

We partition t_w into $v_{w0}v_{w1} \dots v_{w(L-1)}$. For every $0 \leq k < L$, we try every one of the vectors in V as a replacement for v_{wk} . When v_{wk} is replaced by $v_p \in V$, we obtain the test $t_w = v_{w0} \dots v_{w(k-1)}v_p v_{w(k+1)} \dots v_{w(L-1)}$. If $t_w \notin \hat{T}$, we simulate f under t_w . If f is detected, we accept t_w as a new test.

Every time we add a new test t_w to \hat{T} , we update $n_{det}(\hat{f})$ and $T_{det}(\hat{f})$ for every $\hat{f} \in F$. Thus, a fault \hat{f} may be detected by a new test added for another fault, f . This will increase the number of detections of \hat{f} , and may postpone or even eliminate its consideration during the n -detection test generation process. If \hat{f} is considered, its set $T_{det}(\hat{f})$ will contain t_w in addition to all the tests in T that detect it.

The procedure above is referred to as Procedure 2 and demonstrated next using $s27$. We consider $n = 3$ for $s27$. We initially set $\hat{T} = T$, where T is a test set obtained after applying the procedure of Section 2, shown in Table 1. We perform 3-detection fault simulation of this test set and find that 13 faults are detected once, eight faults are detected twice, and the remaining faults are detected three times or more. We first consider faults detected once under \hat{T} . The first fault we consider is f_1 , and it is detected only by $t_3 = 01110000$. We consider each one of the two vectors of t_3 for replacement by each one of the five vectors of V shown in Table 1. The options we consider are summarized in Table 2. In the last columns of Table 2 we indicate whether or not the test is new (not already included in \hat{T}). If it is new, we also indicate whether or not it detects f_1 . We stop when we find the first new test that detects f_1 by replacing v_{w1} with v_1 . This test is added to \hat{T} as t_6 .

Table 1: A test set for $s27$

(a) The set V		(b) The test set T		
i	v_i	j	v_{j0}	v_{j1}
0	0000	0	0000	0111
1	0111	1	1001	0100
2	1001	2	0100	1101
3	0100	3	0111	0000
4	1101	4	1101	0111
		5	0000	0000

Table 2: Options for detecting f_1

k	p	t_w	new	det
0	0	0000 0000	no	
0	1	0111 0000	no	
0	2	1001 0000	yes	no
0	3	0100 0000	yes	no
0	4	1101 0000	yes	no
1	0	0111 0000	no	
1	1	0111 0111	yes	yes

When t_6 is simulated, we find that it increases the numbers of detections of three faults detected once (including f_1) and three faults detected twice by \hat{T} . At the end of the test generation process, \hat{T} contains 17 tests.

Procedure 2 is not guaranteed to find n different tests for a fault. This may happen when the fault does not have n different tests, or because the procedure does not explore all the options for detecting every fault. In the latter case, it may be possible to find additional tests by replacing more than a single vector in every test, or by performing deterministic test generation. However, we found that in benchmark circuits the proposed procedure is sufficient for all the faults.

Considering the variety of tests included in \hat{T} , we observe the following. To obtain a new test t_w , we replace an N_S -bit vector v_{wk} of t_w by a different N_S -bit vector. As a result, the new test differs from an existing test in up to N_S bits. Thus, non-trivial changes are introduced into the new tests included in \hat{T} .

4. Results of n -detection test generation

We applied Procedure 2 using $n = 3$ to ISCAS-89 and ITC-99 benchmark circuits with different numbers of scan chains. We used compacted one-detection test sets for T . These were modified by the procedure of Section 2.

The results are given in Table 3. After the circuit name and the number of scan chains, we show the following information for T and, on a separate row, for \hat{T} . We first show the test set size (column *size*). We then show the number of faults detected by the test set once (column $n=1$), the number of faults detected twice (column $n=2$), and the number of faults detected three times or more (column $n=3$).

Table 3: Results of 3-detection test generation

circuit	NS	set	size	n=1	n=2	n=3
s1423	8	T	26	254	177	1070
		\hat{T}	98	0	0	1501
s1423	4	T	26	291	164	1046
		\hat{T}	88	0	0	1501
s5378	16	T	100	520	316	3727
		\hat{T}	309	0	0	4563
s5378	8	T	100	357	361	3845
		\hat{T}	307	0	0	4563
s9234	16	T	111	1207	680	4588
		\hat{T}	463	0	0	6475
s9234	8	T	111	1043	840	4592
		\hat{T}	448	0	0	6475
s13207	64	T	235	1753	553	7358
		\hat{T}	856	0	0	9664
s13207	32	T	235	1853	653	7158
		\hat{T}	735	0	0	9664
s15850	32	T	97	1441	824	9071
		\hat{T}	432	0	0	11336
s15850	16	T	97	1566	852	8918
		\hat{T}	389	0	0	11336
s38417	128	T	87	2648	2150	26217
		\hat{T}	552	0	0	31015
s38417	64	T	87	2543	1886	26586
		\hat{T}	363	0	0	31015
b14	16	T	194	268	418	7527
		\hat{T}	451	0	0	8213
b14	8	T	194	273	355	7585
		\hat{T}	501	0	0	8213
b20	32	T	335	520	726	18475
		\hat{T}	780	0	0	19721
b20	16	T	335	467	714	18540
		\hat{T}	798	0	0	19721

From Table 3 it is possible to use vectors out of V , which is generated based on a compact one-detection test set T , in order to detect each fault three times. Thus, the use of three-detection test sets does not have to increase the number of distinct vectors the decompressor needs to handle; the same decompressor designed for a compact one-detection test set is also useful for applying a three-detection test set. The number of tests in the three-detection test set \hat{T} is about three times the number of tests in the one-detection test set in most cases.

We also applied Procedure 2 using $n = 10$ and the first value of N_S for every circuit. We found that it is possible to detect each fault 10 times using only vectors out of V .

The run time of Procedure 2 normalized to the time it takes to fault simulate the compacted one-detection test set T is reported in Table 4. In Table 4, $NS1$ stands for the first number of scan chains, and $NS2$ stands for the second number of scan chains for every circuit.

Table 4: Run time of Procedure 2

circuit	NS1	NS1	NS2
	n=3	n=10	n=3
s1423	4.28	18.13	3.81
s5378	4.02	15.79	3.69
s9234	3.19	13.34	3.08
s13207	3.20	14.32	3.24
s15850	4.42	22.85	2.94
s38417	8.05	32.03	4.77
b14	0.99	3.67	1.00
b20	1.06	4.15	1.13

5. Concluding remarks

The size of an n -detection test set grows approximately linearly with n . Therefore, one may expect a decompressor that can apply an n -detection test set to be larger than one required for a conventional compacted one-detection test set. To study this issue, we described a procedure that accepts a set of vectors V corresponding to a compacted one-detection test set T applicable through a decompressor. The procedure constructs an n -detection test set using only the vectors in V . Through this procedure, we demonstrated that it is possible to use a decompressor designed based on a compacted one-detection test set in order to apply an n -detection test set. Thus, we showed that it is not necessarily needed to use a larger decompressor for an n -detection test set as n is increased.

References

- [1] A. Jas, J. Ghosh-Dastidar and N. A. Touba, "Scan Vector Compression/Decompression Using Statistical Coding", in Proc. VLSI Test Symp., 1999, pp. 114-120.
- [2] A. Chandra and K. Chakrabarty, "Frequency-directed Run-length (FDR) Codes with Application to System-on-a-chip Test Data Compression", in Proc. VLSI Test Symp., April 2001, pp. 42-47.
- [3] I. Bayraktaroglu and A. Orailoglu, "Test Volume and Application Time Reduction Through Scan Chain Concealment", in Proc. Design Autom. Conf., June 2001, pp. 151-155.
- [4] S. M. Reddy, K. Miyase, S. Kajihara and I. Pomeranz, "On Test Data Volume Reduction for Multiple Scan Chain Designs", in Proc. VLSI Test Symp., April 2002, pp. 103-108.
- [5] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller and B. Koenemann, "OPMISR: The Foundation for Compressed ATPG Vectors", in Proc. Intl. Test Conf., Oct. 2001, pp. 748-757.
- [6] A. Chandra and M. Chakrabarty, "Test Resource Partitioning and Reduced Pin-Count Testing based on Test Data Compression", in Proc. Design Autom. and Test in Europe Conf., 2002, pp. 598-603.
- [7] J. Rajski, J. Tyszer, M. Kassab, N. Kukherjee, R. Thompson, K.-H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide and J. Qian, "Embedded Deterministic Test for Low Cost Manufacturing Test", in Proc. Intl. Test Conf., 2002, pp. 301-310.
- [8] S. C. Ma, P. Franco and E. J. McCluskey, "An Experimental Chip to Evaluate Test Techniques Experiment Results", in Proc. 1995 Intl. Test Conf., Oct. 1995, pp. 663-672.
- [9] S. M. Reddy, I. Pomeranz and S. Kajihara, "Compact Test Sets for High Defect Coverage", IEEE Trans. on Computer-Aided Design, Aug. 1997, pp. 923-930.
- [10] J. T.-Y. Chang, C.-W. Tseng, C.-M. J. Li, M. Purtell and E. J. McCluskey, "Analysis of Pattern-Dependent and Timing-Dependent Failures in an Experimental Test Chip", in Proc. 1998 Intl. Test Conf., Oct. 1998, pp. 184-193.
- [11] M. R. Grimaila, S. Lee et. al., "REDO - Random Excitation and Deterministic Observation - First Commercial Experiment", in Proc. 17th VLSI Test Symp., April 1999, pp. 268-274.
- [12] I. Pomeranz and S. M. Reddy, "On n -Detection Test Sets and Variable n -Detection Test Sets for Transition Faults", in Proc. 17th VLSI Test Symp., April 1999, pp. 173-179.